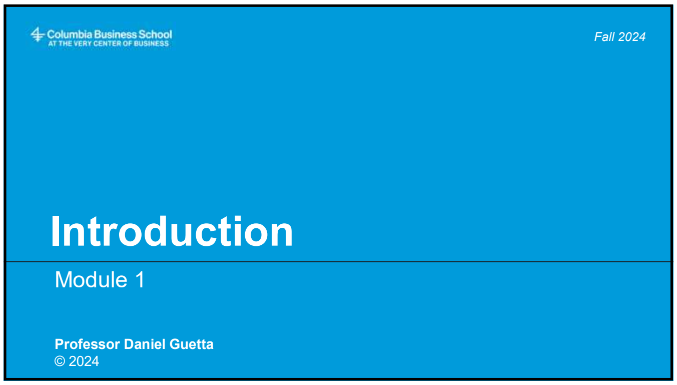
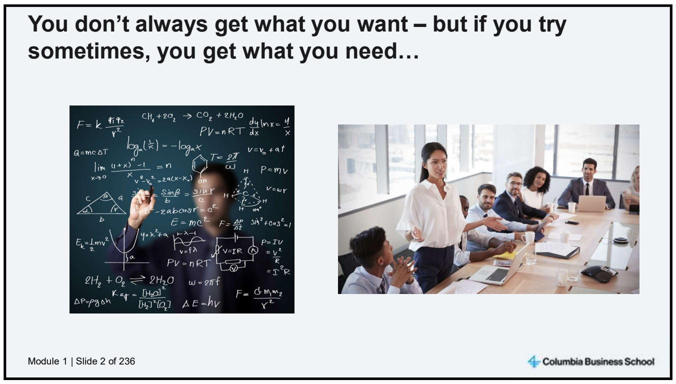

Columbia Business School
AT THE VERY CENTER OF BUSINESS

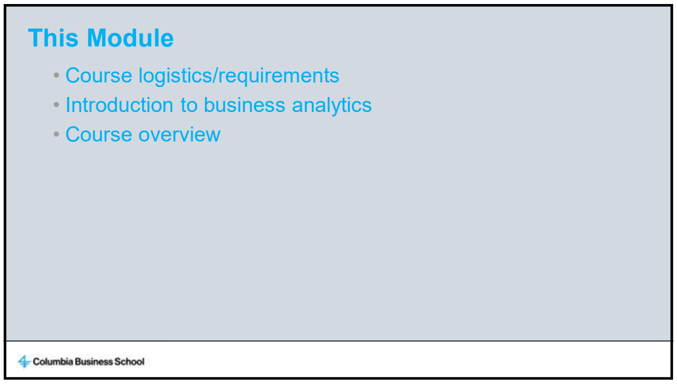
Fall 2024

Introduction

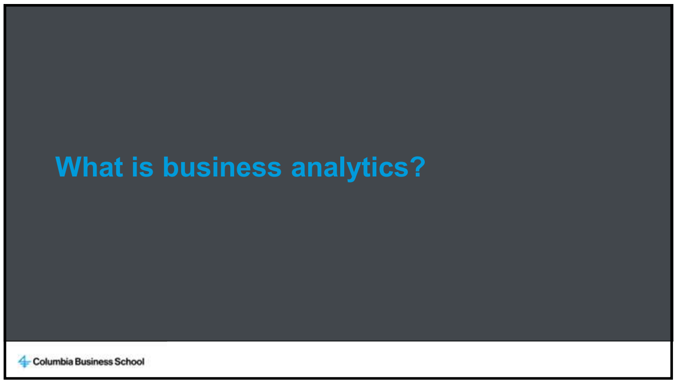
Module 1

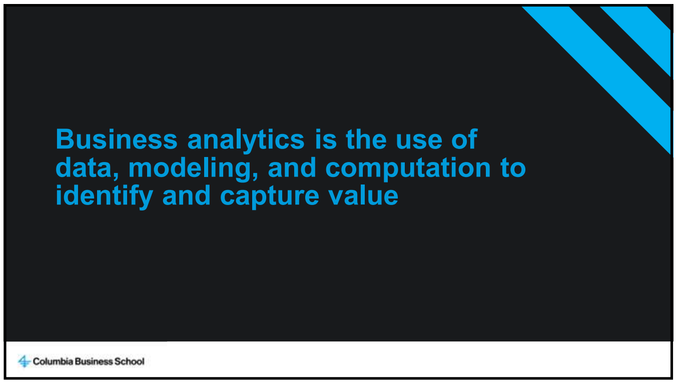
Professor Daniel Guetta
© 2024

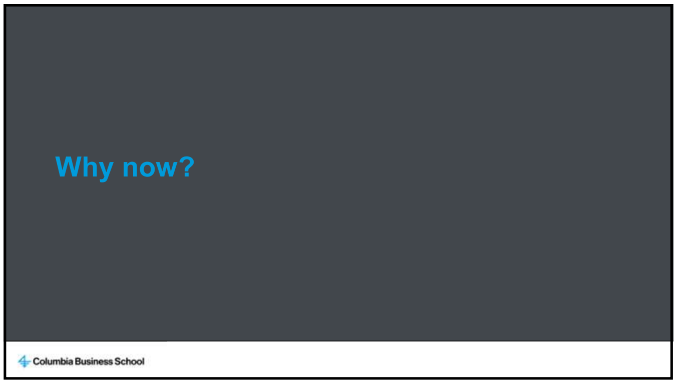

You don't always get what you want – but if you try sometimes, you get what you need…


This Module

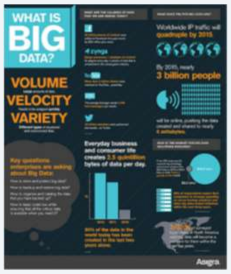
- Course logistics/requirements
- Introduction to business analytics
- Course overview


What is business analytics?

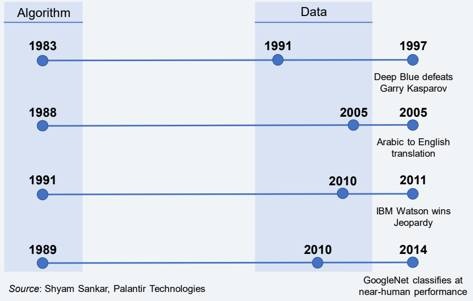

Business analytics is the use of data, modeling, and computation to identify and capture value


Why now?

Available data is exploding



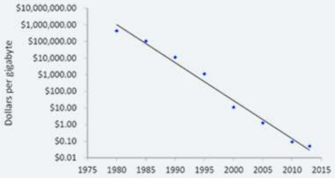
The importance of data



Algorithm	Data	
1983	1991	1997 Deep Blue defeats Garry Kasparov
1988	2005	2005 Arabic to English translation
1991	2010	2011 IBM Watson wins Jeopardy
1989	2010	2014 GoogleNet classifies at near-human performance

Source: Shyam Sankar, Palantir Technologies

Storage cost is plummeting



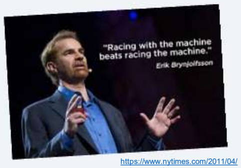
- Complete works of Shakespeare: 5 megabytes
- One DVD: 17 gigabytes (1000 megabytes in a gigabyte)
- US library of congress (print): 10 terabytes (1000 gigabytes in a terabyte)
- A terabyte hard drive now costs about $50

Storage cost is plummeting

In the "I'm getting old" department.., a kid saw this and said, "oh, you 3D-printed the 'Save' Icon."

This is brutal.

The stakes are huge



https://www.nytimes.com/2011/04/24/business/24unboxed.html

MIT/Wharton study (2011, Brynjolfsson et. al.)

- Study of 179 large publicly traded firms
- Firms that emphasize data driven decision making (DDD) and business analytics perform significantly better
- Output and productivity 5-6% higher after adjusting for other factors

Early reports on economic potential of business analytics

McKinsey&Company
McKinsey Global Institute
Big data: The next frontier for innovation, competition, and productivity

- McKinsey report (2011): assessment of potential value
  - US health care: $300B annual savings
  - European public sector: €250B annual value
  - Global retailing: 60% increase in operating margins
  - Personal location data: $600B added consumer surplus

Bottleneck: analytics talent, not data

The need to be well-rounded

McKinsey&Company
McKinsey Analytics

Article
Harvard Business Review
February 2018

Analytics translator: The new must-have role

By Nicolaus Henke, Jordan Levine, and Paul McInerney

https://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/analytics-translator

“The search for vital analytics talent has often focused on data scientists. In this article, we describe the overlooked analytics role that’s even more critical to fill.”

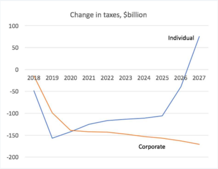
“In many organizations, data professionals and business leaders often struggle to articulate their needs in a language that the other can execute on.”

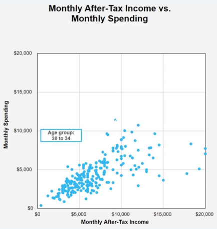
The need to be well-rounded

AGGRESSIVE
Personal Injury Attorneys
Smith & Liggett, P.C.
ATTORNEYS AT LAW
Free Consultations

Believe in something. Even if it means sacrificing everything

The need to be well-rounded





Your unique position

- You are **engineers** at a **business school**
- This puts you in the position to be a super-analytics translator – someone who can not only **understand** the analytics, but **do** it too
- Whether you intend to be closer to the **analytics side** or closer to the **business side**, you can bring both sides together
  - The impact of doing this well can be enormous
- Demand for this combined set of skills is exploding

Four high level goals

- Help you **think critically** about data and the analyses based on those data
- **Identify opportunities** for creating value using business analytics
- Teach you **essential tools and theory** so you can apply these methods yourselves
  - Our focus will be on **deeply understanding** the methods rather than rigorous proofs – but we **will** develop **real** understanding
- Teach you how to **talk about** these concepts to less technical audiences

Columbia Business School
AT THE VERY CENTER OF BUSINESS

An interesting example

Target targeting mothers-to-be

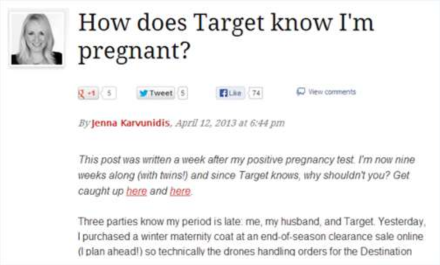
Andrew Pole, Target analytics: asked to identify pregnant women (2002)

What did he do?

https://www.nytimes.com/2012/02/19/magazine/shopping-habits.html

Why target pregnant women?

Negative reactions



How does Target know I'm pregnant?

By Jenna Karvunidis, *April 12, 2013 at 6:44 pm*

This post was written a week after my positive pregnancy test. I'm now nine weeks along (with twins!) and since Target knows, why shouldn't you? Get caught up here and here.

Three parties know my period is late: me, my husband, and Target. Yesterday, I purchased a winter maternity coat at an end-of-season clearance sale online (I plan ahead!) so technically the drones handling orders for the Destination

Colbert report

A matter of framing?

"With the pregnancy products, though, we learned that some women react badly," the executive said. "Then we started mixing in all these ads for things we knew pregnant women would never buy, so the baby ads looked random. We'd put an ad for a lawn mower next to diapers. We'd put a coupon for wineglasses next to infant clothes. That way, it looked like all the products were chosen by chance… As long as we don't spook her, it works."

"We are very conservative about compliance with all privacy laws. But even if you're following the law, you can do things where people get queasy".

Target executive to the New York Times

Business impact

to our shareholders

[illegible]

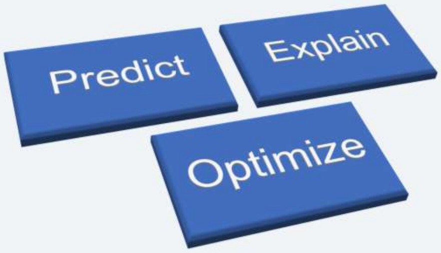
- Strong revenue growth from $44B in 2002 to $67B in 2010
- CEO Steinhafel: results due to **"heightened focus on items and categories that appeal to specific guest segments such as mom and baby."**

Shift in global privacy norms

- Regulations emerging
  - EU General Data Protection Regulation (GDPR)
  - California Consume Privacy Act (CCPA)
- Rights of consumers
  - To obtain their data
  - To prevent the sale of personal data to other parties
  - To be forgotten
- Slow change in norms

How is it done? The course plan…

Three elements of AI



Predictive analytics

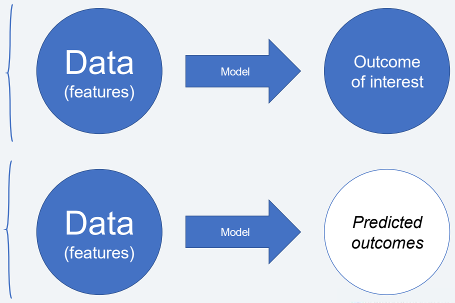
Predictive analytics is about *predicting future outcomes* based on data about *past outcomes*. Predictive analytics use cases form the bulk of the goldmine, and will be the focus of the class

The past data is called *training data* and this is often called *supervised learning*

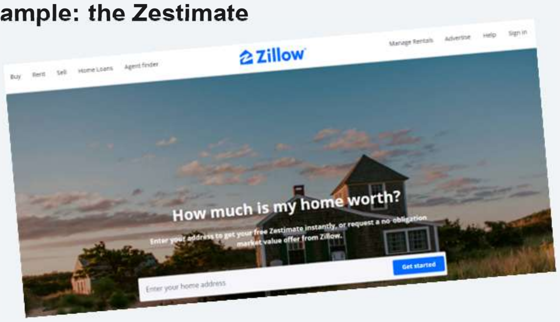
Predictive analytics

Use *past* data with *known* outcomes to *train* a model

Make predictions for new data with unknown outcome. This is known as *inference*

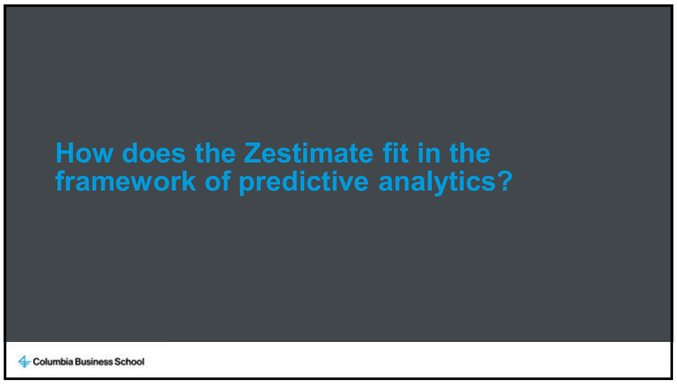


Example: the Zestimate

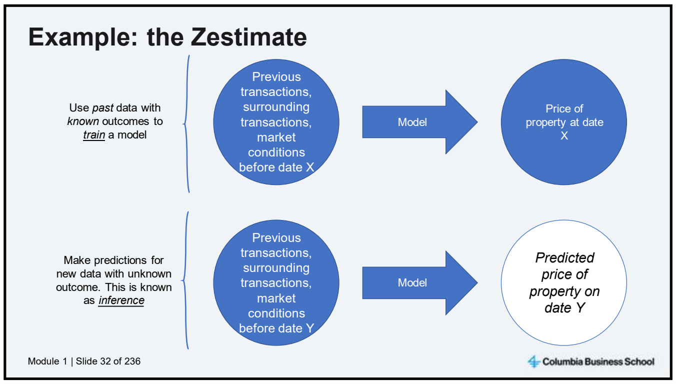


How much is my home worth?

Enter your address to get your free Zestimate instantly, or request a no obligation market value offer from Zillow.



How does the Zestimate fit in the framework of predictive analytics?



Example: the Zestimate

Use *past* data with *known* outcomes to *train* a model

Previous transactions, surrounding transactions, market conditions before date X

Model

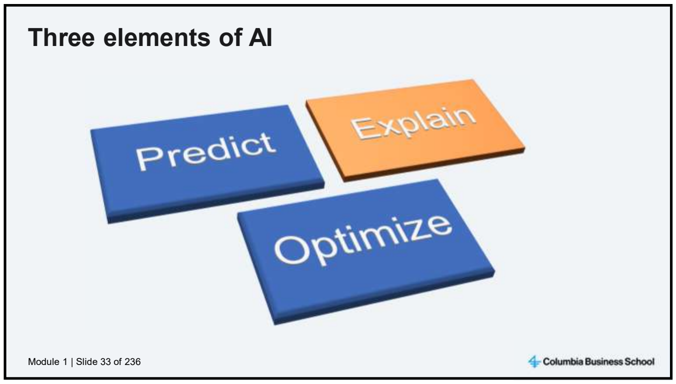
Price of property at date X

Make predictions for new data with unknown outcome. This is known as *inference*

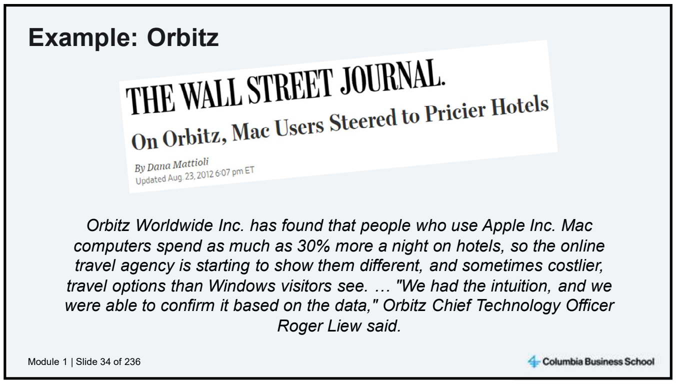
Previous transactions, surrounding transactions, market conditions before date Y

Model

Predicted price of property on date Y



Three elements of AI



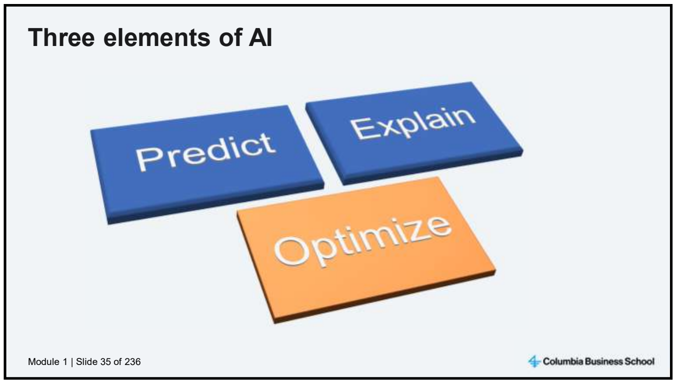
Example: Orbitz

THE WALL STREET JOURNAL.

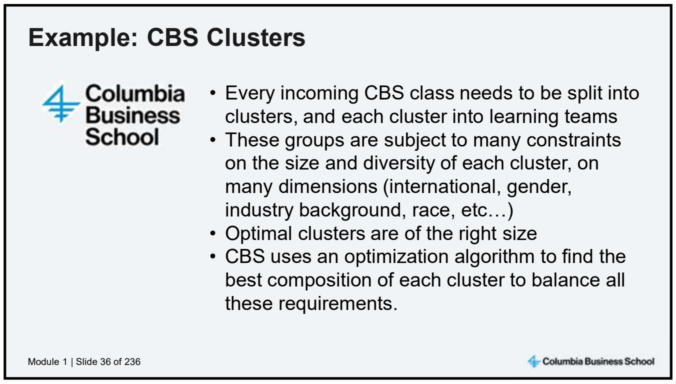
On Orbitz, Mac Users Steered to Pricier Hotels

By Dana Mattioli
Updated Aug. 23, 2012 6:07 pm ET

Orbitz Worldwide Inc. has found that people who use Apple Inc. Mac computers spend as much as 30% more a night on hotels, so the online travel agency is starting to show them different, and sometimes costlier, travel options than Windows visitors see. … "We had the intuition, and we were able to confirm it based on the data," Orbitz Chief Technology Officer Roger Liew said.



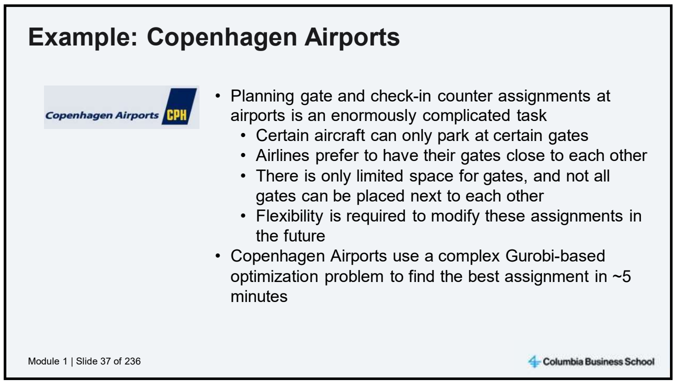
Three elements of AI



Example: CBS Clusters

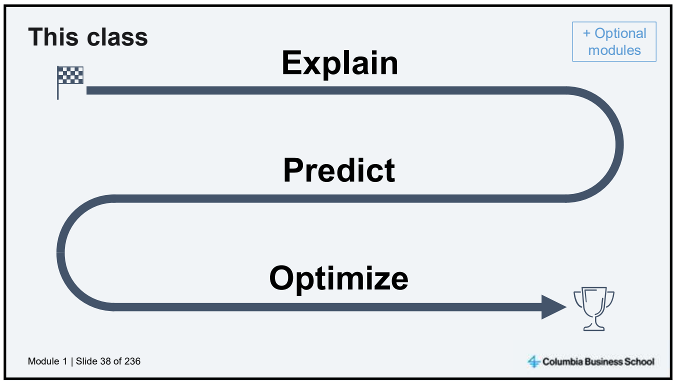
Columbia Business School

- Every incoming CBS class needs to be split into clusters, and each cluster into learning teams
- These groups are subject to many constraints on the size and diversity of each cluster, on many dimensions (international, gender, industry background, race, etc…)
- Optimal clusters are of the right size
- CBS uses an optimization algorithm to find the best composition of each cluster to balance all these requirements.



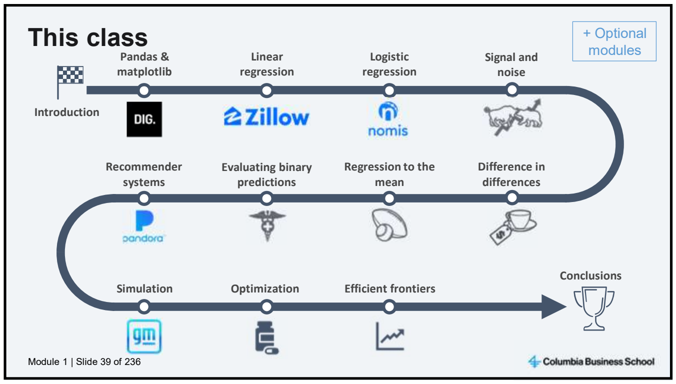
Example: Copenhagen Airports

- Planning gate and check-in counter assignments at airports is an enormously complicated task
  - Certain aircraft can only park at certain gates
  - Airlines prefer to have their gates close to each other
  - There is only limited space for gates, and not all gates can be placed next to each other
  - Flexibility is required to modify these assignments in the future
- Copenhagen Airports use a complex Gurobi-based optimization problem to find the best assignment in ~5 minutes



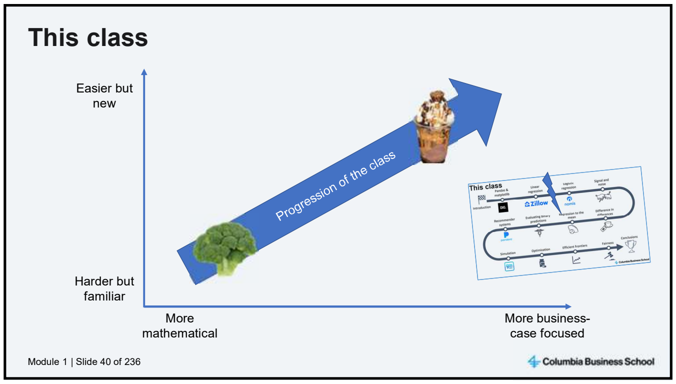
This class

+ Optional modules

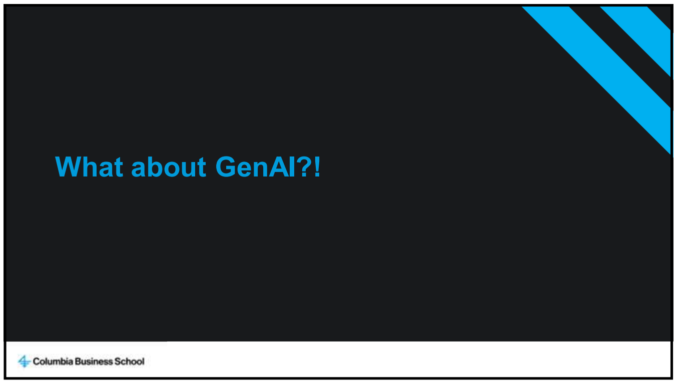


This class

+ Optional modules



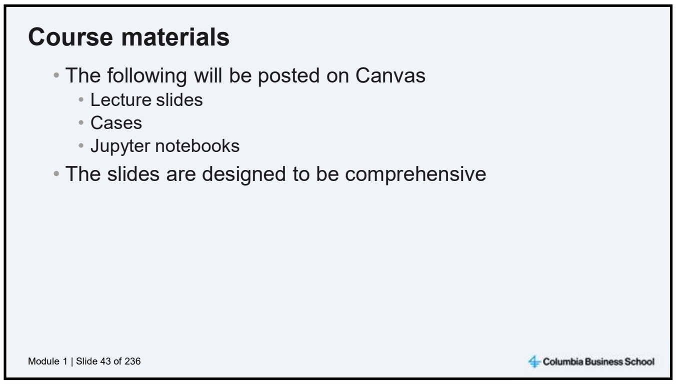
This class



What about GenAI?!

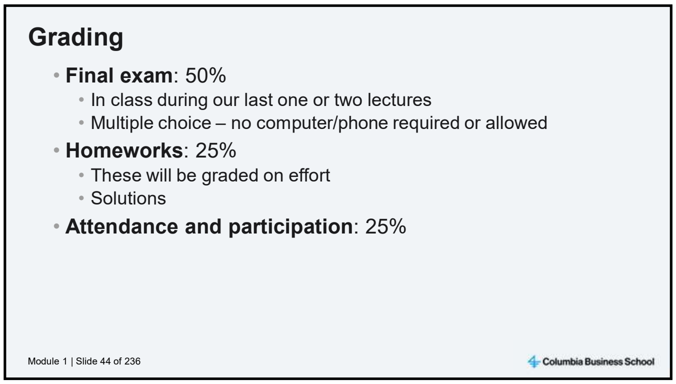
Columbia Business School
AT THE VERY CENTER OF BUSINESS

Logistics 🥱



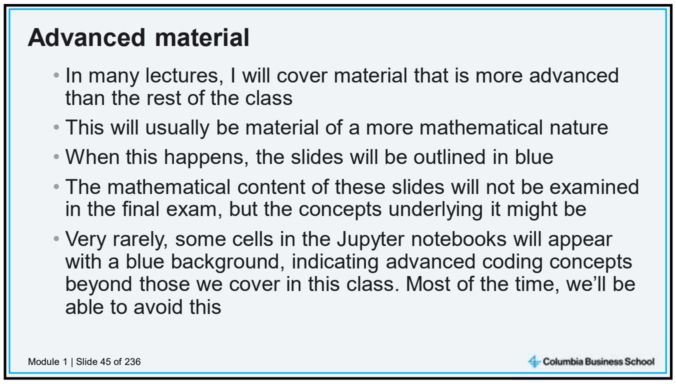
Course materials

- The following will be posted on Canvas
  - Lecture slides
  - Cases
  - Jupyter notebooks
- The slides are designed to be comprehensive



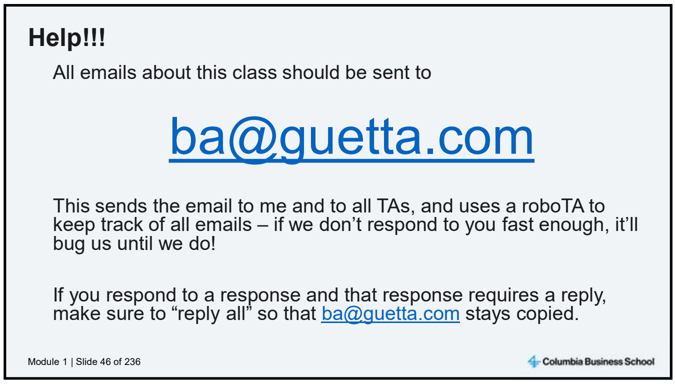
Grading

- **Final exam**: 50%
  - In class during our last one or two lectures
  - Multiple choice – no computer/phone required or allowed
- **Homeworks**: 25%
  - These will be graded on effort
  - Solutions
- **Attendance and participation**: 25%



Advanced material

- In many lectures, I will cover material that is more advanced than the rest of the class
- This will usually be material of a more mathematical nature
- When this happens, the slides will be outlined in blue
- The mathematical content of these slides will not be examined in the final exam, but the concepts underlying it might be
- Very rarely, some cells in the Jupyter notebooks will appear with a blue background, indicating advanced coding concepts beyond those we cover in this class. Most of the time, we'll be able to avoid this



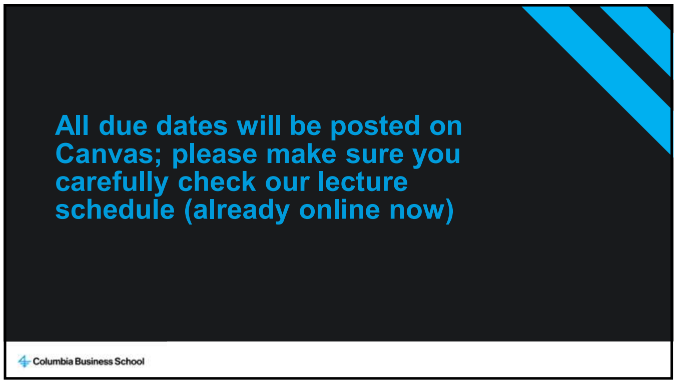
Help!!!

All emails about this class should be sent to

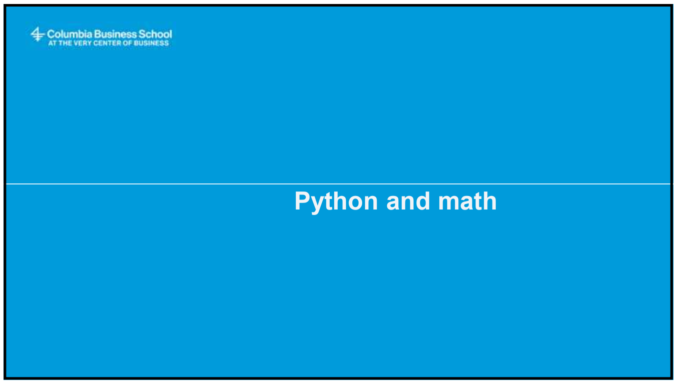
ba@guetta.com

This sends the email to me and to all TAs, and uses a roboTA to keep track of all emails – if we don't respond to you fast enough, it'll bug us until we do!

If you respond to a response and that response requires a reply, make sure to "reply all" so that ba@guetta.com stays copied.



All due dates will be posted on Canvas; please make sure you carefully check our lecture schedule (already online now)



Columbia Business School
AT THE VERY CENTER OF BUSINESS

Python and math

Doing business analytics without coding and math is like playing Chopin with oven mitts

Python and math

- **Coding** and **mathematics** are the workhorses of Business Analytics
- But they are **not** what this class is about
- That said, there's going to be no way around knowing **some** coding/math to appreciate what we're doing in this class
- You will find the first 3-4 lectures will be *much* heavier on the mathematics/coding, whereas the remaining lectures will still introduce new techniques, but will also be much more case-focused

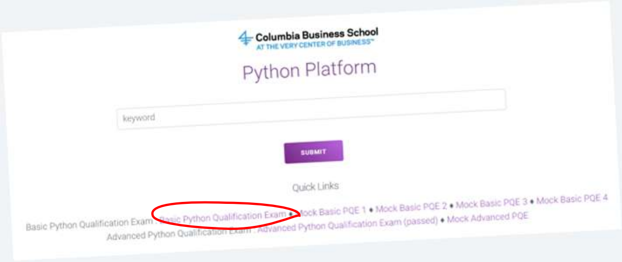
Python

- The first homework for this class won't be due for at least 3-4 weeks
- During this time, I will expect you to go through a basic Python class, to learn the fundamentals of the language
  - Some of you will already know Python, or have gone through this class pre-semester and won't need to do this
- You will do this by going through the following Canvas class

  https://courseworks2.columbia.edu/courses/152704
- By the end of week 3, I will expect you to have passed the Basic Python Qualification exam here:

  http://cbspython.herokuapp.com/

Python



Columbia Business School
AT THE VERY CENTER OF BUSINESS™

Python Platform

keyword

Quick Links

Basic Python Qualification Exam : Basic Python Qualification Exam • Mock Basic PQE 1 • Mock Basic PQE 2 • Mock Basic PQE 3 • Mock Basic PQE 4

Advanced Python Qualification Exam : Advanced Python Qualification Exam (passed) • Mock Advanced PQE

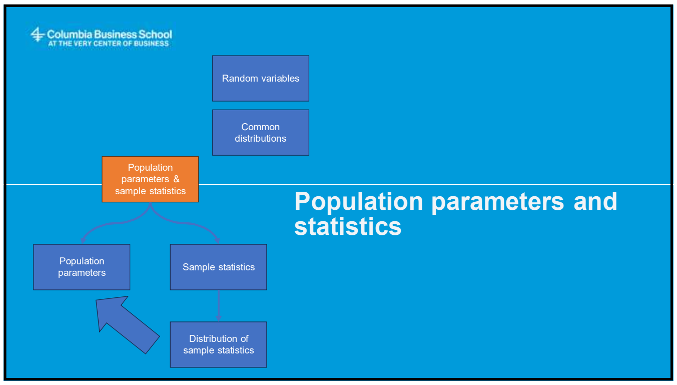
Math pre-requisites

- Basic algebra
- Basic calculus
- Basic matrix algebra
- I have included two handouts on Canvas to help you catch up with these pre-reqs if you're rusty
- They include "class exercises", which will guide you through calculations we'll do in class before we do them

Important note

⚠

This is **not** a programming class and **not** a math class. We will cover programming and math, but only to the extent they are required to demonstrate, and deeply understand the tools we will learn. We will cut corners by writing code that is not as efficient as it could be. We will also eschew mathematical details in proofs. There are plenty of other classes I'll recommend at the end that you can take to go more in-depth; our focus will be **business** analytics



Random variables

Common distributions

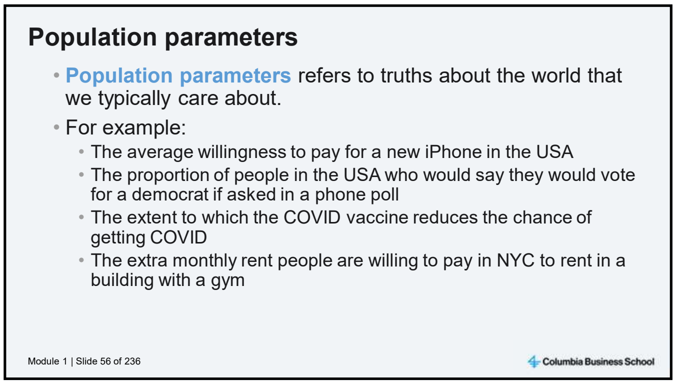
Population parameters & sample statistics

Population parameters

Sample statistics

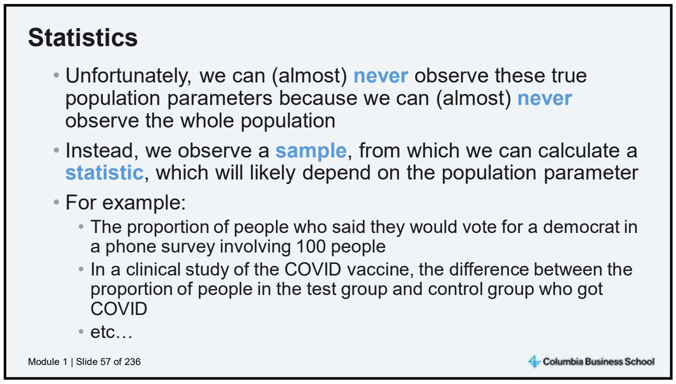
Distribution of sample statistics

Population parameters and statistics



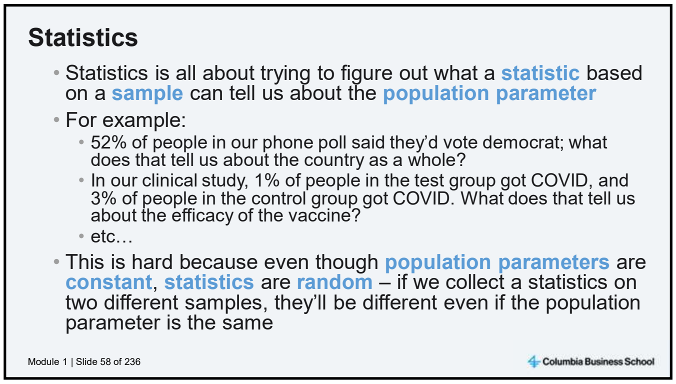
Population parameters

- **Population parameters** refers to truths about the world that we typically care about.
- For example:
  - The average willingness to pay for a new iPhone in the USA
  - The proportion of people in the USA who would say they would vote for a democrat if asked in a phone poll
  - The extent to which the COVID vaccine reduces the chance of getting COVID
  - The extra monthly rent people are willing to pay in NYC to rent in a building with a gym



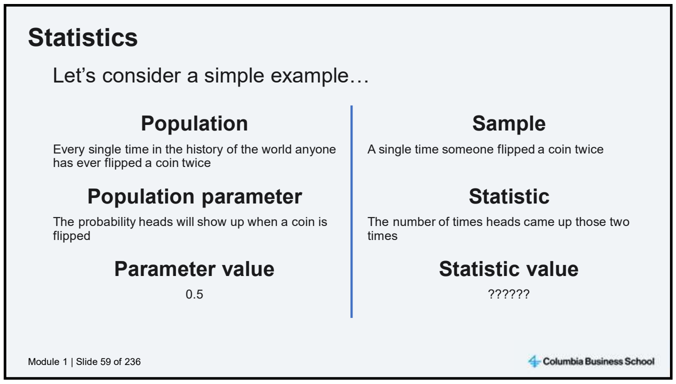
Statistics

- Unfortunately, we can (almost) **never** observe these true population parameters because we can (almost) **never** observe the whole population
- Instead, we observe a **sample**, from which we can calculate a **statistic**, which will likely depend on the population parameter
- For example:
  - The proportion of people who said they would vote for a democrat in a phone survey involving 100 people
  - In a clinical study of the COVID vaccine, the difference between the proportion of people in the test group and control group who got COVID
  - etc…



Statistics

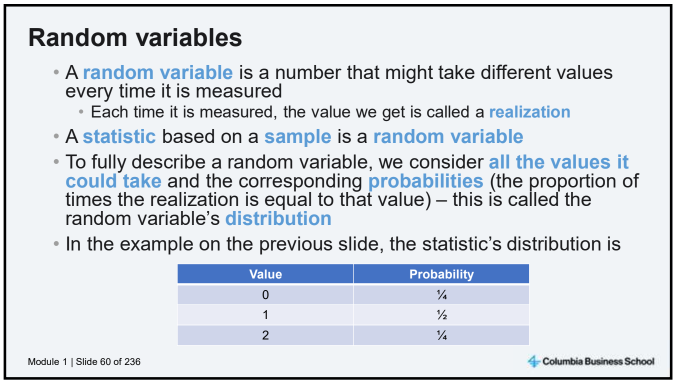
- Statistics is all about trying to figure out what a **statistic** based on a **sample** can tell us about the **population parameter**
- For example:
  - 52% of people in our phone poll said they'd vote democrat; what does that tell us about the country as a whole?
  - In our clinical study, 1% of people in the test group got COVID, and 3% of people in the control group got COVID. What does that tell us about the efficacy of the vaccine?
  - etc…
- This is hard because even though **population parameters** are **constant**, **statistics** are **random** – if we collect a statistics on two different samples, they'll be different even if the population parameter is the same



Statistics

Let's consider a simple example…

Population	Sample
Every single time in the history of the world anyone has ever flipped a coin twice	A single time someone flipped a coin twice
Population parameter	**Statistic**
The probability heads will show up when a coin is flipped	The number of times heads came up those two times
Parameter value	**Statistic value**
0.5	??????



Random variables

- A **random variable** is a number that might take different values every time it is measured
  - Each time it is measured, the value we get is called a **realization**
- A **statistic** based on a **sample** is a **random variable**
- To fully describe a random variable, we consider **all the values it could take** and the corresponding **probabilities** (the proportion of times the realization is equal to that value) – this is called the random variable's **distribution**
- In the example on the previous slide, the statistic's distribution is

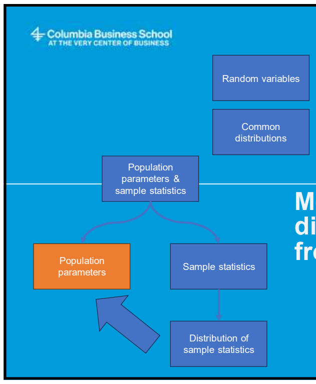
Value	Probability
0	¼
1	½
2	¼

Random variables

- How did we figure out the probabilities on the previous slide?
- We simply looked at every possible outcome the variable could take…
  - HH (2 heads)
  - HT (1 head)
  - TH (1 head)
  - TT (0 heads)
- …and calculated probabilities using the proportion of outcomes that would lead to that value of the statistic given the population parameter

In this exercise, we assumed we knew the population parameter, and we worked out the distribution of the statistic

The first part of this class is all about going in the opposite direction


Random variables
Common distributions
Population parameters & sample statistics
Population parameters
Sample statistics
Distribution of sample statistics

Measures of location, dispersion, and correlation; from concepts to numbers

So far so good

But how do we define the population parameters we care about?

Evaluating salespeople

McKinsey & Company

Growth, Marketing & Sales

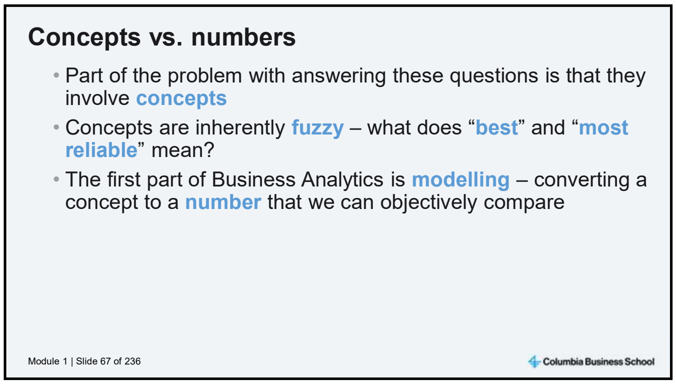
How top performers outpace peers in sales productivity

July 6, 2023 | Article

https://www.mckinsey.com/capabilities/growth-marketing-and-sales/our-insights/how-top-performers-outpace-peers-in-sales-productivity

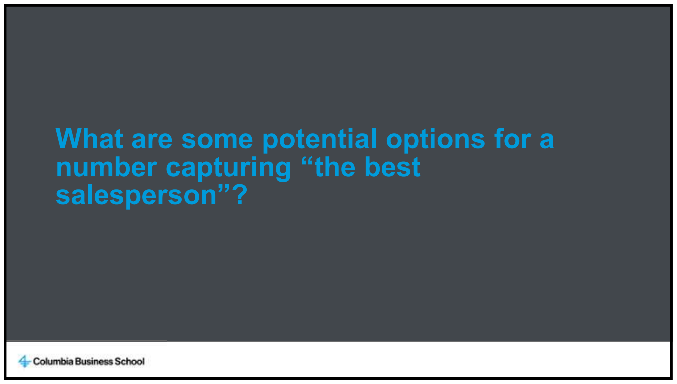
You have data on the number of contracts each of your salespeople have closed since they've been employed with you (at least 12 months, at most 60)…

How can you determine who your best salesperson is? How about your most reliable salesperson?

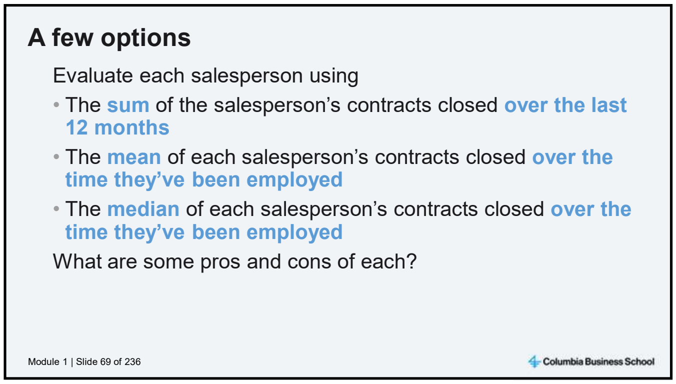


Concepts vs. numbers

- Part of the problem with answering these questions is that they involve **concepts**
- Concepts are inherently **fuzzy** – what does "**best**" and "**most reliable**" mean?
- The first part of Business Analytics is **modelling** – converting a concept to a **number** that we can objectively compare



What are some potential options for a number capturing "the best salesperson"?

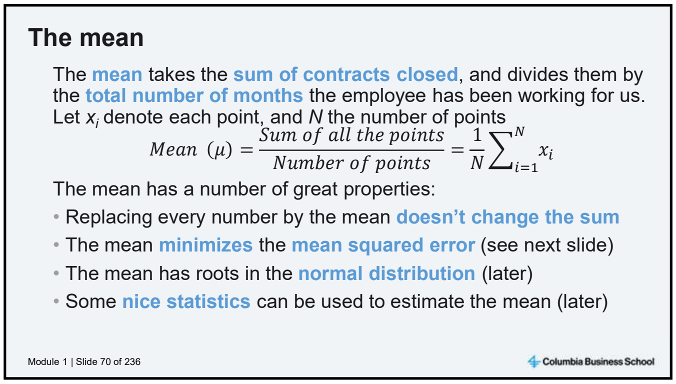


A few options

Evaluate each salesperson using

- The **sum** of the salesperson's contracts closed **over the last 12 months**
- The **mean** of each salesperson's contracts closed **over the time they've been employed**
- The **median** of each salesperson's contracts closed **over the time they've been employed**

What are some pros and cons of each?



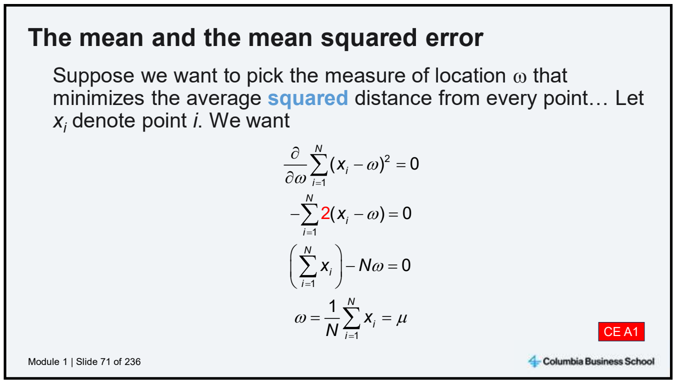
The mean

The **mean** takes the **sum of contracts closed**, and divides them by the **total number of months** the employee has been working for us. Let x_i denote each point, and N the number of points

$$Mean\ (\mu) = \frac{Sum\ of\ all\ the\ points}{Number\ of\ points} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

The mean has a number of great properties:

- Replacing every number by the mean **doesn't change the sum**
- The mean **minimizes** the **mean squared error** (see next slide)
- The mean has roots in the **normal distribution** (later)
- Some **nice statistics** can be used to estimate the mean (later)



The mean and the mean squared error

Suppose we want to pick the measure of location ω that minimizes the average **squared** distance from every point… Let x_i denote point *i*. We want

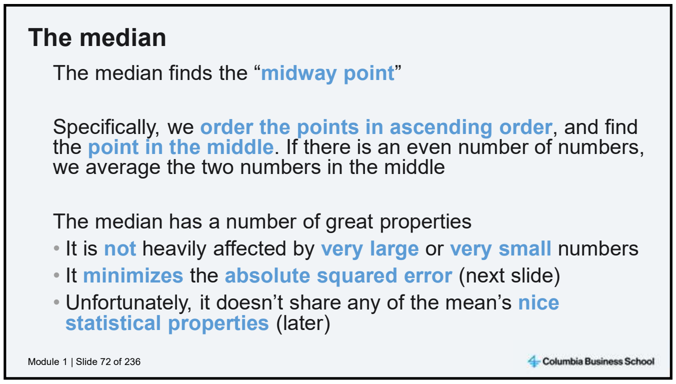
$$\frac{\partial}{\partial \omega}\sum_{i=1}^{N}(x_i-\omega)^2 = 0$$

$$-\sum_{i=1}^{N} 2(x_i-\omega) = 0$$

$$\left(\sum_{i=1}^{N} x_i\right) - N\omega = 0$$

$$\omega = \frac{1}{N}\sum_{i=1}^{N} x_i = \mu$$

CE A1



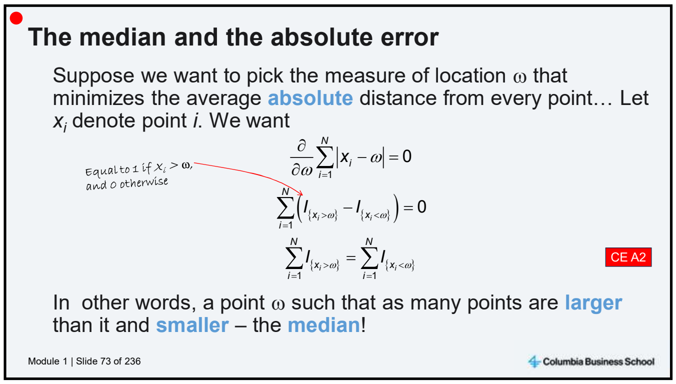
The median

The median finds the "**midway point**"

Specifically, we **order the points in ascending order**, and find the **point in the middle**. If there is an even number of numbers, we average the two numbers in the middle

The median has a number of great properties

- It is **not** heavily affected by **very large** or **very small** numbers
- It **minimizes** the **absolute squared error** (next slide)
- Unfortunately, it doesn't share any of the mean's **nice statistical properties** (later)



The median and the absolute error

Suppose we want to pick the measure of location ω that minimizes the average **absolute** distance from every point… Let x_i denote point *i*. We want

$$\frac{\partial}{\partial\omega}\sum_{i=1}^{N}\left|x_i-\omega\right|=0$$

Equal to 1 if $x_i > \omega$, and 0 otherwise

$$\sum_{i=1}^{N}\left(I_{\{x_i>\omega\}}-I_{\{x_i<\omega\}}\right)=0$$

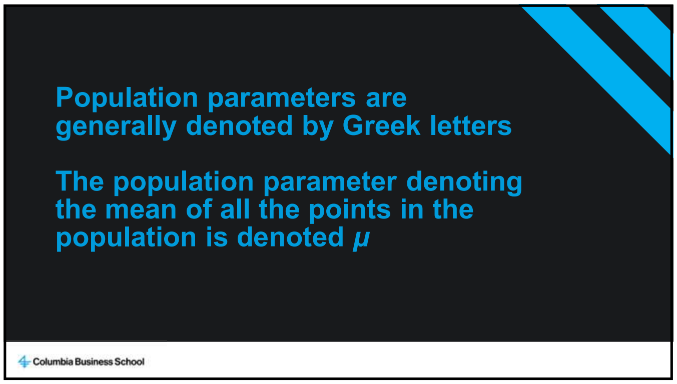
$$\sum_{i=1}^{N}I_{\{x_i>\omega\}}=\sum_{i=1}^{N}I_{\{x_i<\omega\}}$$

CE A2

In other words, a point ω such that as many points are **larger** than it and **smaller** – the **median**!

Module 1 | Slide 73 of 236

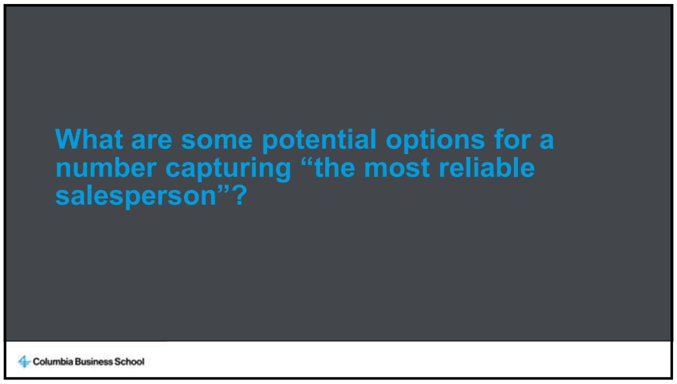
Columbia Business School



Population parameters are generally denoted by Greek letters

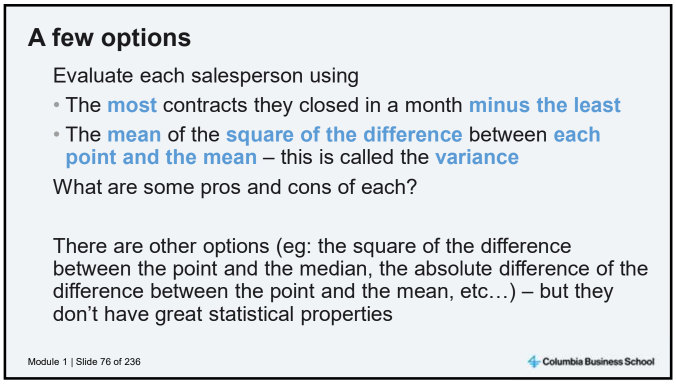
The population parameter denoting the mean of all the points in the population is denoted *μ*

Columbia Business School



What are some potential options for a number capturing "the most reliable salesperson"?

Columbia Business School



A few options

Evaluate each salesperson using

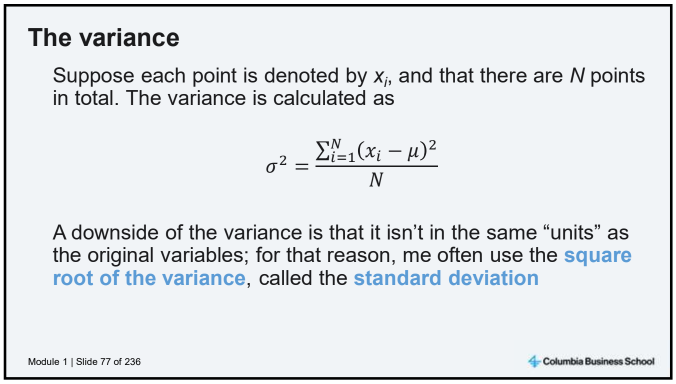
- The **most** contracts they closed in a month **minus the least**
- The **mean** of the **square of the difference** between **each point and the mean** – this is called the **variance**

What are some pros and cons of each?

There are other options (eg: the square of the difference between the point and the median, the absolute difference of the difference between the point and the mean, etc…) – but they don't have great statistical properties

Module 1 | Slide 76 of 236

Columbia Business School



The variance

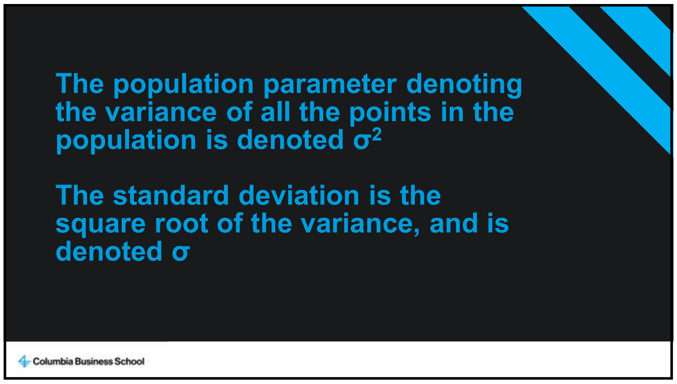
Suppose each point is denoted by x_i, and that there are *N* points in total. The variance is calculated as

$$\sigma^2=\frac{\sum_{i=1}^{N}(x_i-\mu)^2}{N}$$

A downside of the variance is that it isn't in the same "units" as the original variables; for that reason, me often use the **square root of the variance**, called the **standard deviation**

Module 1 | Slide 77 of 236

Columbia Business School



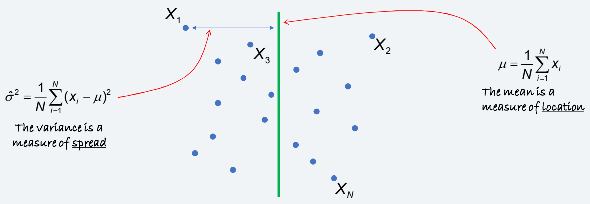
The population parameter denoting the variance of all the points in the population is denoted σ²

The standard deviation is the square root of the variance, and is denoted σ

Columbia Business School

The variance

Suppose we have N points, each denoted x_i.



$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

The variance is a measure of spread

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

The mean is a measure of location

An easier way to calculate the variance

There's an easier way to calculate the variance

$$\begin{aligned}\sigma^2 &= \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2\\ &= \frac{1}{N}\sum_{i=1}^{N}\left(x_i^2 - 2\mu x_i + \mu^2\right)\\ &= \mu^2 + \frac{1}{N}\sum_{i=1}^{N}x_i^2 - \frac{2\mu}{N}\sum_{i=1}^{N}x_i\\ &= \mu^2 + \frac{1}{N}\sum_{i=1}^{N}x_i^2 - 2\mu^2\\ &= \frac{1}{N}\sum_{i=1}^{N}x_i^2 - \mu^2\end{aligned}$$

Back to salesforce analytics

So how could we initially find the best salesperson and the most reliable?

- **Best**: find the mean for each salesperson, find the one with the best mean
- **Most reliable**: find the variance for each salesperson, find the one with the lowest variance

What if you wanted to figure out whether the performance of two salespeople was related?

When Juan does well, does Xie tend to do well also? Or is it the other way round? Or are Xie and Juan's performances completely unrelated?

The covariance

The covariance allows us to figure out whether two variables tend to move "**in the same direction**". Suppose we have N observations of Xie's and Juan's performance. Let Juan's performance in a given month be x_i, and Xie's be y_i.

$$\text{Covariance} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_X)(y_i - \mu_Y)$$

If Juan's performance tends to be higher than average when Xie's is, both terms will be positive and negative at the same time; the covariance will be **positive**. If they're unrelated, the terms will have the same sign sometimes, and different signs other times; they'll cancel out; the covariance will be **close to 0**

An easier way to calculate the covariance

$$\begin{aligned}\text{Covariance} &= \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_X)(y_i - \mu_Y)\\ &= \frac{1}{N}\left(\sum_{i=1}^{N}x_i y_i - \mu_X\sum_{i=1}^{N}y_i - \mu_Y\sum_{i=1}^{N}x_i + \mu_X\mu_Y\sum_{i=1}^{N}1\right)\\ &= \frac{1}{N}\left(\sum_{i=1}^{N}x_i y_i - N\mu_X\mu_Y - N\mu_Y\mu_X + N\mu_X\mu_Y\right)\\ &= \text{Mean of the product of the two variables} - \mu_X\mu_Y\end{aligned}$$

The correlation

The problem with the covariance is that it depends on the **scale** of the variables. If the **variables are large**, the **covariance will be large**

The correlation **standardizes** by the **variance** of each variable to get a **number between –1 and 1**

$$\text{Correlation}(X,Y)\,(\rho) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$$

The population parameter denoting the correlation between two variables is denoted *ρ*

Back to salesforce analytics

McKinsey & Company

How top performers outpace peers in sales productivity

- Salespeople are often **paid** based on the **number of contracts** they close
- There might therefore be **incentives** to "**game**" the system
- Suppose your salespeople close a **mean of 23 contracts/month**, with a **standard deviation of 4 contracts**
- One month, Bob reports closing **47 contracts** – seems a little high. But is it **suspiciously high**? What's the **probability** of his closing so many contracts?

Chebyshev's Inequality

Astonishingly the **mean** and the **variance** alone are enough to make a **strikingly general statement**

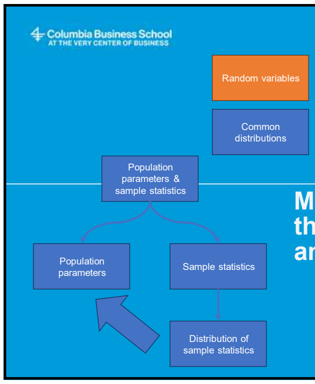
The probability of observing a value of a quantity more than *k* standard deviations away from its mean is less than $1/k^2$

This is called **Chebyshev's Inequality**, and we'll be able to prove it a little bit later

Back to salesforce analytics

- The mean contracts closed per month is **23**, with a standard deviation of **4**
- Bob closed **47** contracts – that is **(47 – 23)/4 = 6 standard deviations** away from the mean
- According to Chebyshev's Inequality, the probability of observing a value this far from the mean is **less than $1/6^2$ = around 3 in 100**
- As we'll see later, if we know more about this quantity we might be able to get this **even tighter** (i.e., the probability will be even less)

Columbia Business School AT THE VERY CENTER OF BUSINESS



Random variables

Common distributions

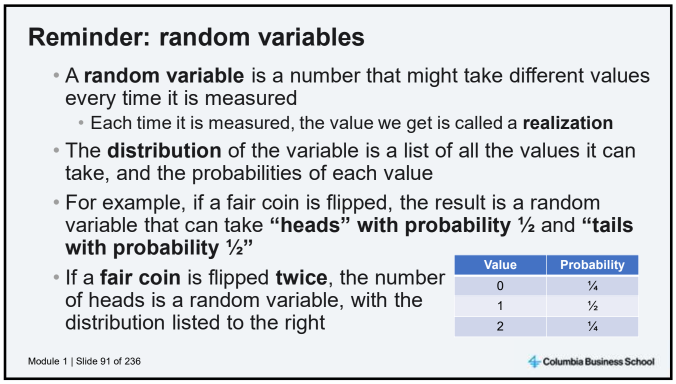
Population parameters & sample statistics

Population parameters

Sample statistics

Distribution of sample statistics

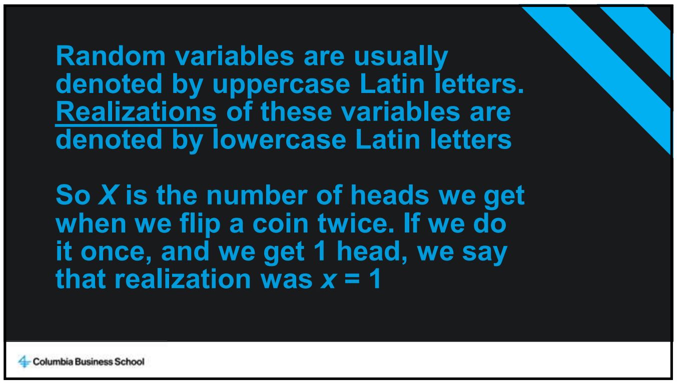
More on random variables; the expectation, the variance, and the covariance



Reminder: random variables

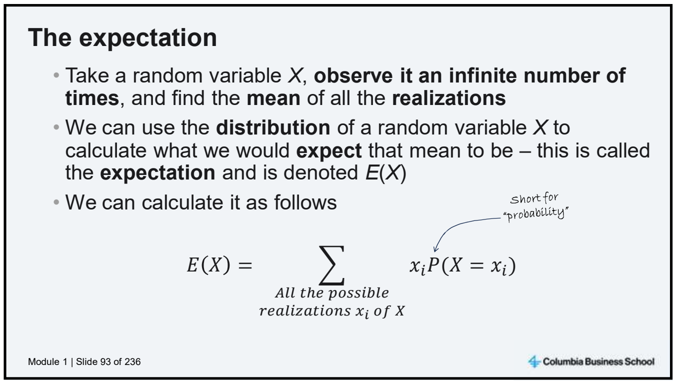
- A **random variable** is a number that might take different values every time it is measured
  - Each time it is measured, the value we get is called a **realization**
- The **distribution** of the variable is a list of all the values it can take, and the probabilities of each value
- For example, if a fair coin is flipped, the result is a random variable that can take **"heads" with probability** ½ and **"tails with probability** ½**"**
- If a **fair coin** is flipped **twice**, the number of heads is a random variable, with the distribution listed to the right

Value	Probability
0	¼
1	½
2	¼



Random variables are usually denoted by uppercase Latin letters. Realizations of these variables are denoted by lowercase Latin letters

So *X* is the number of heads we get when we flip a coin twice. If we do it once, and we get 1 head, we say that realization was *x* = 1

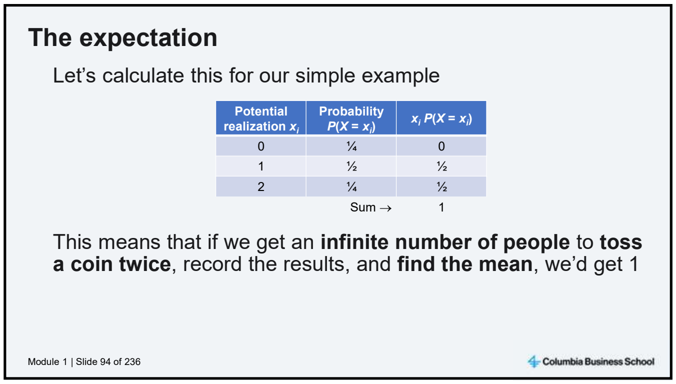


The expectation

- Take a random variable *X*, **observe it an infinite number of times**, and find the **mean** of all the **realizations**
- We can use the **distribution** of a random variable *X* to calculate what we would **expect** that mean to be – this is called the **expectation** and is denoted *E*(*X*)
- We can calculate it as follows

$$E(X) = \underbrace{\sum}_{\substack{\text{All the possible} \\ \text{realizations } x_i \text{ of } X}} x_i P(X = x_i)$$

Short for "probability"

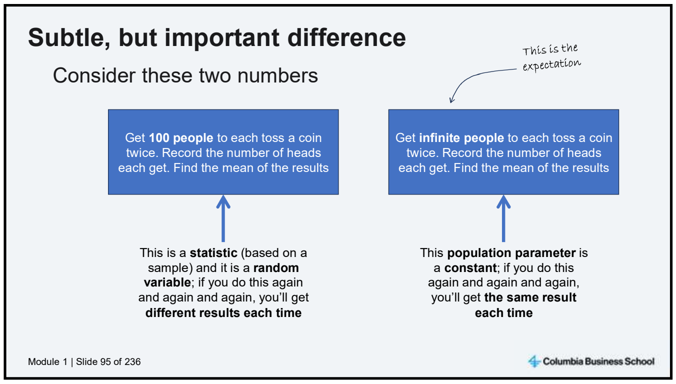


The expectation

Let's calculate this for our simple example

Potential realization x_i	Probability $P(X = x_i)$	$x_i\, P(X = x_i)$
0	¼	0
1	½	½
2	¼	½
	Sum →	1

This means that if we get an **infinite number of people** to **toss a coin twice**, record the results, and **find the mean**, we'd get 1



Subtle, but important difference

Consider these two numbers

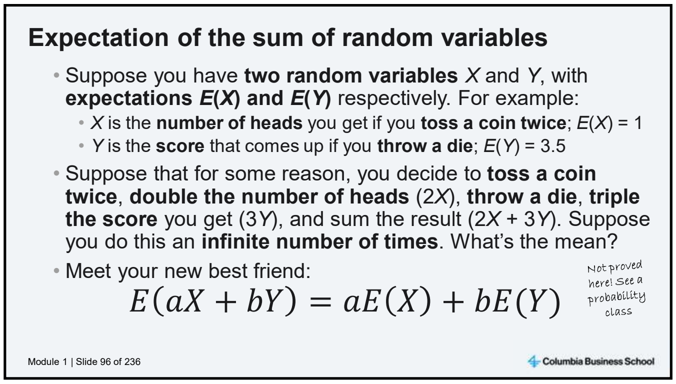
This is the expectation

Get **100 people** to each toss a coin twice. Record the number of heads each get. Find the mean of the results

This is a **statistic** (based on a sample) and it is a **random variable**; if you do this again and again and again, you'll get **different results each time**

Get **infinite people** to each toss a coin twice. Record the number of heads each get. Find the mean of the results

This **population parameter** is a **constant**; if you do this again and again and again, you'll get **the same result each time**



Expectation of the sum of random variables

- Suppose you have **two random variables** *X* and *Y*, with **expectations *E*(*X*) and *E*(*Y*)** respectively. For example:
  - *X* is the **number of heads** you get if you **toss a coin twice**; *E*(*X*) = 1
  - *Y* is the **score** that comes up if you **throw a die**; *E*(*Y*) = 3.5
- Suppose that for some reason, you decide to **toss a coin twice**, **double the number of heads** (2*X*), **throw a die**, **triple the score** you get (3*Y*), and sum the result (2*X* + 3*Y*). Suppose you do this an **infinite number of times**. What's the mean?
- Meet your new best friend:

$$E(aX + bY) = aE(X) + bE(Y)$$

Not proved here! See a probability class

The variance

- Take a random variable *X*, **observe it an infinite number of times**, and find the **variance** of all the **realizations**
  - In other words, for every realization *x*, subtract the population parameter *E*(*X*), square it…
  - …and then find the average of all of them
- It should be straightforward to see that

$$Var(X) = E[(X - E[X])^2]$$

An easier way to calculate the variance

Again, there is an easier way to calculate the variance

$$
\begin{aligned}
Var(X) &= E\left[\left(X - E[X]\right)^2\right] \\
&= E\left(X^2 - 2XE[X] + E[X]^2\right) \\
&= E(X^2) - 2E(X)E\left[E(X)\right] + E\left[E(X)^2\right] \\
&= E(X^2) - 2E(X)E(X) + E(X)^2 \\
&= E(X^2) - E(X)^2
\end{aligned}
$$

Now use E(aX+bY) = aE(X) + bE(Y); every item in red in this equation is a random variable, and every item in green is a constant

This is the expected value of a constant – it's just equal to the constant

The variance

Let's calculate this for our simple example

Potential realization x_i	$(x_i)^2$	Probability $P(X = x_i)$	$x_i\ P(X = x_i)$	$(x_i)^2\ P(X = x_i)$
0	0	¼	0	0
1	1	½	½	½
2	4	¼	½	1
		Sum →	$E(X) = 1$	$E(X^2) = 1.5$

Therefore

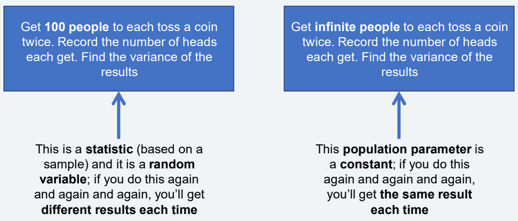
$$Var(X) = E(X^2) - E(X)^2 = 1.5 - 1^2 = 0.5$$

This means that if we get an **infinite number of people** to **toss a coin twice**, record the results, and **find the variance**, we'd get 0.5

Subtle, but important difference

Consider these two numbers

This is the variance of a random variable



Get **100 people** to each toss a coin twice. Record the number of heads each get. Find the variance of the results

Get **infinite people** to each toss a coin twice. Record the number of heads each get. Find the variance of the results

This is a **statistic** (based on a sample) and it is a **random variable**; if you do this again and again and again, you'll get **different results each time**

This **population parameter** is a **constant**; if you do this again and again and again, you'll get **the same result each time**

The covariance

- Take two random variables *X* and *Y*; observe **pairs of realizations** an **infinite number of times**, and find the **covariance** of the results
- It should be straightforward to see that

$$Cov(X, Y) = E[(X - E[X])(Y - E[Y])]$$

- Using a trick similar to the one we've been using…

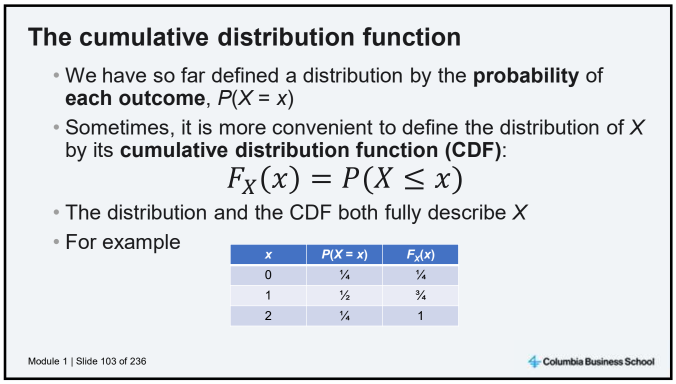
$$Cov(X, Y) = E[XY] - E[X]E[Y]$$

Variance of the sum of random variables

- Suppose you have **two random variables** *X* and *Y*, with **expectations *E*(*X*) and *E*(*Y*)** respectively. For example:
  - *X* is the **number of heads** you get if you **toss a coin twice**; *E*(*X*) = 1
  - *Y* is the **score** that comes up if you **throw a die**; *E*(*Y*) = 3.5
- Suppose that for some reason, you decide to **toss a coin twice**, **double the number of heads** (2*X*), **throw a die**, **triple the score** you get (3*Y*), and sum the result (2*X* + 3*Y*). Suppose you do this an **infinite number of times**. What's the variance?
- Meet your (second) new best friend:

Not proved here!

$$Var(aX + bY) = a^2Var(X) + b^2Var(Y) + 2abCov(X, Y)$$



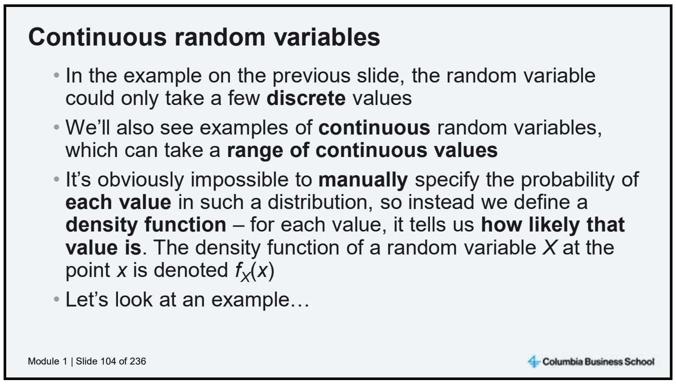
The cumulative distribution function

- We have so far defined a distribution by the **probability** of **each outcome**, $P(X = x)$
- Sometimes, it is more convenient to define the distribution of X by its **cumulative distribution function (CDF)**:

$$F_X(x) = P(X \leq x)$$

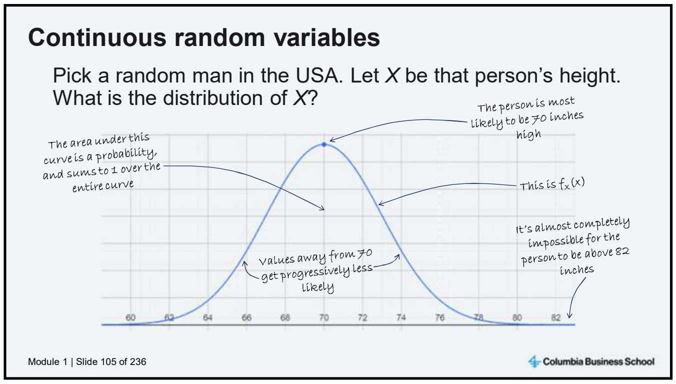
- The distribution and the CDF both fully describe X
- For example

x	$P(X = x)$	$F_X(x)$
0	¼	¼
1	½	¾
2	¼	1



Continuous random variables

- In the example on the previous slide, the random variable could only take a few **discrete** values
- We'll also see examples of **continuous** random variables, which can take a **range of continuous values**
- It's obviously impossible to **manually** specify the probability of **each value** in such a distribution, so instead we define a **density function** – for each value, it tells us **how likely that value is**. The density function of a random variable X at the point x is denoted $f_X(x)$
- Let's look at an example…



Continuous random variables

Pick a random man in the USA. Let X be that person's height. What is the distribution of X?

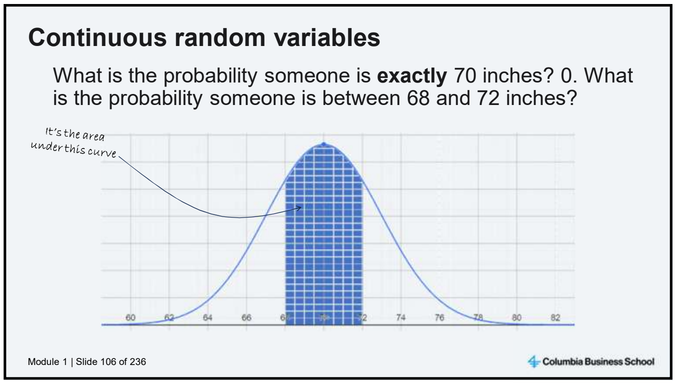
The person is most likely to be 70 inches high

The area under this curve is a probability, and sums to 1 over the entire curve

This is $f_X(x)$

It's almost completely impossible for the person to be above 82 inches

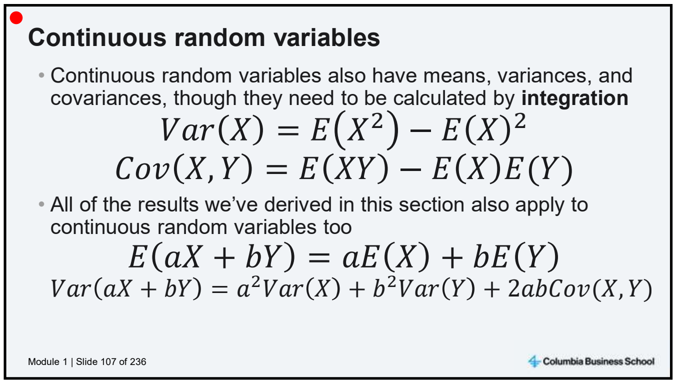
Values away from 70 get progressively less likely



Continuous random variables

What is the probability someone is **exactly** 70 inches? 0. What is the probability someone is between 68 and 72 inches?

It's the area under this curve



Continuous random variables

- Continuous random variables also have means, variances, and covariances, though they need to be calculated by **integration**

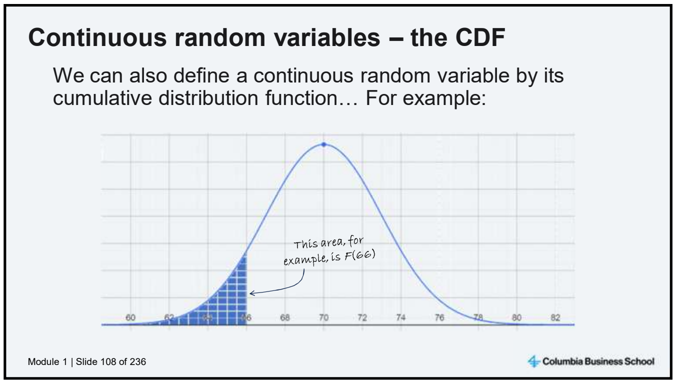
$$Var(X) = E\left(X^2\right) - E(X)^2$$

$$Cov(X,Y) = E(XY) - E(X)E(Y)$$

- All of the results we've derived in this section also apply to continuous random variables too

$$E(aX + bY) = aE(X) + bE(Y)$$

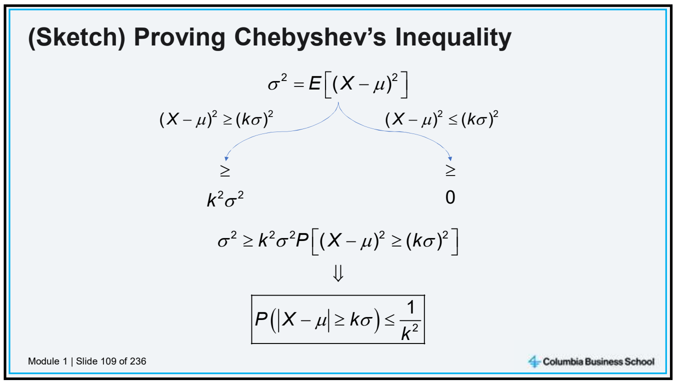
$$Var(aX + bY) = a^2Var(X) + b^2Var(Y) + 2abCov(X,Y)$$



Continuous random variables – the CDF

We can also define a continuous random variable by its cumulative distribution function… For example:

This area, for example, is $F(66)$



(Sketch) Proving Chebyshev's Inequality

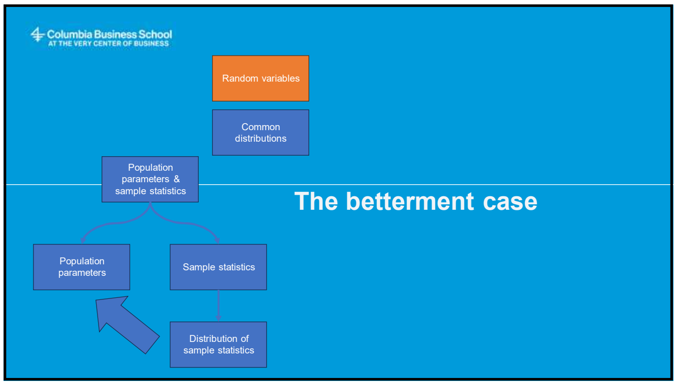
$$\sigma^2 = E\left[(X-\mu)^2\right]$$

$$\underbrace{(X-\mu)^2 \geq (k\sigma)^2}_{\geq\ k^2\sigma^2} \qquad \underbrace{(X-\mu)^2 \leq (k\sigma)^2}_{\geq\ 0}$$

$$\sigma^2 \geq k^2\sigma^2 P\left[(X-\mu)^2 \geq (k\sigma)^2\right]$$

$$\Downarrow$$

$$\boxed{P\left(|X-\mu| \geq k\sigma\right) \leq \frac{1}{k^2}}$$



Columbia Business School
AT THE VERY CENTER OF BUSINESS

Random variables

Common distributions

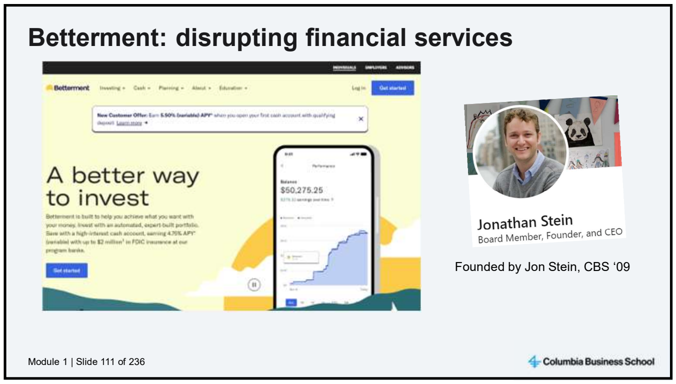
Population parameters & sample statistics

Population parameters

Sample statistics

Distribution of sample statistics

The betterment case



Betterment: disrupting financial services

INDIVIDUALS EMPLOYERS ADVISORS

Betterment Investing Cash Planning About Education Log in Get started

New Customer Offer: Earn 5.50% (variable) APY* when you open your first cash account with qualifying deposit. Learn more

A better way to invest

Betterment is built to help you achieve what you want with your money. Invest with an automated, expert-built portfolio. Save with a high-interest cash account, earning 4.75% APY* (variable) with up to $2 million¹ in FDIC insurance at our program banks.

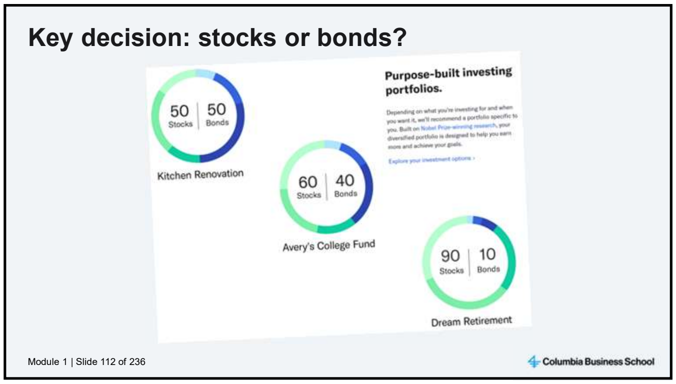
Get started

Performance

Balance
$50,275.25

Jonathan Stein
Board Member, Founder, and CEO

Founded by Jon Stein, CBS '09



Key decision: stocks or bonds?

Purpose-built investing portfolios.

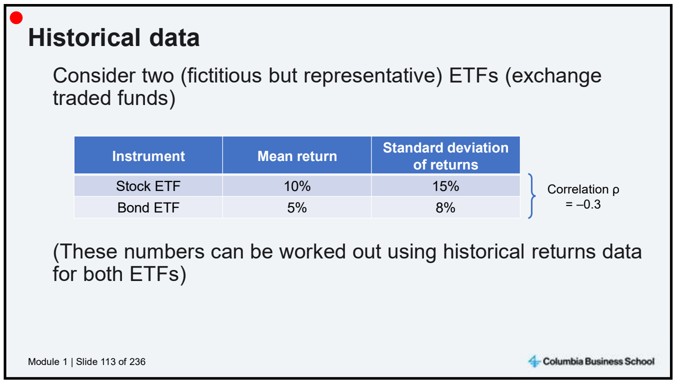
Depending on what you're investing for and when you want it, we'll recommend a portfolio specific to you. Built on Nobel Prize-winning research, your diversified portfolio is designed to help you earn more and achieve your goals.

Explore your investment options ›

50 Stocks | 50 Bonds
Kitchen Renovation

60 Stocks | 40 Bonds
Avery's College Fund

90 Stocks | 10 Bonds
Dream Retirement



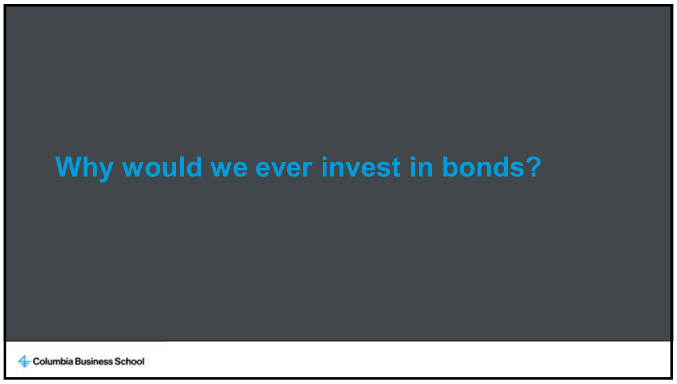
Historical data

Consider two (fictitious but representative) ETFs (exchange traded funds)

Instrument	Mean return	Standard deviation of returns
Stock ETF	10%	15%
Bond ETF	5%	8%

Correlation ρ = –0.3

(These numbers can be worked out using historical returns data for both ETFs)



Why would we ever invest in bonds?

Suppose you have a portfolio that is *a*% stocks, and *b*% bonds… What would the expected return of the portfolio be? What about the standard deviation?

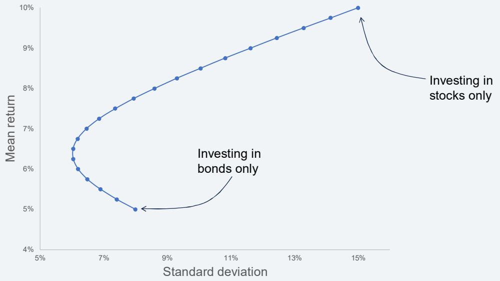
Expected and standard deviation of returns

Suppose we invest in a portfolio that is *a*% stocks, and *b*% bonds…

$$
\begin{aligned}
E(\text{Portfolio}) &= aE(\text{Stock}) + bE(\text{Bonds}) \\
&= 0.1a + 0.05b
\end{aligned}
$$

$$
\begin{aligned}
Std(\text{Portfolio}) &= \sqrt{a^2 Std(\text{Stock})^2 + b^2 Std(\text{Bond})^2 + 2ab Cov(\text{Stock,Bond})} \\
&= \sqrt{a^2 \times 0.15^2 + b^2 \times 0.08^2 - 2ab \times 0.3 \times 0.15 \times 0.08} \\
&= \sqrt{0.0225a^2 + 0.0064b^2 - 0.0072ab}
\end{aligned}
$$

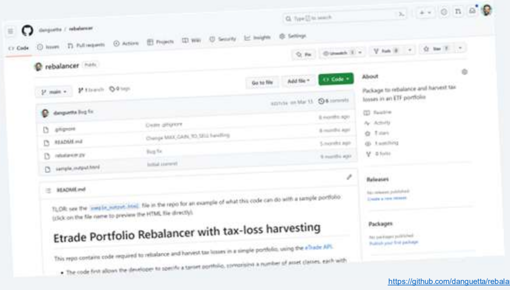
Resulting portfolios



What would happen if the correlation were <u>positive</u>?

We will look at more complex efficient frontiers in our last module…

Side note – what I do…



rebalancer Public

main 1 branch 0 tags

danguetta Bug fix

.gitignore Create .gitignore

README.md Change MAX_GAIN_TO_SELL handling

rebalancer.py Bug fix

sample_output.html Initial commit

About

Package to rebalance and harvest tax losses in an ETF portfolio

README.md

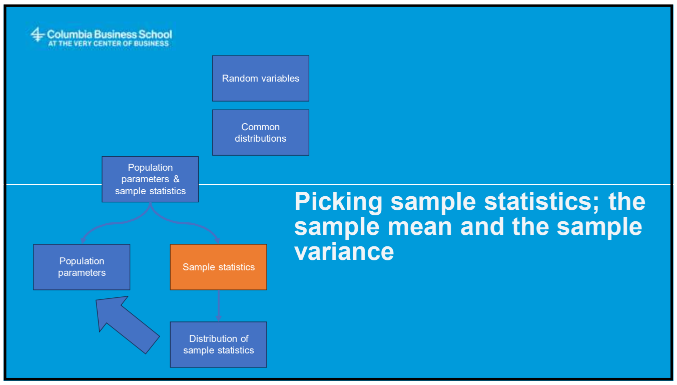
TLDR: see the sample_output.html file in the repo for an example of what this code can do with a sample portfolio (click on the file name to preview the HTML file directly).

Etrade Portfolio Rebalancer with tax-loss harvesting

This repo contains code required to rebalance and harvest tax losses in a simple portfolio, using the eTrade API.

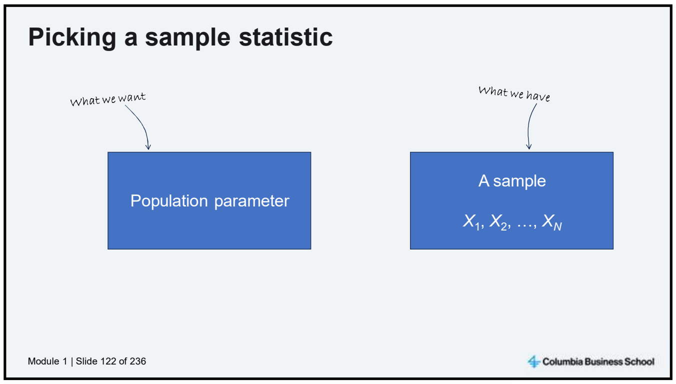
- The code first allows the developer to specify a target portfolio, comprising a number of asset classes, each with

https://github.com/danguetta/rebalancer

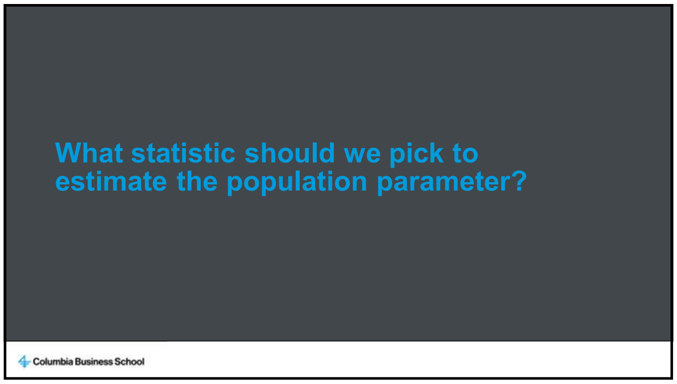


Columbia Business School
AT THE VERY CENTER OF BUSINESS
Random variables
Common distributions
Population parameters & sample statistics
Population parameters
Sample statistics
Distribution of sample statistics

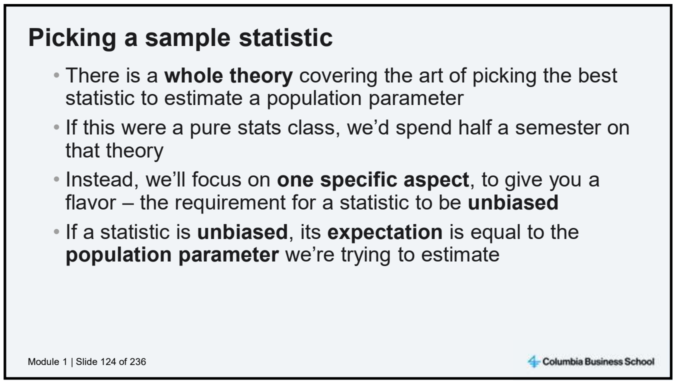
Picking sample statistics; the sample mean and the sample variance



Picking a sample statistic

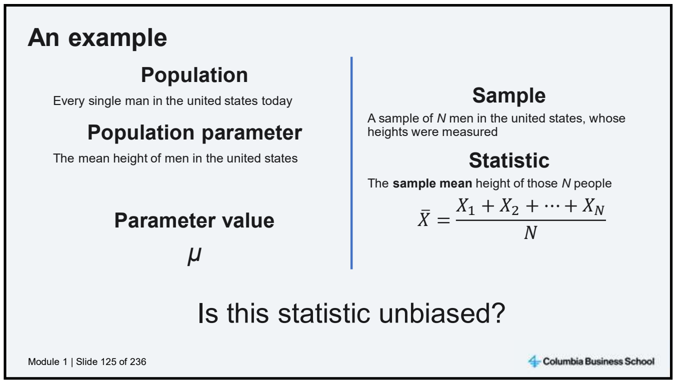


What statistic should we pick to estimate the population parameter?



Picking a sample statistic

- There is a **whole theory** covering the art of picking the best statistic to estimate a population parameter
- If this were a pure stats class, we'd spend half a semester on that theory
- Instead, we'll focus on **one specific aspect**, to give you a flavor – the requirement for a statistic to be **unbiased**
- If a statistic is **unbiased**, its **expectation** is equal to the **population parameter** we're trying to estimate



An example

Population

Every single man in the united states today

Population parameter

The mean height of men in the united states

Parameter value

μ

Sample

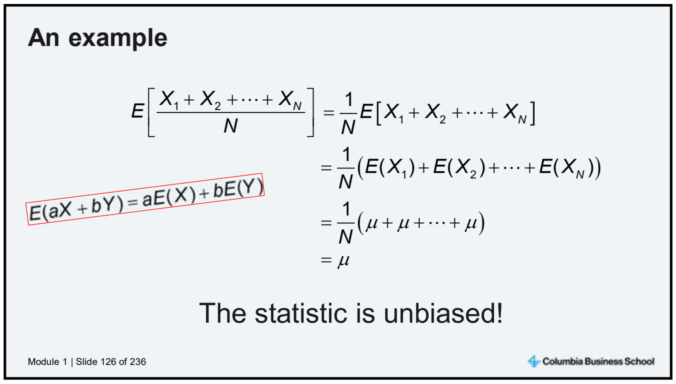
A sample of *N* men in the united states, whose heights were measured

Statistic

The **sample mean** height of those *N* people

$$\bar{X} = \frac{X_1 + X_2 + \cdots + X_N}{N}$$

Is this statistic unbiased?

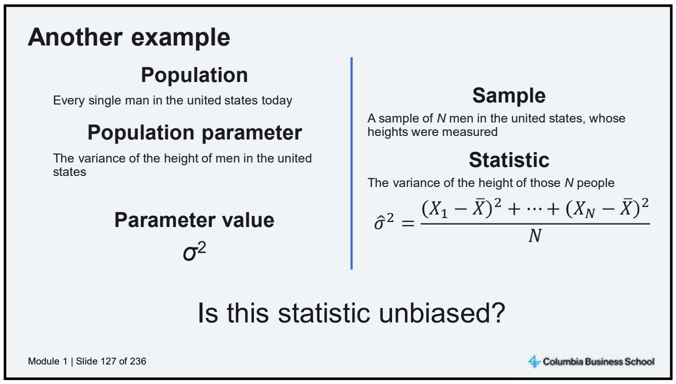


An example

$$\begin{aligned} E\left[\frac{X_1 + X_2 + \cdots + X_N}{N}\right] &= \frac{1}{N} E\left[X_1 + X_2 + \cdots + X_N\right] \\ &= \frac{1}{N}\left(E(X_1) + E(X_2) + \cdots + E(X_N)\right) \\ &= \frac{1}{N}\left(\mu + \mu + \cdots + \mu\right) \\ &= \mu \end{aligned}$$

$\boxed{E(aX + bY) = aE(X) + bE(Y)}$

The statistic is unbiased!



Another example

Population

Every single man in the united states today

Population parameter

The variance of the height of men in the united states

Parameter value

σ^2

Sample

A sample of *N* men in the united states, whose heights were measured

Statistic

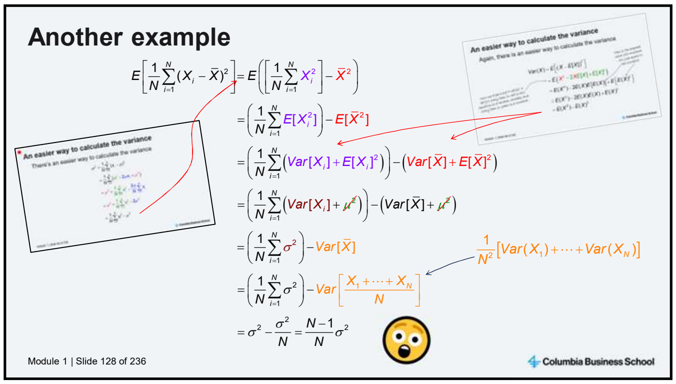
The variance of the height of those *N* people

$$\hat{\sigma}^2 = \frac{(X_1 - \bar{X})^2 + \cdots + (X_N - \bar{X})^2}{N}$$

Is this statistic unbiased?

Module 1 | Slide 127 of 236

Columbia Business School



Another example

$$E\left[\frac{1}{N}\sum_{i=1}^{N}(X_i - \bar{X})^2\right] = E\left(\left[\frac{1}{N}\sum_{i=1}^{N}X_i^2\right] - \bar{X}^2\right)$$

$$= \left(\frac{1}{N}\sum_{i=1}^{N}E[X_i^2]\right) - E[\bar{X}^2]$$

$$= \left(\frac{1}{N}\sum_{i=1}^{N}\left(Var[X_i] + E[X_i]^2\right)\right) - \left(Var[\bar{X}] + E[\bar{X}]^2\right)$$

$$= \left(\frac{1}{N}\sum_{i=1}^{N}\left(Var[X_i] + \cancel{\mu^2}\right)\right) - \left(Var[\bar{X}] + \cancel{\mu^2}\right)$$

$$= \left(\frac{1}{N}\sum_{i=1}^{N}\sigma^2\right) - Var[\bar{X}]$$

$$\frac{1}{N^2}[Var(X_1) + \cdots + Var(X_N)]$$

$$= \left(\frac{1}{N}\sum_{i=1}^{N}\sigma^2\right) - Var\left[\frac{X_1 + \cdots + X_N}{N}\right]$$

$$= \sigma^2 - \frac{\sigma^2}{N} = \frac{N-1}{N}\sigma^2$$

An easier way to calculate the variance

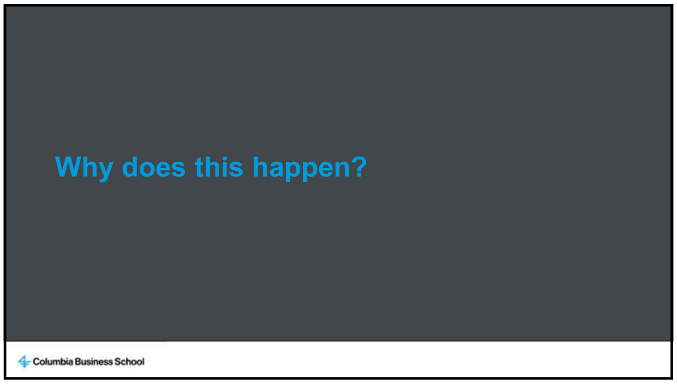
There's an easier way to calculate the variance

An easier way to calculate the variance

Again, there is an easier way to calculate the variance

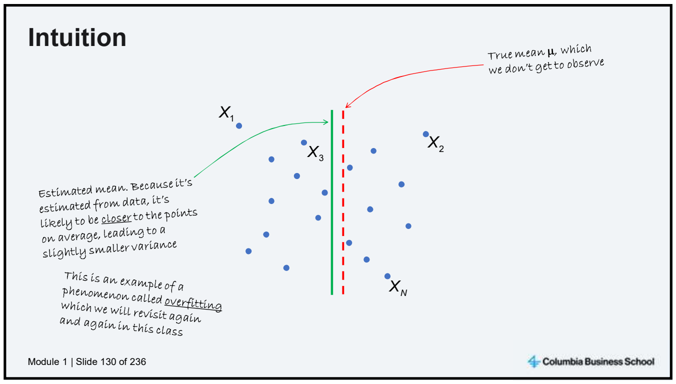
Module 1 | Slide 128 of 236

Columbia Business School



Why does this happen?

Columbia Business School



Intuition

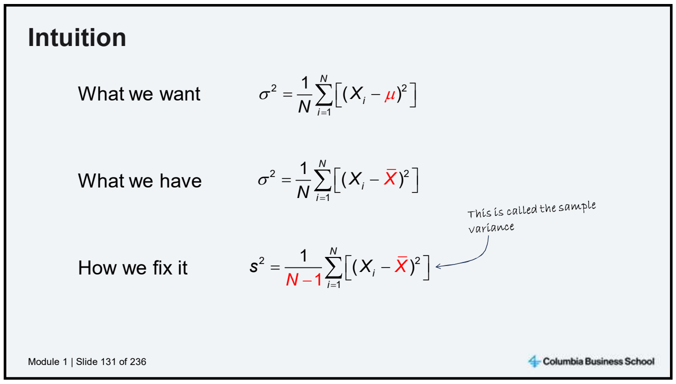
True mean **μ**, which we don't get to observe

Estimated mean. Because it's estimated from data, it's likely to be closer to the points on average, leading to a slightly smaller variance

This is an example of a phenomenon called overfitting which we will revisit again and again in this class

Module 1 | Slide 130 of 236

Columbia Business School



Intuition

What we want $\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}\left[(X_i - \mu)^2\right]$

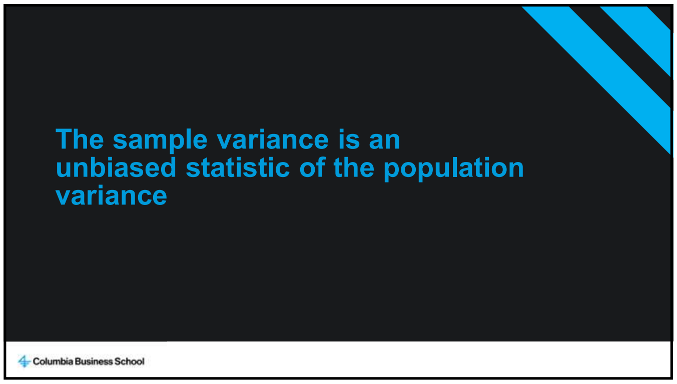
What we have $\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}\left[(X_i - \bar{X})^2\right]$

How we fix it $s^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left[(X_i - \bar{X})^2\right]$

This is called the sample variance

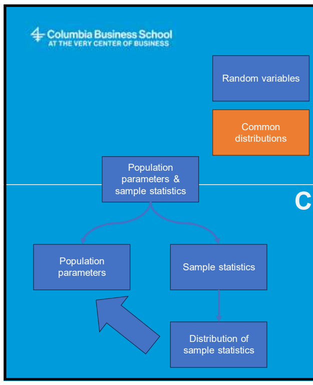
Module 1 | Slide 131 of 236

Columbia Business School



The sample variance is an unbiased statistic of the population variance

Columbia Business School


Random variables
Common distributions
Population parameters & sample statistics
Population parameters
Sample statistics
Distribution of sample statistics

Common distributions

Common distributions

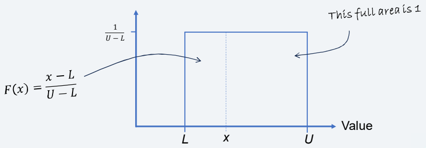
- There are **many distributions** that **commonly arise** in business applications
- You can learn about them all, and more, in a probability class
- In this section, we'll cover **three distributions** which will be essential in this class
  - The **uniform distribution** (discrete and continuous)
  - The **binomial distribution**
  - The **normal distribution**

The discrete uniform distribution

- The **discrete uniform distribution** can take integer values between a **lower point *L*** and an **upper point *U*** with **equal probability**
- For example, the score you will get if you roll a fair die will be **uniformly distributed** between **1 and 6**
- There are *U* – *L* + 1 possible points, so the probability of each point is 1 / (*U* – *L* + 1)

The continuous uniform distribution

- The **continuous uniform distribution** can take any value between a **lower point *L*** and an **upper point *U***, with **equal probability**
- For example if you **randomly throw a dart at a rectangle**, the **distance** it will land from the **end of the rectangle** will have a **uniform distribution**



The bionomial distribution

- Consider a game in which the probability of winning is *p*, and the probability of losing is 1 – *p*
- Suppose you play this game *n* times, and that each of those plays are independent
- Let *X* be the number of wins you achieve in total
- Then *X* follows a binomial distribution, with parameters *n* and *p*

The binomial distribution

- The green card lottery provides some non-citizens the opportunity to get a "green card" every year
- For French citizens, the probability of winning is 1.17%
- Suppose 20 French citizens apply in one year. Let *X* be the number of people who win it

Call us at 1-844-USAGOV1
Search all government
All topics and services
About the U.S. and its government
Government benefits
Housing help
Scams and fraud
Taxes
Travel
Home > Immigration and U.S. citizenship > Immigrant visas and Green Cards > Apply for an immigrant visa >
Diversity Immigrant Visa (Green Card Lottery)
Apply for an immigrant visa
Immigrant work visa
Diversity Immigrant Visa (Green Card Lottery)
Eligibility for the Diversity Visa (DV) Lottery
Diversity Visa Lottery results
Family-based immigrant visas
Check the status of your visa application
Visa application rejected
K-1 fiancé(e) visa

Diversity Immigrant Visa Program (Green Card Lottery)

The Diversity Immigrant Visa (DV) Program awards up to 50,000 immigrant visas each year. Learn if you can apply, see the results, and know what to do if you are selected.

Find out if you are eligible for the Diversity Visa (DV) Lottery and how to register

The Diversity Immigrant Visa Program (DV Program) makes immigrant visas available to citizens of countries with low immigration rates. Find out if you qualify and how to apply.

The binomial distribution

In this scenario

$$X \sim Binomial(n = 20, p = 0.0117)$$

Number of "tries"

Probability of "success"

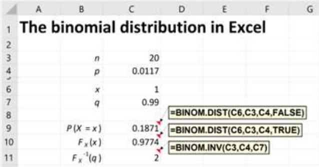
The binomial distribution

Consider three questions

- What is the probability exactly one person wins the lottery
  $P(X = 1)$
- What is the probability no more than one person wins the lottery
  $P(X \leq 1) = F_X(1)$
- I want to buy enough "congratulations" presents for winners, and I want to guarantee there's at least a 99% chance I have enough presents. How many should I buy?
  $F_X^{-1}(0.99)$

Important properties of the binomial

- **Mean**: *np*
- **Variance**: *np*(1 – *p*)

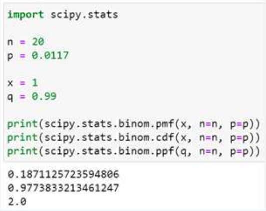


	A	B	C	D	E	F	G
1	The binomial distribution in Excel						
2							
3		n	20				
4		p	0.0117				
6		x	1				
7		q	0.99				
8							
9		$P(X = x)$	0.1871				
10		$F_X(x)$	0.9774				
11		$F_X^{-1}(q)$	2				

=BINOM.DIST(C6,C3,C4,FALSE)

=BINOM.DIST(C6,C3,C4,TRUE)

=BINOM.INV(C3,C4,C7)



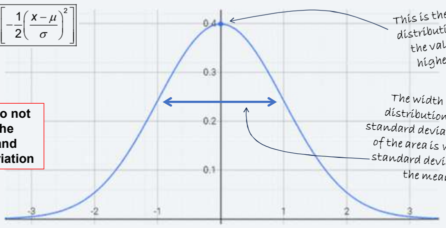
```
import scipy.stats

n = 20
p = 0.0117

x = 1
q = 0.99

print(scipy.stats.binom.pmf(x, n=n, p=p))
print(scipy.stats.binom.cdf(x, n=n, p=p))
print(scipy.stats.binom.ppf(q, n=n, p=p))
```

```
0.1871125723594806
0.9773833213461247
2.0
```

The normal distribution

The normal distribution is the most important continuous distribution out there. Its probability density function is

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma} \right)^2 \right]$$



This is the mean of the distribution – it's also the value with the highest density

The width of the distribution is the standard deviation; 68% of the area is within 1 standard deviation of the mean

Important: do not confuse the variance and standard deviation

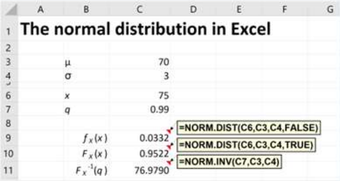
The normal distribution

You own a tanning bed company, and you design your tanning beds to accommodate people up to 75 inches tall. Let *X* be the height of a man in the united states

- What is the probability a man will fit in your tanning bed
  $P(X \leq 75) = F_X(75)$
- You want to make sure your tanning bed fits at least 99% of men. How tall should you make it?
  $F_X^{-1}(0.99)$

Important properties of the normal distribution

- **Mean**: μ
- **Variance**: σ^2

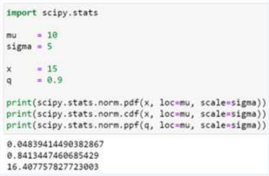


	A	B	C	D	E	F	G
1	The normal distribution in Excel						
2							
3		μ	70				
4		σ	3				
6		x	75				
7		q	0.99				
8							
9		$f_X(x)$	0.0332				
10		$F_X(x)$	0.9522				
11		$F_X^{-1}(q)$	76.9790				

=NORM.DIST(C6,C3,C4,FALSE)

=NORM.DIST(C6,C3,C4,TRUE)

=NORM.INV(C7,C3,C4)



```
import scipy.stats

mu    = 10
sigma = 5

x     = 15
q     = 0.9

print(scipy.stats.norm.pdf(x, loc=mu, scale=sigma))
print(scipy.stats.norm.cdf(x, loc=mu, scale=sigma))
print(scipy.stats.norm.ppf(q, loc=mu, scale=sigma))
```

```
0.04839414490382867
0.8413447460685429
16.407757827723003
```

Important properties of the normal distribution

If

$$X \sim Norm(\mu, \sigma^2)$$

then

$$aX + b \sim Norm(a\mu + b, a^2\sigma^2)$$

(Note: this is a stronger statement than just saying $E(aX) = aE(X)$ and $Var(aX) = a^2 Var(X)$, which is true for *all* random variables)

Important properties of the normal distribution

If

$$X \sim Norm(\mu_X, \sigma_X^2) \text{ and } Y \sim Norm(\mu_Y, \sigma_Y^2)$$

then

$$X + Y \sim Norm\left(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2 + 2Cov(X, Y)\right)$$

(Note: this is a stronger statement than just saying $E(X+Y) = E(X) + E(Y)$ and $Var(X + Y) = Var(X) + Var(Y) + 2Cov(X, Y)$ which is true for *all* random variables)

The *Z*-score

- Suppose *X* is a normally distributed random variable with mean *μ* and variance *σ*^{2}
- Because of the properties of the normal distribution that we discussed

$$\frac{X - \mu}{\sigma} \sim N(\mu = 0, \sigma = 1)$$

- This is called a "Z-score", and it is a convenient way to compare values from different normal distributions with different parameters

The normal approximation to the binomial

- When *n* is large, it can be shown that the binomial distribution is very closely approximated by the normal distribution
- So if

X ~ Binomial(*n* = *n*, *p* = *p*)

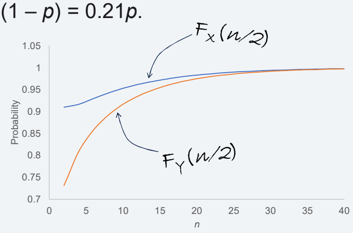
and *n* is very large, we can approximately say that

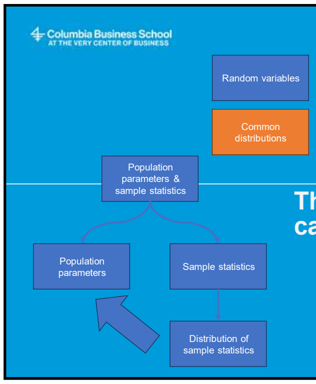
X ~ Normal(μ = *np*, σ^2 = *np*(1 – *p*))

- This is useful because the sum of two normal random variables is normal, but the sum of two binomials is not binomial

The normal approximation to the binomial

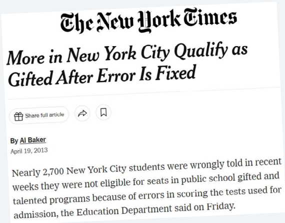
Suppose *X* is a binomial random variable with *n* = *n* and *p* = 0.3. Let *Y* be a normal random variable with mean *np* = 0.3*p*, and variance *np*(1 – *p*) = 0.21*p*.




Random variables
Common distributions
Population parameters & sample statistics
Population parameters
Sample statistics
Distribution of sample statistics

The NYC gifted and talented case

The NYC Gifted & Talented Exam Case



The New York Times

More in New York City Qualify as Gifted After Error Is Fixed

Share full article

By Al Baker

April 19, 2013

Nearly 2,700 New York City students were wrongly told in recent weeks they were not eligible for seats in public school gifted and talented programs because of errors in scoring the tests used for admission, the Education Department said on Friday.

https://www.nytimes.com/2013/04/20/education/score-corrections-qualify-nearly-2700-more-pupils-for-gifted-programs.html

The errors were discovered when two parents, one a statistician, complained that their children had been incorrectly scored, the department said.

According to Pearson, three mistakes were made. Students' ages, which are used to calculate their percentile ranking against students of similar age, were recorded in years and months, but should also have counted days to be precise. Incorrect scoring tables were used. And the formula used to combine the two test parts into one percentile ranking contained an error.

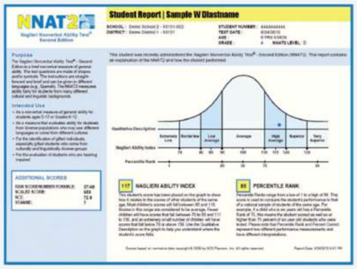
The NYC Gifted & Talented Exam Case

Students are eligible for district G&T programs if they score in the 90th percentile… Anne scored as follows

Test	Score	Mean	Standard deviation	Percentile
Verbal	119/150	100	16	88.30
Non-verbal	123/160	100	16	92.51

What do those percentiles mean? Where do they come from?

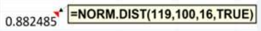
The test results are normally distributed



NNAT2
Student Report | Sample W Dlastname
Purpose
Intended Use
ADDITIONAL SCORES
Qualitative Description
Extremely Low
Borderline
Low Average
Average
High Average
Superior
Very Superior
Naglieri Ability Index
Percentile Rank
117
NAGLIERI ABILITY INDEX
PERCENTILE RANK

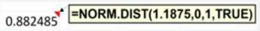
Verbal score

Anne's verbal score is 119. On average, students scored 100 with a standard deviation of 16. What proportion of students did worse than that?



0.882485 =NORM.DIST(119,100,16,TRUE)

Hence scoring in the "88th percentile". Note that a more traditional way to get this number is to first find the so-called "z-score" (119 – 100)/16 = 1.1875, and then to calculate



0.882485 =NORM.DIST(1.1875,0,1,TRUE)

Why does this work? Because the *z*-score is *N*(0, 1)

How can we calculate Anne's "combined" *z*-score?

Do you agree with Barnett's argument that "higher correlation is less favorable to Anne Elizabeth"?

The "combined" score

The combined score is

0.35 × Verbal + 0.65 × Nonverbal

So Anne's combined score is 121.6.

The mean of combined scores is

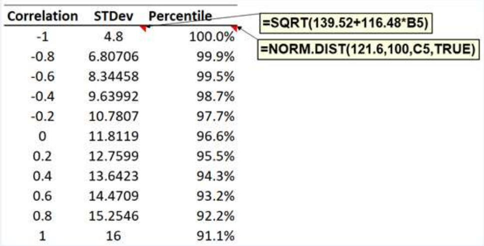
0.35 × *E*(Verbal) + 0.65 × *E*(Nonverbal) = 100

The overall standard deviation is

$$\sqrt{0.35^2 \times 16^2 + 0.65^2 \times 16^2 + 2 \times 0.35 \times 0.65 \times 16 \times 16 \times \rho}$$
$$= \sqrt{139.52 + 116.48\rho}$$

The "combined" score will also be normally distributed… Why?

The "combined" score

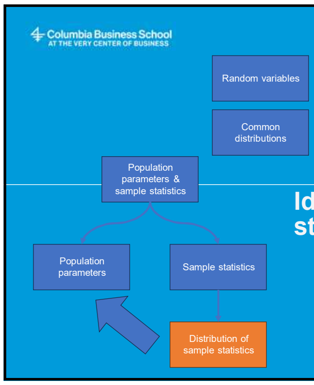


Correlation	STDev	Percentile
-1	4.8	100.0%
-0.8	6.80706	99.9%
-0.6	8.34458	99.5%
-0.4	9.63992	98.7%
-0.2	10.7807	97.7%
0	11.8119	96.6%
0.2	12.7599	95.5%
0.4	13.6423	94.3%
0.6	14.4709	93.2%
0.8	15.2546	92.2%
1	16	91.1%

=SQRT(139.52+116.48*B5)

=NORM.DIST(121.6,100,C5,TRUE)

Columbia Business School
AT THE VERY CENTER OF BUSINESS

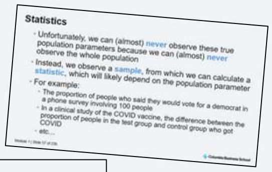

Random variables
Common distributions
Population parameters & sample statistics
Population parameters
Sample statistics
Distribution of sample statistics

Identifying the distribution of statistics

Reminder

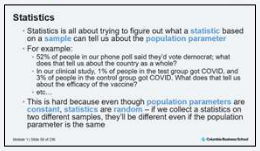
Population parameters

- Population parameters refers to truths about the world that we typically care about.
- For example:
  - The average willingness to pay for a new iPhone in the USA
  - The proportion of people in the USA who would say they would vote for a democrat if asked in a phone poll
  - The extent to which the COVID vaccine reduces the chance of getting COVID
  - The extra monthly rent people are willing to pay in NYC to rent in a building with a gym



Statistics

- Unfortunately, we can (almost) never observe these true population parameters because we can (almost) never observe the whole population
- Instead, we observe a sample, from which we can calculate a statistic, which will likely depend on the population parameter
- For example:
  - The proportion of people who said they would vote for a democrat in a phone survey involving 100 people
  - In a clinical study of the COVID vaccine, the difference between the proportion of people in the test group and control group who got COVID
  - etc...



Statistics

- Statistics is all about trying to figure out what a statistic based on a sample can tell us about the population parameter
- For example:
  - 52% of people in our phone poll said they'd vote democrat; what does that tell us about the country as a whole?
  - In our clinical study, 1% of people in the test group got COVID, and 3% of people in the control group got COVID. What does that tell us about the efficacy of the vaccine?
  - etc...
- This is hard because even though population parameters are constant, statistics are random – if we collect a statistics on two different samples, they'll be different even if the population parameter is the same

The first step to going through the process of going from statistic → population parameter is going *the other way around…*

…suppose I knew the population parameter, what would the statistic look like?

A first easy example from before

Population

Every single time in the history of the world anyone has ever flipped a **biased** coin twice

Population parameter

The "biasedness" of the coin; i.e., the probability p of the coin coming out heads

Parameter value

p

Sample

A single time someone flipped a coin twice

Statistic

The number of times heads came up those two times

Statistic distribution

Value x_i	$P(X = x_i)$
0	$(1 – p)^2$
1	$2p(1 – p)$
2	p^2

Side note: $(1-p)^2 + 2p(1-p) + p^2 = 1$, as expected…

A second example

Population

Every single person in the united states today

Population parameter

The proportion of people in the united states today who would answer "democrat" if asked "whom you would vote for if the election were held today"

Parameter value

p

Sample

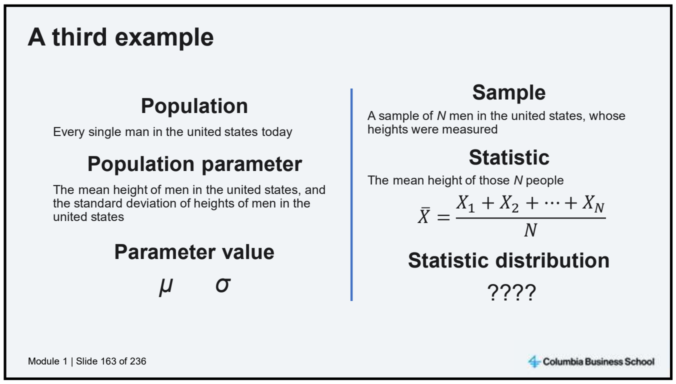
A survey carried out using N respondents, asking them whether they would vote democrat or republican if the vote happened today

Statistic

The number of these N people who said "democrat"

Statistic distribution

$$Binom(n = N, p = p)$$



A third example

Population

Every single man in the united states today

Population parameter

The mean height of men in the united states, and the standard deviation of heights of men in the united states

Parameter value

$\mu \quad \sigma$

Sample

A sample of *N* men in the united states, whose heights were measured

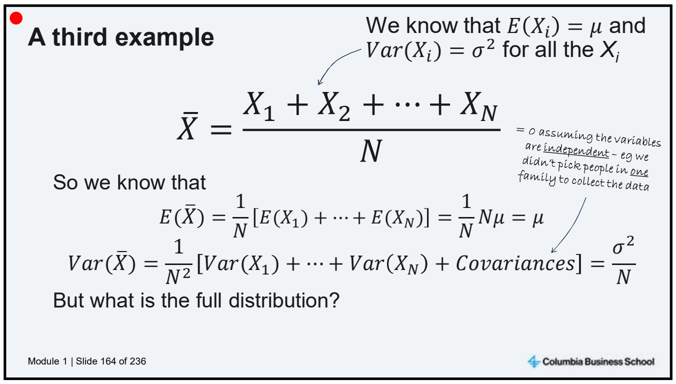
Statistic

The mean height of those *N* people

$$\bar{X} = \frac{X_1 + X_2 + \cdots + X_N}{N}$$

Statistic distribution

????



A third example

We know that $E(X_i) = \mu$ and $Var(X_i) = \sigma^2$ for all the X_i

$$\bar{X} = \frac{X_1 + X_2 + \cdots + X_N}{N}$$

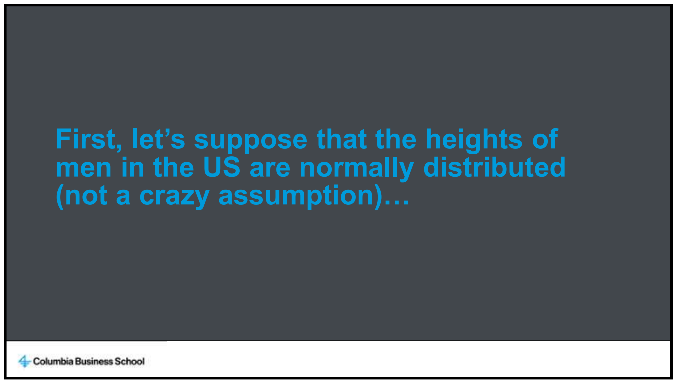
So we know that

$$E(\bar{X}) = \frac{1}{N}[E(X_1) + \cdots + E(X_N)] = \frac{1}{N}N\mu = \mu$$

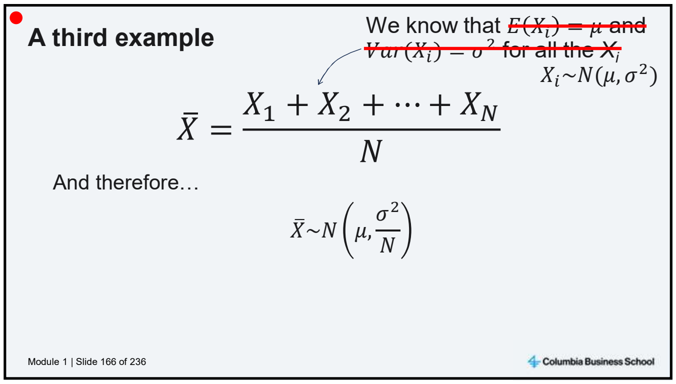
$$Var(\bar{X}) = \frac{1}{N^2}[Var(X_1) + \cdots + Var(X_N) + Covariances] = \frac{\sigma^2}{N}$$

= 0 assuming the variables are independent – eg we didn't pick people in one family to collect the data

But what is the full distribution?



First, let's suppose that the heights of men in the US are normally distributed (not a crazy assumption)…



A third example

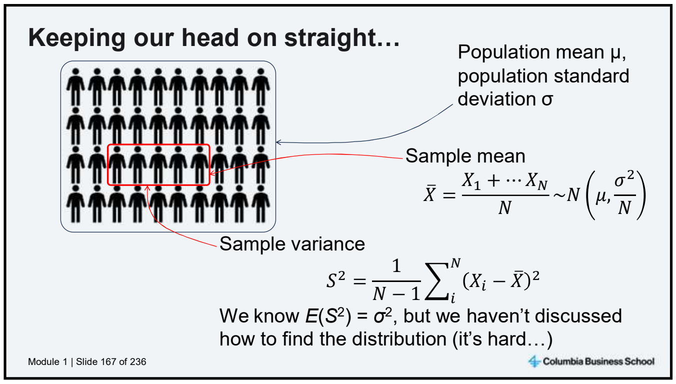
We know that ~~$E(X_i) = \mu$ and $Var(X_i) = \sigma^2$ for all the X_i~~

$X_i \sim N(\mu, \sigma^2)$

$$\bar{X} = \frac{X_1 + X_2 + \cdots + X_N}{N}$$

And therefore…

$$\bar{X} \sim N\left(\mu, \frac{\sigma^2}{N}\right)$$



Keeping our head on straight…

Population mean μ, population standard deviation σ

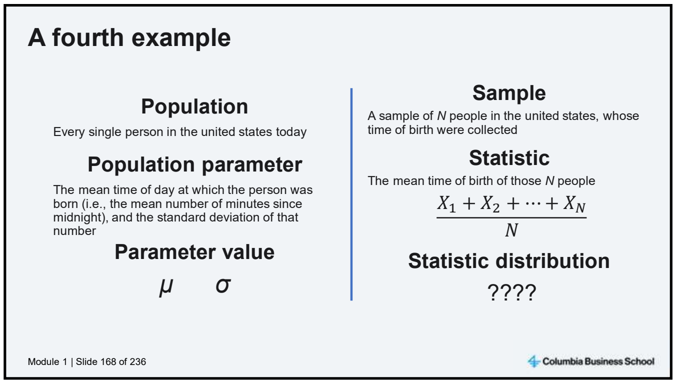
Sample mean

$$\bar{X} = \frac{X_1 + \cdots X_N}{N} \sim N\left(\mu, \frac{\sigma^2}{N}\right)$$

Sample variance

$$S^2 = \frac{1}{N-1}\sum_i^N (X_i - \bar{X})^2$$

We know *E*(*S*²) = σ², but we haven't discussed how to find the distribution (it's hard…)



A fourth example

Population

Every single person in the united states today

Population parameter

The mean time of day at which the person was born (i.e., the mean number of minutes since midnight), and the standard deviation of that number

Parameter value

$\mu \quad \sigma$

Sample

A sample of *N* people in the united states, whose time of birth were collected

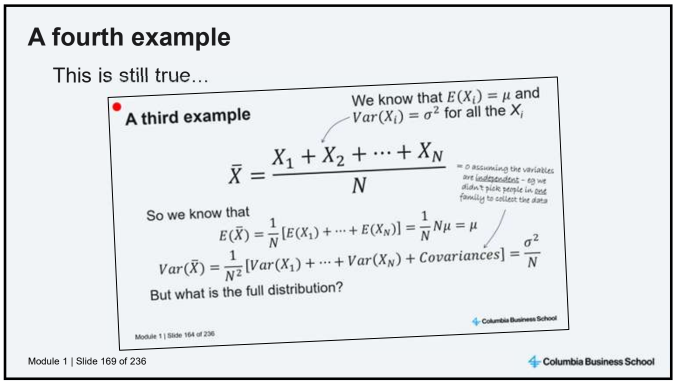
Statistic

The mean time of birth of those *N* people

$$\frac{X_1 + X_2 + \cdots + X_N}{N}$$

Statistic distribution

????



A fourth example

This is still true…

A third example

We know that $E(X_i) = \mu$ and $Var(X_i) = \sigma^2$ for all the X_i

$$\bar{X} = \frac{X_1 + X_2 + \cdots + X_N}{N}$$

= 0 assuming the variables are independent – eg we didn't pick people in one family to collect the data

So we know that

$$E(\bar{X}) = \frac{1}{N}[E(X_1) + \cdots + E(X_N)] = \frac{1}{N}N\mu = \mu$$

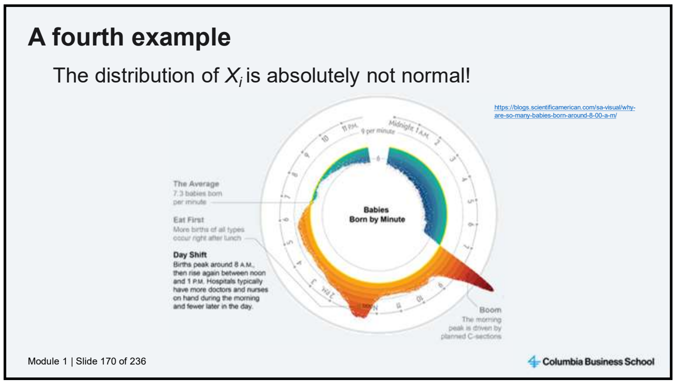
$$Var(\bar{X}) = \frac{1}{N^2}[Var(X_1) + \cdots + Var(X_N) + Covariances] = \frac{\sigma^2}{N}$$

But what is the full distribution?

Module 1 | Slide 164 of 236

Columbia Business School

Columbia Business School

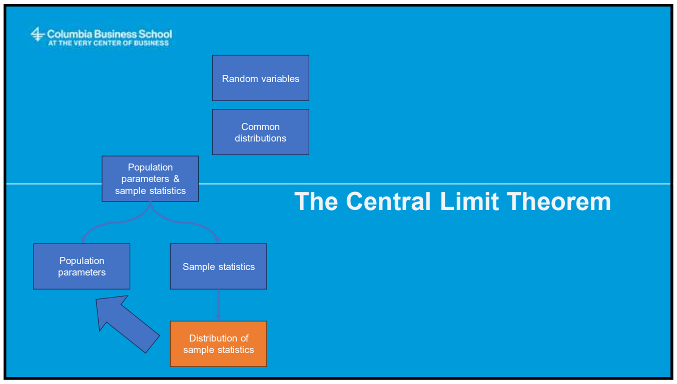


A fourth example

The distribution of X_i is absolutely not normal!

https://blogs.scientificamerican.com/sa-visual/why-are-so-many-babies-born-around-8-00-a-m/

Columbia Business School



Columbia Business School
AT THE VERY CENTER OF BUSINESS

Random variables

Common distributions

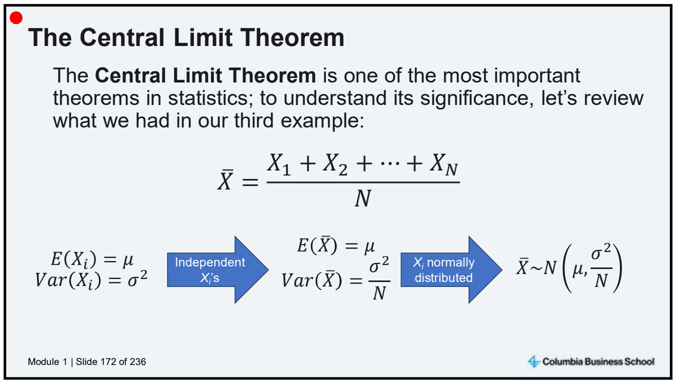
Population parameters & sample statistics

Population parameters

Sample statistics

Distribution of sample statistics

The Central Limit Theorem



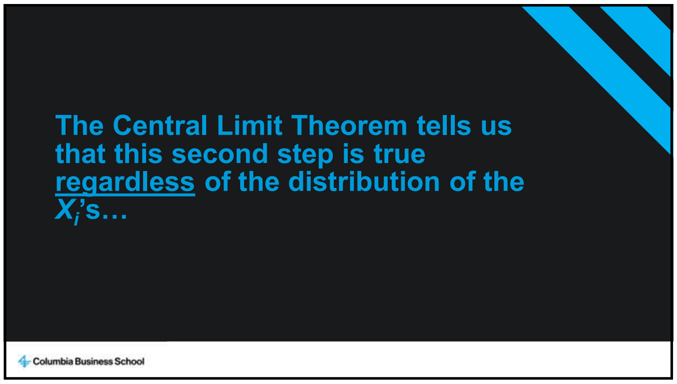
The Central Limit Theorem

The **Central Limit Theorem** is one of the most important theorems in statistics; to understand its significance, let's review what we had in our third example:

$$\bar{X} = \frac{X_1 + X_2 + \cdots + X_N}{N}$$

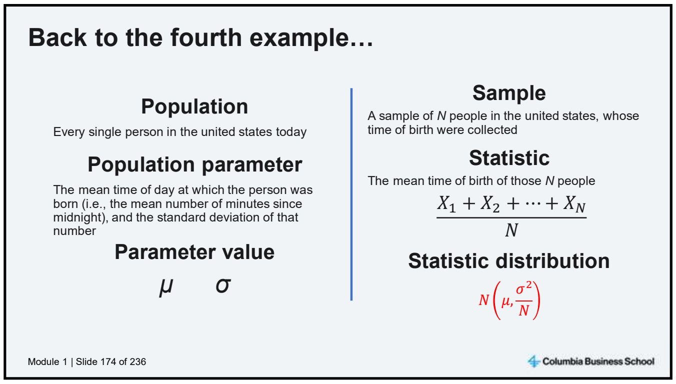
$$\begin{matrix} E(X_i) = \mu \\ Var(X_i) = \sigma^2 \end{matrix} \xrightarrow{\text{Independent } X_i\text{'s}} \begin{matrix} E(\bar{X}) = \mu \\ Var(\bar{X}) = \dfrac{\sigma^2}{N} \end{matrix} \xrightarrow{X_i \text{ normally distributed}} \bar{X} \sim N\left(\mu, \frac{\sigma^2}{N}\right)$$

Columbia Business School



The Central Limit Theorem tells us that this second step is true <u>regardless</u> of the distribution of the *X_i's*…

Columbia Business School



Back to the fourth example…

Population

Every single person in the united states today

Population parameter

The mean time of day at which the person was born (i.e., the mean number of minutes since midnight), and the standard deviation of that number

Parameter value

μ σ

Sample

A sample of *N* people in the united states, whose time of birth were collected

Statistic

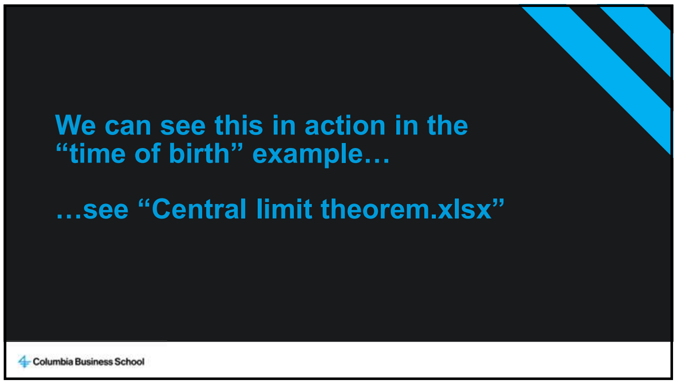
The mean time of birth of those *N* people

$$\frac{X_1 + X_2 + \cdots + X_N}{N}$$

Statistic distribution

$$N\left(\mu, \frac{\sigma^2}{N}\right)$$

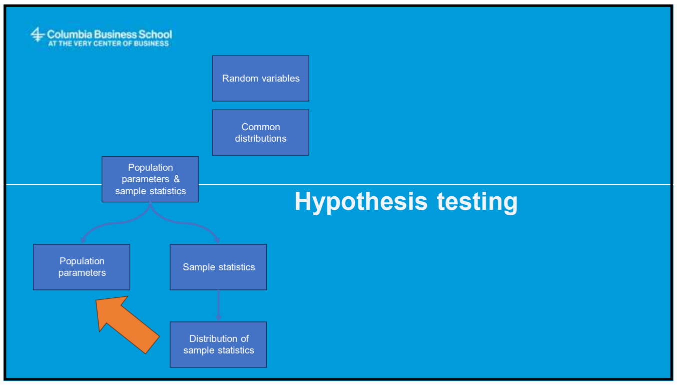
Columbia Business School



We can see this in action in the "time of birth" example…

…see "Central limit theorem.xlsx"

Columbia Business School



Columbia Business School
AT THE VERY CENTER OF BUSINESS

Random variables

Common distributions

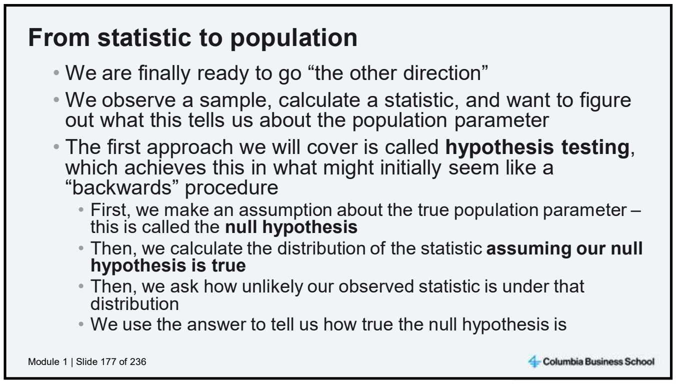
Population parameters & sample statistics

Population parameters

Sample statistics

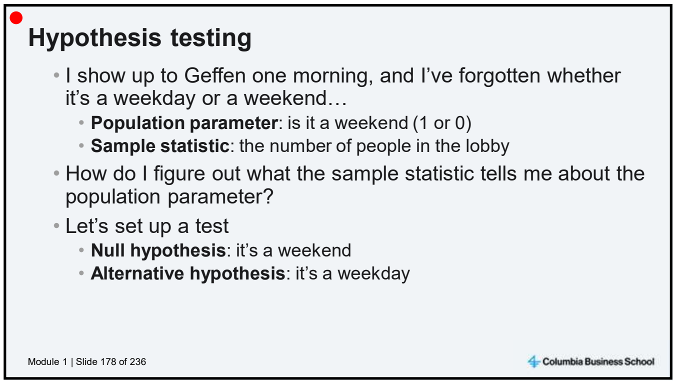
Distribution of sample statistics

Hypothesis testing



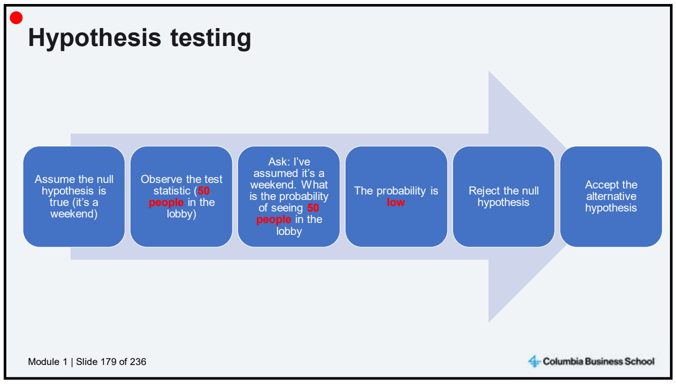
From statistic to population

- We are finally ready to go "the other direction"
- We observe a sample, calculate a statistic, and want to figure out what this tells us about the population parameter
- The first approach we will cover is called **hypothesis testing**, which achieves this in what might initially seem like a "backwards" procedure
  - First, we make an assumption about the true population parameter – this is called the **null hypothesis**
  - Then, we calculate the distribution of the statistic **assuming our null hypothesis is true**
  - Then, we ask how unlikely our observed statistic is under that distribution
  - We use the answer to tell us how true the null hypothesis is



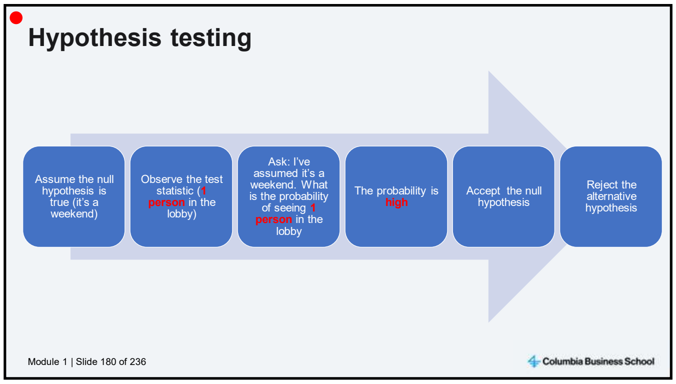
Hypothesis testing

- I show up to Geffen one morning, and I've forgotten whether it's a weekday or a weekend…
  - **Population parameter**: is it a weekend (1 or 0)
  - **Sample statistic**: the number of people in the lobby
- How do I figure out what the sample statistic tells me about the population parameter?
- Let's set up a test
  - **Null hypothesis**: it's a weekend
  - **Alternative hypothesis**: it's a weekday



Hypothesis testing

- Assume the null hypothesis is true (it's a weekend)
- Observe the test statistic (**50 people** in the lobby)
- Ask: I've assumed it's a weekend. What is the probability of seeing **50 people** in the lobby
- The probability is **low**
- Reject the null hypothesis
- Accept the alternative hypothesis



Hypothesis testing

- Assume the null hypothesis is true (it's a weekend)
- Observe the test statistic (**1 person** in the lobby)
- Ask: I've assumed it's a weekend. What is the probability of seeing **1 person** in the lobby
- The probability is **high**
- Accept the null hypothesis
- Reject the alternative hypothesis

Is the height of men in this room different than average?

- The mean height of men in the USA is 70 inches; the standard deviation is 3 inches
- You measure the height of 10 men in this room and calculate the sample mean $\bar{X}$ – you find it is 71 inches

$$H_0 \text{ (null hypothesis) : } \mu = 70$$
$$H_1 \text{ (alternative hypothesis) : } \mu \neq 70$$

Step 1: assume the null hypothesis is true; $\mu = 70$

What statistic should we use?

- What statistic should we use to test this hypothesis?
- In theory, we could use $\bar{X}$ itself (71)
- In practice, it is more common to use the so-called *Z*-score, which calculates the sample mean, minus the population mean, divided by the population standard deviation divided by the square root of the number of observations

$$\frac{\bar{X} - \mu}{\sigma/\sqrt{n}}$$

- In this case, our sample mean was 71, the population mean is 70, the population standard deviation is 3, and $n = 10$
- So the statistic here is 1.054

The *Z*-score

- Assuming the null hypothesis ($\mu = 70$) is true…
- …what would the distribution of $\bar{X}$ be?

$$\bar{X} \sim N\left(\mu, \left[\frac{\sigma}{\sqrt{n}}\right]^2\right)$$

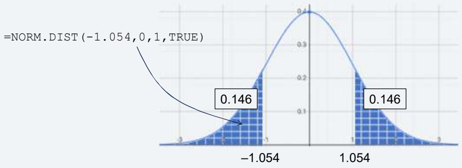
- And therefore, what would the distribution of the test statistic (Z) be?

$$Z = \frac{\bar{X} - \mu}{\sigma/\sqrt{n}} \sim N(0,1)$$

- This is why the Z statistic is so useful

Testing the null hypothesis

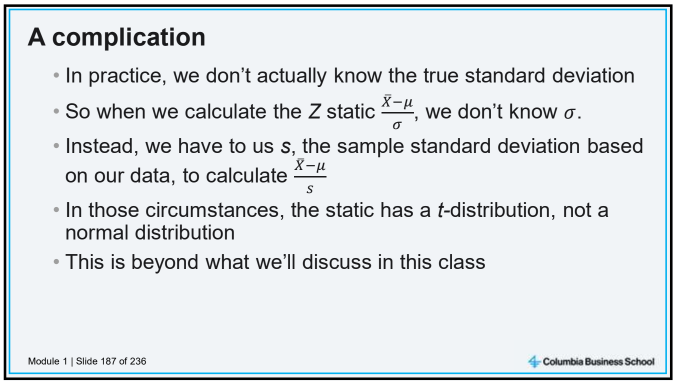
- OK, so assuming the null hypothesis is true, $Z \sim N(0, 1)$.
- We observed $Z = 1.054$. What is the probability of observing this kind of deviation if the null hypothesis were true?



Answer: 0.29

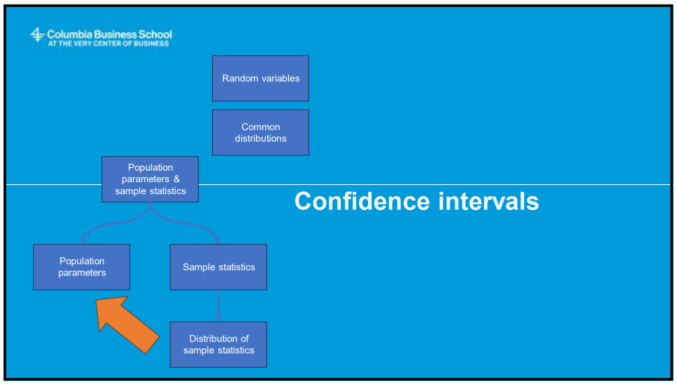
Testing the null hypothesis

- If the null hypothesis is true, there is a **29% chance** of observing our **sample mean** of **71**
  - This is called the ***p-value*** of the test
- That's quite a high probability
- So we **accept the null hypothesis!** Heights in this room are **no different than the average in the country**
- What we count as a "high probability" is somewhat arbitrary – traditionally, we use 5% – 0.05
- If the *p*-value had been *smaller* than 0.05, we conclude the null hypothesis is **very** unlikely, and we **reject** it



A complication

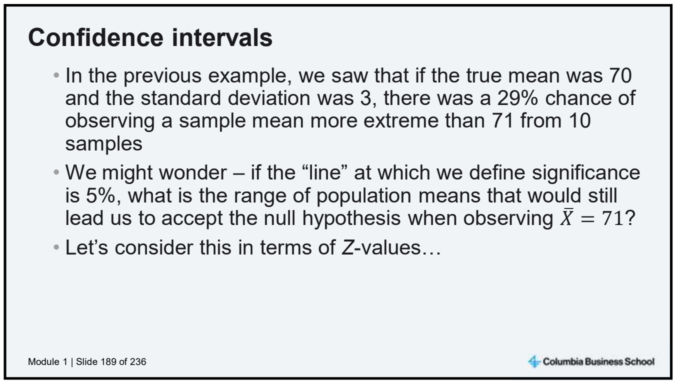
- In practice, we don't actually know the true standard deviation
- So when we calculate the *Z* static $\frac{\bar{X}-\mu}{\sigma}$, we don't know σ.
- Instead, we have to us *s*, the sample standard deviation based on our data, to calculate $\frac{\bar{X}-\mu}{s}$
- In those circumstances, the static has a *t*-distribution, not a normal distribution
- This is beyond what we'll discuss in this class



Columbia Business School AT THE VERY CENTER OF BUSINESS

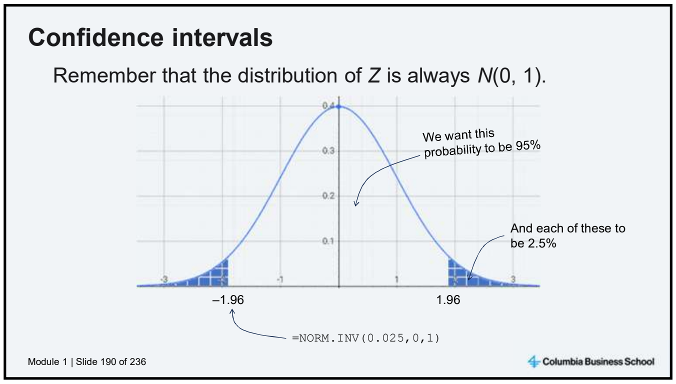
Random variables
Common distributions
Population parameters & sample statistics
Population parameters
Sample statistics
Distribution of sample statistics

Confidence intervals



Confidence intervals

- In the previous example, we saw that if the true mean was 70 and the standard deviation was 3, there was a 29% chance of observing a sample mean more extreme than 71 from 10 samples
- We might wonder – if the "line" at which we define significance is 5%, what is the range of population means that would still lead us to accept the null hypothesis when observing $\bar{X} = 71$?
- Let's consider this in terms of *Z*-values…



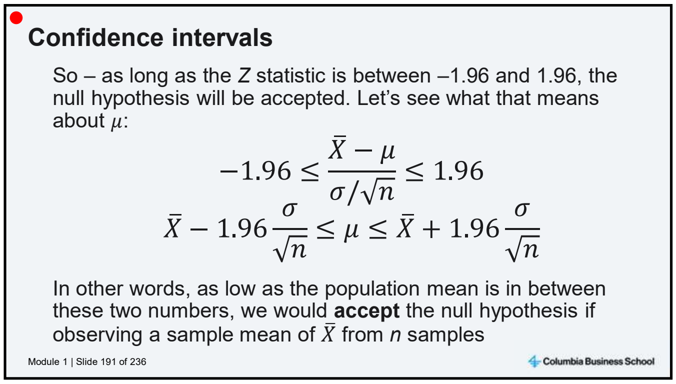
Confidence intervals

Remember that the distribution of *Z* is always *N*(0, 1).

We want this probability to be 95%

And each of these to be 2.5%

```
=NORM.INV(0.025,0,1)
```



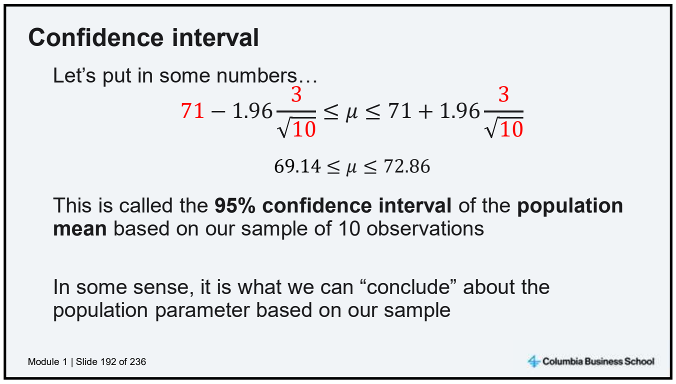
Confidence intervals

So – as long as the *Z* statistic is between –1.96 and 1.96, the null hypothesis will be accepted. Let's see what that means about μ:

$$-1.96 \leq \frac{\bar{X}-\mu}{\sigma/\sqrt{n}} \leq 1.96$$

$$\bar{X} - 1.96\frac{\sigma}{\sqrt{n}} \leq \mu \leq \bar{X} + 1.96\frac{\sigma}{\sqrt{n}}$$

In other words, as low as the population mean is in between these two numbers, we would **accept** the null hypothesis if observing a sample mean of $\bar{X}$ from *n* samples



Confidence interval

Let's put in some numbers…

$$71 - 1.96\frac{3}{\sqrt{10}} \leq \mu \leq 71 + 1.96\frac{3}{\sqrt{10}}$$

$$69.14 \leq \mu \leq 72.86$$

This is called the **95% confidence interval** of the **population mean** based on our sample of 10 observations

In some sense, it is what we can "conclude" about the population parameter based on our sample

We have finally achieved our aim!

We can go from our sample to a conclusion about our population parameter

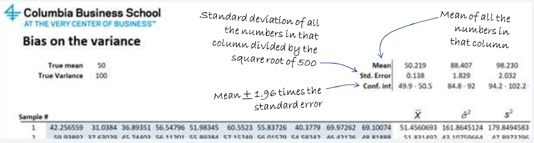
The confidence interval

Let X be a normally distributed random variable. Let $X_1, X_2, \cdots X_N$ be independent samples of this random variable. A 95% confidence interval on the population mean can be calculated as

$$\bar{X} \pm 1.96 \cdot \frac{s}{\sqrt{N}}$$

In reality, we'd need to use the t distribution instead of the normal distribution to get this number, because we're using s instead of σ, but it's common to just us 1.96 anyway

Practicing with Cis in Excel



Columbia Business School
AT THE VERY CENTER OF BUSINESS™

Bias on the variance

True mean	50
True Variance	100

Mean	50.219	88.407	98.230
Std. Error	0.138	1.829	2.032
Conf. int	49.9 - 50.5	84.8 - 92	94.2 - 102.2

Sample #											$\bar{X}$	$\hat{\sigma}^2$	s^2
1	42.256559	31.0384	36.89351	56.54796	51.98345	60.5523	55.83726	40.3779	69.97262	69.10074	51.4560693	161.8645124	179.8494583
2	59.93892	37.63029	45.74403	56.11301	55.80384	57.15749	56.01579	54.58242	46.42136	48.81888	51.831402	43.10759664	47.8973796

Standard deviation of all the numbers in that column divided by the square root of 500

Mean of all the numbers in that column

Mean ± 1.96 times the standard error

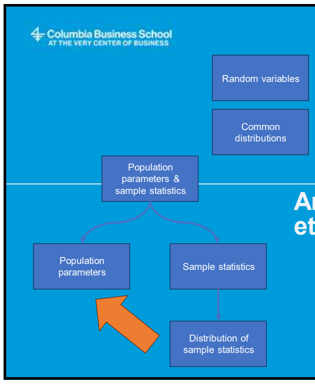
500 samples, each with 10 normally distributed points with a mean of 50 and a variance of 100

The mean of the 10 points in that row

The estimated variance for the 10 points in that row dividing by N (10)

The estimated variance for the 10 points in that row dividing by N – 1 (9)

Columbia Business School
AT THE VERY CENTER OF BUSINESS


Random variables
Common distributions
Population parameters & sample statistics
Population parameters
Sample statistics
Distribution of sample statistics

An application: dishwasher etiquette

Dishwasher etiquette

martha stewart

Should You Point Silverware Up or Down in the Dishwasher? 3 Experts Weigh In

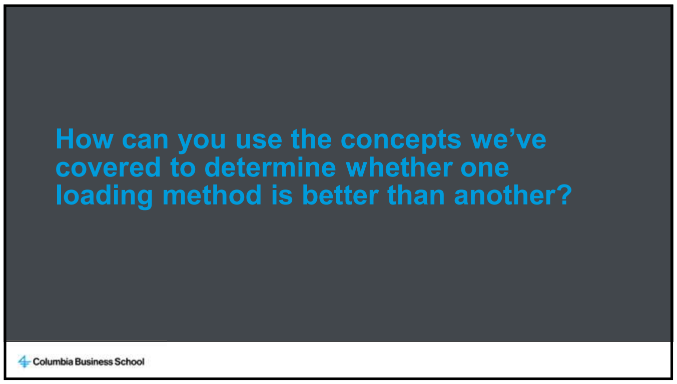
This highly contested debate ultimately comes down to personal preference.

By Nashia Baker and Madeline Buiano | Updated on August 9, 2023

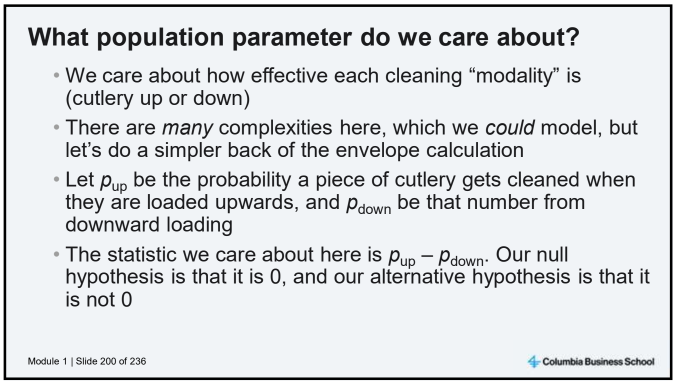
An experiment

You carry out experiments to find the best way to load your dishwasher

Experiment #	Loading style	Total pieces	Clean pieces
1	Up	20	19
2	Up	25	25
3	Up	19	17
4	Down	23	21
5	Up	20	18
6	Down	28	21
7	Down	19	18
8	Down	23	22
9	Up	25	23
10	Down	30	30

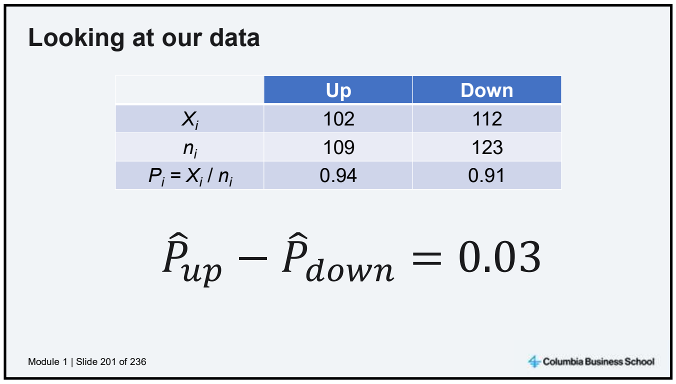


How can you use the concepts we've covered to determine whether one loading method is better than another?



What population parameter do we care about?

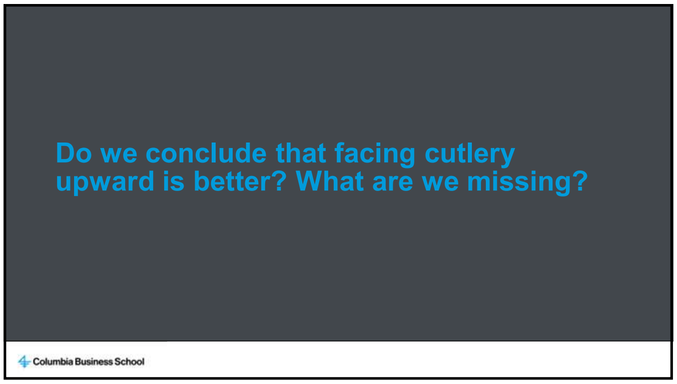
- We care about how effective each cleaning "modality" is (cutlery up or down)
- There are *many* complexities here, which we *could* model, but let's do a simpler back of the envelope calculation
- Let p_{up} be the probability a piece of cutlery gets cleaned when they are loaded upwards, and p_{down} be that number from downward loading
- The statistic we care about here is $p_{up} - p_{down}$. Our null hypothesis is that it is 0, and our alternative hypothesis is that it is not 0



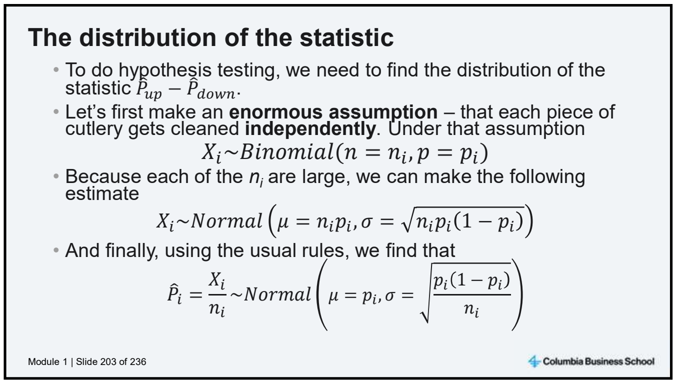
Looking at our data

	Up	Down
X_i	102	112
n_i	109	123
$P_i = X_i / n_i$	0.94	0.91

$$\hat{P}_{up} - \hat{P}_{down} = 0.03$$



Do we conclude that facing cutlery upward is better? What are we missing?



The distribution of the statistic

- To do hypothesis testing, we need to find the distribution of the statistic $\hat{P}_{up} - \hat{P}_{down}$.
- Let's first make an **enormous assumption** – that each piece of cutlery gets cleaned **independently**. Under that assumption

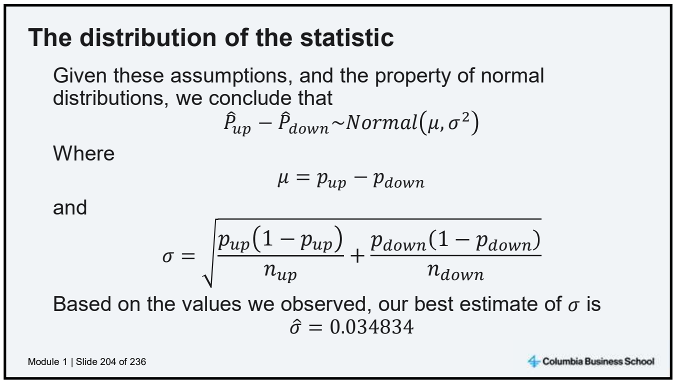
$$X_i \sim Binomial(n = n_i, p = p_i)$$

- Because each of the n_i are large, we can make the following estimate

$$X_i \sim Normal\left(\mu = n_i p_i, \sigma = \sqrt{n_i p_i (1 - p_i)}\right)$$

- And finally, using the usual rules, we find that

$$\hat{P}_i = \frac{X_i}{n_i} \sim Normal\left(\mu = p_i, \sigma = \sqrt{\frac{p_i(1 - p_i)}{n_i}}\right)$$



The distribution of the statistic

Given these assumptions, and the property of normal distributions, we conclude that

$$\hat{P}_{up} - \hat{P}_{down} \sim Normal\left(\mu, \sigma^2\right)$$

Where

$$\mu = p_{up} - p_{down}$$

and

$$\sigma = \sqrt{\frac{p_{up}\left(1 - p_{up}\right)}{n_{up}} + \frac{p_{down}(1 - p_{down})}{n_{down}}}$$

Based on the values we observed, our best estimate of σ is

$$\hat{\sigma} = 0.034834$$

Hypothesis testing

- Our hypotheses are as follows

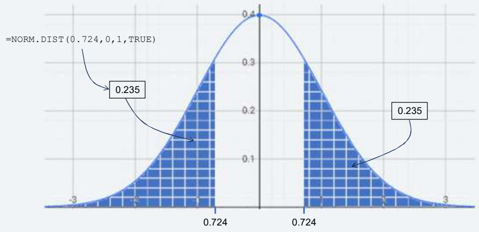
$$H_0 : p_{up} - p_{down} = 0$$
$$H_1 : p_{up} - p_{down} \neq 0$$

- Our observed test statistic is $\hat{p}_{up} - \hat{p}_{down} = 0.025211$, with a *z*-value of

$$\frac{0.025211 - 0}{0.034834} = 0.723742$$

- Let's see what the probability is of observing a deviation from the mean this high if the null hypothesis were true…

Hypothesis testing



The probability of observing a deviation this extreme if the null hypothesis was true is 2 ×0.235 = 0.47 – the *p*-value for our test

Hypothesis testing

- The probability of observing a deviation this extreme if the null hypothesis was true is 2 ×0.235 = 0.47 – the *p*-value for our test
- This is quite a high probability, so we do **not** reject the null hypothesis
- We conclude that the null hypothesis is true – the direction makes **no difference** to cleaning ability

Confidence intervals

- We can also calculate a **confidence interval** on the population parameter $p_{up} - p_{down}$
- We can do this using our normal approximation

$$\hat{p}_{up} - \hat{p}_{down} \sim Normal(\mu, \sigma^2)$$

with $\hat{\sigma} = 0.034834$

- We would accept our null hypothesis as long as the true population parameter was between

$$0.025211 \pm 1.96 \times 0.034834$$

Calculating, we get a CI of

–0.0431 to 0.0935

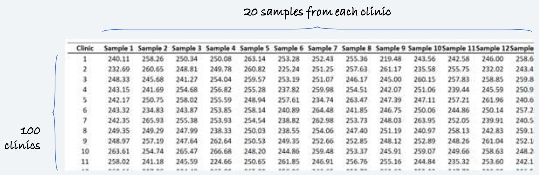
Columbia Business School
AT THE VERY CENTER OF BUSINESS

Hypothesis testing in action – a COVID testing example (optional)

A mini-case

- You are working at a chain of clinics dispensing Moderna COVID vaccines
- Each Moderna injection should contain **250 μg of vaccine**; the correct doses are **normally** distributed with **mean 250 μg** and **standard deviation 10 μg**
- It has come to your attention that due to **a typo in instructions**, some of your clinics have been systematically administering **too much vaccine per syringe**
- There are **no adverse effects on health** (the large doses are still within allowable volumes) but in aggregate, this **wastes supply** of precious vaccines
- Unfortunately, the instructions have all been thrown out so you can't check them, but you have samples of **20 syringes** from **each of your 100 clinics**

Data



20 samples from each clinic

100 clinics

Clinic	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample 6	Sample 7	Sample 8	Sample 9	Sample 10	Sample 11	Sample 12	Sample
1	240.11	258.26	250.34	250.08	263.14	253.28	252.43	255.36	219.48	243.56	242.58	246.00	258.6
2	232.69	260.65	248.81	249.78	260.82	225.24	251.25	257.63	261.17	235.58	255.75	232.02	243.4
3	248.33	245.68	241.27	254.04	259.57	253.19	251.07	246.17	245.00	260.15	257.83	258.85	259.8
4	243.15	241.69	254.68	256.82	255.28	237.82	259.98	254.51	242.07	251.06	239.44	245.59	250.9
5	242.17	250.75	258.02	255.59	248.94	257.61	234.74	263.47	247.39	247.11	257.21	261.96	240.6
6	243.32	234.83	243.87	253.85	258.14	240.89	264.48	241.85	246.75	250.06	244.86	250.14	257.2
7	242.35	265.93	255.38	253.93	254.54	238.82	262.98	253.73	248.03	263.95	252.05	239.91	240.5
8	249.35	249.29	247.99	238.33	250.03	238.55	254.06	247.40	251.19	240.97	258.13	242.83	259.1
9	248.97	257.19	247.64	262.64	250.53	249.35	252.66	252.85	248.12	252.89	248.26	261.04	252.1
10	263.61	254.74	265.47	266.68	248.20	244.86	259.48	253.37	245.91	259.07	249.66	258.63	248.2
11	258.02	241.18	245.59	224.66	250.65	261.85	246.91	256.76	255.16	244.84	235.32	253.60	242.1

How would you use the 20-syringe sample from each clinic to figure out whether the clinic had incorrect instructions?

Hypothesis tests

We want to carry out the following hypothesis test for each clinic

- **Null hypothesis (H_0)**: mean dose is **250 µg**
- **Alternative hypothesis (H_1)**: mean dose is **> 250 µg**

The test statistic is the mean of the 20 doses at each clinic. Under the **null hypothesis**, the distribution of the test statistic is

$$N\left(\mu = 250, \sigma = \frac{10}{\sqrt{20}}\right)$$

p-values

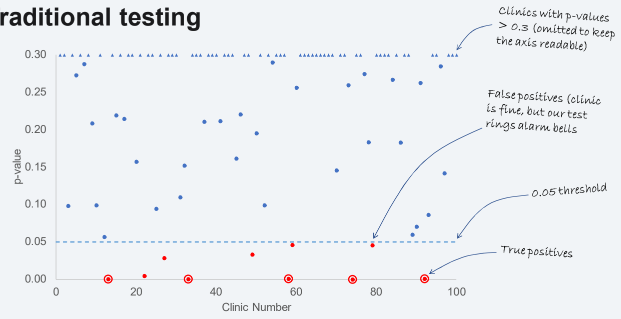
Suppose the average dose observed at clinic *i* is $\bar{X}_i$. The *p*-value associated with this test statistic is

$$P\left(\text{Observing } \bar{X}_i \text{ or worse} \,|\, H_0 \text{ is true}\right)$$
$$= P\left(N\left(\mu = 250, \sigma = \frac{10}{\sqrt{20}}\right) \geq \bar{X}_i\right)$$
$$= 1 - P\left(Z \leq \frac{\bar{X}_i - 250}{10/\sqrt{20}}\right)$$

The traditional *p*-value test would have us reject null hypotheses for al clinics with *p* < 0.05

What are some issues with doing this?

Traditional testing



Clinics with p-values > 0.3 (omitted to keep the axis readable)

False positives (clinic is fine, but our test rings alarm bells

0.05 threshold

True positives

The problem with multiple testing

- $p < 0.05$ means that there is a < 5% chance of observing such a large test statistic if the null hypothesis were true
- The problem is that we're doing 100 tests
- Intuitively, if there is a 5% chance each null hypothesis will be falsely rejected and we do it 100 times, there's a very high chance at least one of our 100 hypotheses will be falsely rejected
- So we are almost guaranteed to have some perfectly true null hypotheses be rejected
- In practice: clinics that had correct instructions that will be flagged as problematic

Getting mathematical

Suppose we have *N* tests total, we reject any *p*-value ≤ α, and we make no assumptions about the tests. Then:

$$
\begin{aligned}
&P(\text{Any hypothesis rejected} \mid \text{All } H_0 \text{ true}) \\
&\quad = P\left(\text{At least one } p \text{ value} \le \alpha \mid \text{All } H_0 \text{ true}\right) \\
&\quad \le \sum_{i=1}^{N} P\left(p \text{ value } i \le \alpha \mid i^{\text{th}}\ H_0 \text{ true}\right) \\
&\quad = \sum_{i=1}^{N} \alpha \\
&\quad = N\alpha
\end{aligned}
$$

In our case, N = 100 and α = 5% so we can guarantee this probably will be ≤ 5... Thanks a lot!!

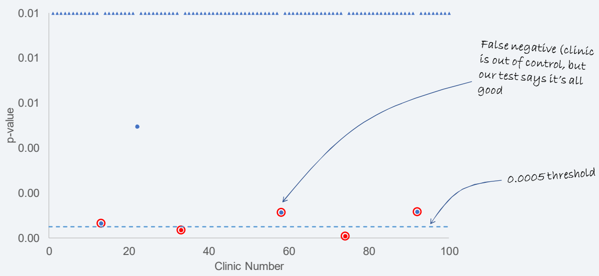
Because P(A or B) = P(A) + P(B) – P(A and B)

The Bonferroni Correction

- We can only guarantee the probability of incorrectly rejecting a null hypothesis is ≤ *Nα*
- The Bonferroni Correction basically says "I want to guarantee this is ≤ 0.05"
- For this to be true, the α for each individual hypothesis should be 0.05/100 = 0.0005

Any issues with the Bonferroni Correction approach?

The Bonferroni Correction



False negative (clinic is out of control, but our test says it's all good

0.0005 threshold

How can we fix this?

The Benjamini-Hochberg procedure

- The BH procedure **changes the question** completely
- Instead of asking "what is the probability of incorrectly rejecting **any null hypothesis**", it asks "what is the **proportion of rejected null hypotheses** that were actually **true**"
- Framed in the language of our case
  - The **Bonferroni Correction** asks "what is the probability **any** clinic that is fine will be flagged as out of control"
  - The **BH procedure** asks "of all the clinics that **flagged** as out of control, how many of them **deserved it**"
- Clearly, the BH question is far more relevant in many business applications

The Bonferroni Correction controls the probability any null hypothesis is rejected. This is called the familywise error rate (FWER)

The Benjamini-Hochberg procedure controls the proportion of rejected null hypotheses that are incorrectly rejected. This is called the false discovery proportion (FDP)

The Benjamini-Hochberg procedure

Suppose you want to ensure that no more than a **proportion α** of **rejected null hypotheses** were **actually true**

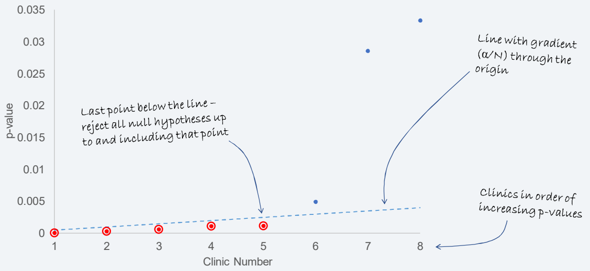
- **Step 1**: sort all the *p*-values from smallest to largest

$$p_{(1)} \le p_{(2)} \le \cdots \le p_{(N)}$$

- **Step 2**: start from $p_{(1)}$ and work your way upwards; for each *p*-value, check whether $p_{(k)} \le (\alpha/N)k$, where *N* is the **total number of hypotheses**. Let the largest *p* value for which this is true be p^*
- **Step 3**: reject all null hypotheses with $p \le p^*$

That's a lot of words… Let's see it in practice…

The Benjamini-Hochberg procedure

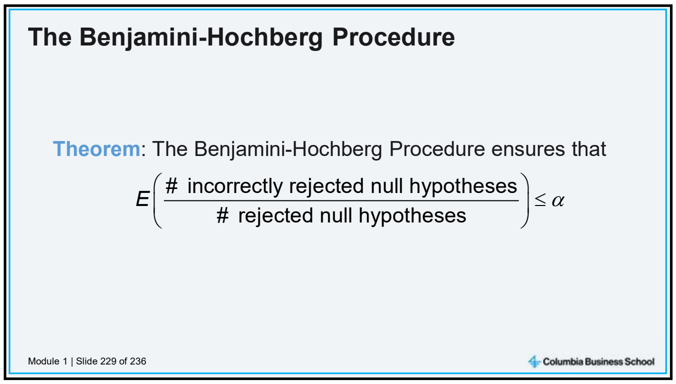


Last point below the line – reject all null hypotheses up to and including that point

Line with gradient (α/N) through the origin

Clinics in order of increasing p-values

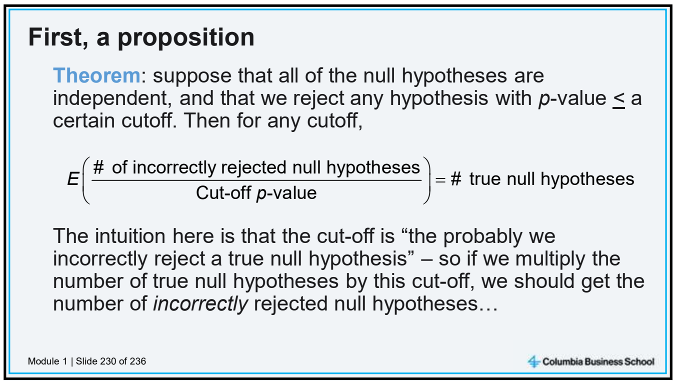
Why does this work?!



The Benjamini-Hochberg Procedure

Theorem: The Benjamini-Hochberg Procedure ensures that

$$E\left(\frac{\#\text{ incorrectly rejected null hypotheses}}{\#\text{ rejected null hypotheses}}\right) \le \alpha$$

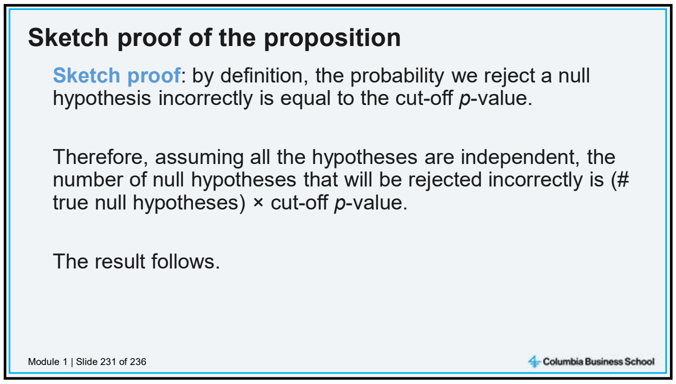


First, a proposition

Theorem: suppose that all of the null hypotheses are independent, and that we reject any hypothesis with *p*-value ≤ a certain cutoff. Then for any cutoff,

$$E\left(\frac{\#\text{ of incorrectly rejected null hypotheses}}{\text{Cut-off } p\text{-value}}\right) = \#\text{ true null hypotheses}$$

The intuition here is that the cut-off is "the probably we incorrectly reject a true null hypothesis" – so if we multiply the number of true null hypotheses by this cut-off, we should get the number of *incorrectly* rejected null hypotheses…

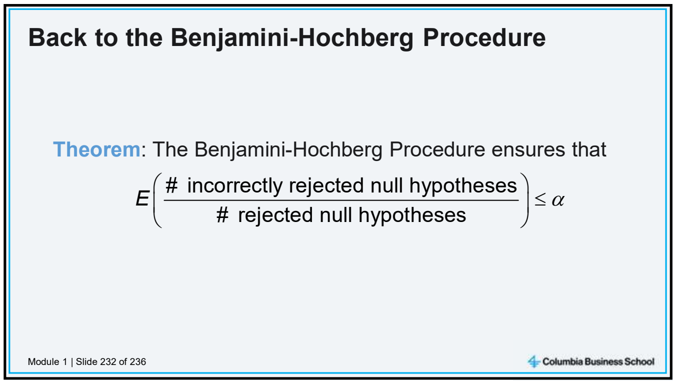


Sketch proof of the proposition

Sketch proof: by definition, the probability we reject a null hypothesis incorrectly is equal to the cut-off *p*-value.

Therefore, assuming all the hypotheses are independent, the number of null hypotheses that will be rejected incorrectly is (# true null hypotheses) × cut-off *p*-value.

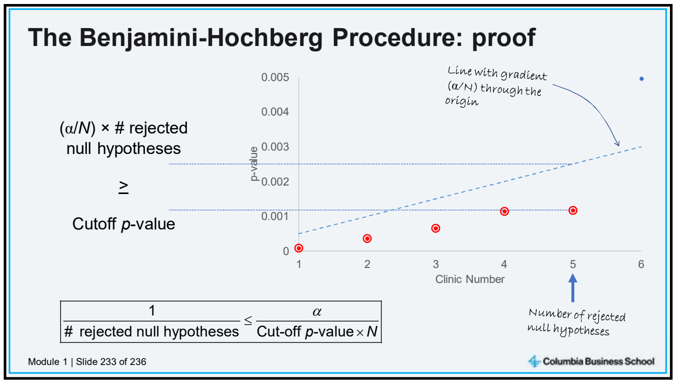
The result follows.



Back to the Benjamini-Hochberg Procedure

Theorem: The Benjamini-Hochberg Procedure ensures that

$$E\left(\frac{\#\text{ incorrectly rejected null hypotheses}}{\#\text{ rejected null hypotheses}}\right) \le \alpha$$



The Benjamini-Hochberg Procedure: proof

(α/*N*) × # rejected null hypotheses

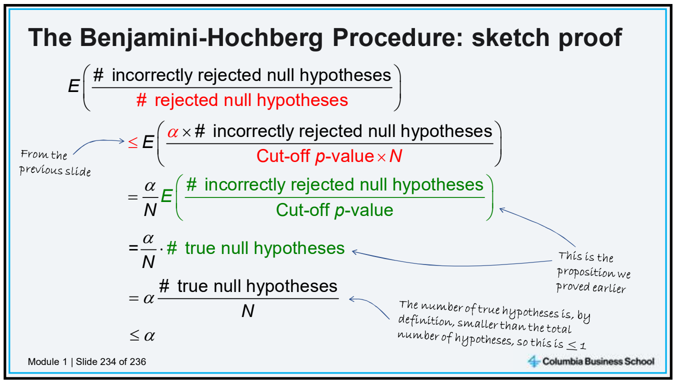
≥

Cutoff *p*-value

Line with gradient (α/N) through the origin

Number of rejected null hypotheses

$$\boxed{\frac{1}{\#\text{ rejected null hypotheses}} \le \frac{\alpha}{\text{Cut-off } p\text{-value} \times N}}$$



The Benjamini-Hochberg Procedure: sketch proof

$$\begin{aligned}
&E\left(\frac{\#\text{ incorrectly rejected null hypotheses}}{\#\text{ rejected null hypotheses}}\right)\\
&\le E\left(\frac{\alpha \times \#\text{ incorrectly rejected null hypotheses}}{\text{Cut-off } p\text{-value} \times N}\right)\\
&= \frac{\alpha}{N} E\left(\frac{\#\text{ incorrectly rejected null hypotheses}}{\text{Cut-off } p\text{-value}}\right)\\
&= \frac{\alpha}{N} \cdot \#\text{ true null hypotheses}\\
&= \alpha \frac{\#\text{ true null hypotheses}}{N}\\
&\le \alpha
\end{aligned}$$

From the previous slide

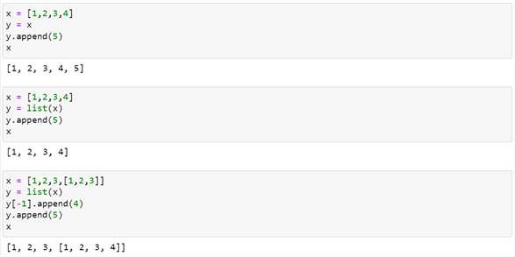
This is the proposition we proved earlier

The number of true hypotheses is, by definition, smaller than the total number of hypotheses, so this is ≤ 1

Columbia Business School
AT THE VERY CENTER OF BUSINESS

One last point: pointers in Python

Pointers in Python



```
x = [1,2,3,4]
y = x
y.append(5)
x
```

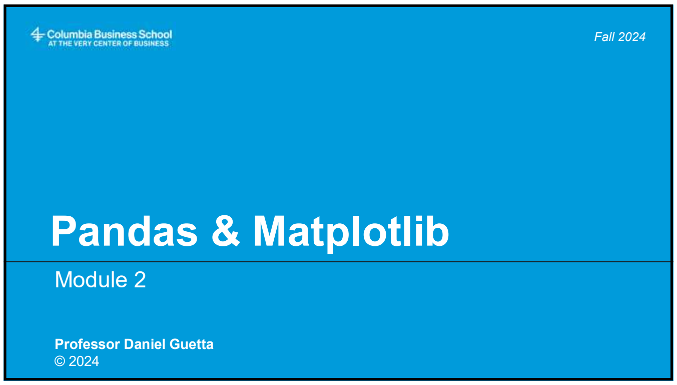
```
[1, 2, 3, 4, 5]
```

```
x = [1,2,3,4]
y = list(x)
y.append(5)
x
```

```
[1, 2, 3, 4]
```

```
x = [1,2,3,[1,2,3]]
y = list(x)
y[-1].append(4)
y.append(5)
x
```

```
[1, 2, 3, [1, 2, 3, 4]]
```



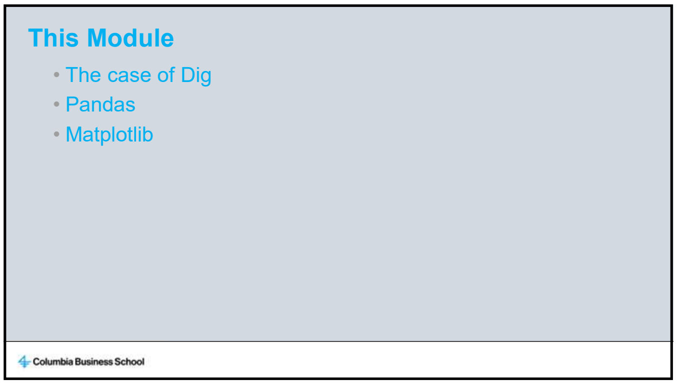
Columbia Business School
AT THE VERY CENTER OF BUSINESS

Fall 2024

Pandas & Matplotlib

Module 2

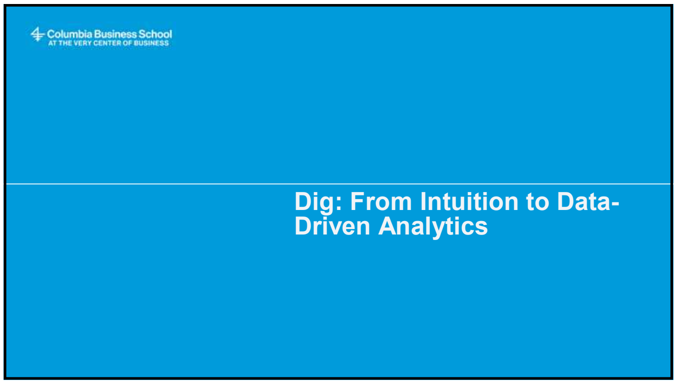
Professor Daniel Guetta



This Module

- The case of Dig
- Pandas
- Matplotlib

Columbia Business School



Columbia Business School
AT THE VERY CENTER OF BUSINESS

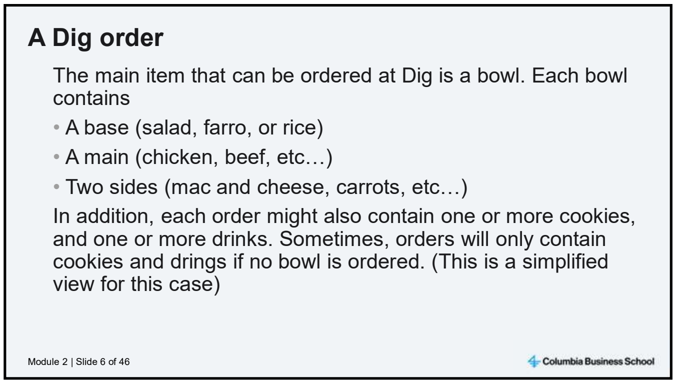
Dig: From Intuition to Data-Driven Analytics

Dig

Columbia Business School

Dig

Columbia Business School



A Dig order

The main item that can be ordered at Dig is a bowl. Each bowl contains

- A base (salad, farro, or rice)
- A main (chicken, beef, etc…)
- Two sides (mac and cheese, carrots, etc…)

In addition, each order might also contain one or more cookies, and one or more drinks. Sometimes, orders will only contain cookies and drings if no bowl is ordered. (This is a simplified view for this case)

Columbia Business School

Simulated Dig data

The main table we'll use in our introduction to Pandas is `BA orders.zip`, with the following columns

- `ORDER_ID`: ID of the order
- `DATETIME`: the date and time the order was placed
- `RESTAURANT`: the name of the restaurant at which the order was made
- `TYPE`: the order type (`IN_STORE`, `PICKUP`, or `DELIVERY`)
- `DRINK`: the number of drinks in the order
- `COOKIES`: the number of cookies in the order
- `MAIN`, `BASE`, `SIDE_1`, `SIDE_2`: the main, base, and sides in the bowl (these are missing if the order does not include a bowl)
- `ORDER_TIME`: how long it took to process the order (either in the store or digitally)

This file is impossible to open in Excel – too many rows!

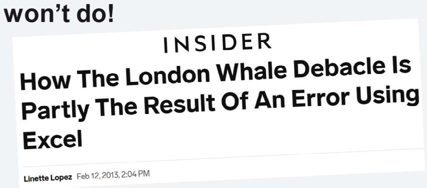
Introducing Pandas

When Excel just won't do!

Why go beyond Excel?

- **Scale**: dealing with really large data
- **Robustness**: it can be exceptionally difficult to get a "big picture" idea of what a large/complex Excel workbook is doing
- **Automation**: automating repetitive tasks many times, or on many files
- **Integration**: Python is a "real" programming language, and allows your data work to interact with other systems

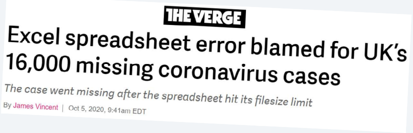
When Excel just won't do!



INSIDER

How The London Whale Debacle Is Partly The Result Of An Error Using Excel

Linette Lopez Feb 12, 2013, 2:04 PM



THE VERGE

Excel spreadsheet error blamed for UK's 16,000 missing coronavirus cases

The case went missing after the spreadsheet hit its filesize limit

By James Vincent | Oct 5, 2020, 9:41am EDT

Important note

This won't be a comprehensive introduction to Pandas. We'll only introduce the bits we'll need for this class. You'll notice we'll include more obscure parts and leave out more straightforward parts, simply because we want to cover everything we'll need in this class, but no more.

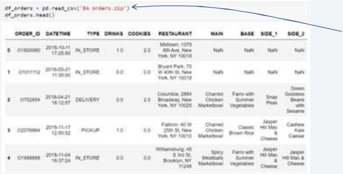
In later lectures, you can always return to these slides to look up any features we use that you are unfamiliar with.

Importing Pandas and loading a file

Tell Python we're going to need Pandas

```
import pandas as pd
```

Load the Dig file



```
df_orders = pd.read_csv('BA orders.zip')
df_orders.head()
```

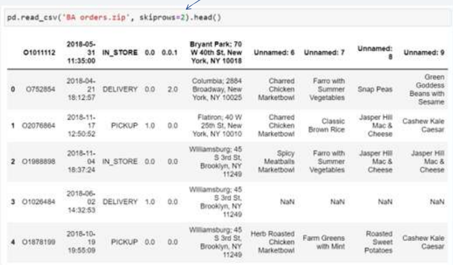
	ORDER_ID	DATETIME	TYPE	DRINKS	COOKIES	RESTAURANT	MAIN	BASE	SIDE_1	SIDE_2
0	O1820060	2018-10-11 17:25:50	IN_STORE	1.0	2.0	Midtown, 1379 6th Ave, New York, NY 10019	NaN	NaN	NaN	NaN
1	O1011112	2018-05-31 11:35:00	IN_STORE	0.0	0.0	Bryant Park, 70 W 40th St, New York, NY 10018	NaN	NaN	NaN	NaN
2	O752854	2018-04-21 18:12:57	DELIVERY	0.0	2.0	Columbia, 2884 Broadway, New York, NY 10025	Charred Chicken Marketbowl	Farro with Summer Vegetables	Snap Peas	Green Goddess Beans with Sesame
3	O2076864	2018-11-17 12:50:52	PICKUP	1.0	0.0	Flatiron, 40 W 25th St, New York, NY 10010	Charred Chicken Marketbowl	Classic Brown Rice	Jasper Hill Mac & Cheese	Cashew Kale Caesar
4	O1988898	2018-11-04 18:37:24	IN_STORE	0.0	0.0	Williamsburg, 45 S 3rd St, Brooklyn, NY 11249	Spicy Meatballs Marketbowl	Farro with Summer Vegetables	Jasper Hill Mac & Cheese	Jasper Hill Mac & Cheese

View the first 5 rows in the file

A csv file can be read directly inside a zip file, without unzipping!

Loading a file and skipping rows

Number of rows to skip



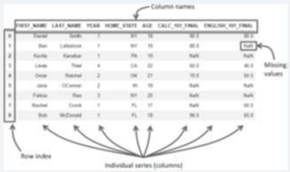
```
pd.read_csv('BA orders.zip', skiprows=2).head()
```

	O1011112	2018-05-31 11:35:00	IN_STORE	0.0	0.0.1	Bryant Park; 70 W 40th St, New York, NY 10018	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9
0	O752854	2018-04-21 18:12:57	DELIVERY	0.0	2.0	Columbia; 2884 Broadway, New York, NY 10025	Charred Chicken Marketbowl	Farro with Summer Vegetables	Snap Peas	Green Goddess Beans with Sesame
1	O2076864	2018-11-17 12:50:52	PICKUP	1.0	0.0	Flatiron; 40 W 25th St, New York, NY 10010	Charred Chicken Marketbowl	Classic Brown Rice	Jasper Hill Mac & Cheese	Cashew Kale Caesar
2	O1988898	2018-11-04 18:37:24	IN_STORE	0.0	0.0	Williamsburg; 45 S 3rd St, Brooklyn, NY 11249	Spicy Meatballs Marketbowl	Farro with Summer Vegetables	Jasper Hill Mac & Cheese	Jasper Hill Mac & Cheese
3	O1026484	2018-06-02 14:32:53	DELIVERY	1.0	0.0	Williamsburg; 45 S 3rd St, Brooklyn, NY 11249	NaN	NaN	NaN	NaN
4	O1878199	2018-10-19 19:55:09	PICKUP	0.0	0.0	Williamsburg; 45 S 3rd St, Brooklyn, NY 11249	Herb Roasted Chicken Marketbowl	Farm Greens with Mint	Roasted Sweet Potatoes	Cashew Kale Caesar

Pandas elements

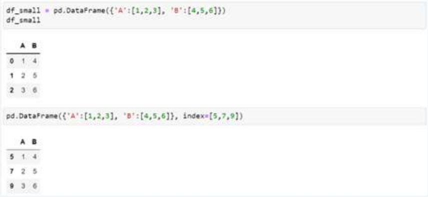
Pandas makes two new data types available to us

- The `Series`, which can store columns; each row has an index (think of this as a "row name")
- The `DataFrame`, which collects multiple series (columns) together with their titles to give us a full table



Creating DataFrames from dictionaries

As well as reading DataFrames from files, we can also create them from dictionaries. In so doing, we can specify the index we want to use, which doesn't have to be the row number!



```
df_small = pd.DataFrame({'A':[1,2,3], 'B':[4,5,6]})
df_small
```

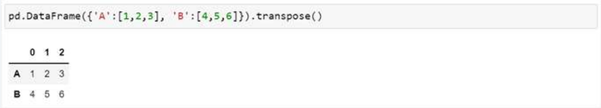
	A	B
0	1	4
1	2	5
2	3	6

```
pd.DataFrame({'A':[1,2,3], 'B':[4,5,6]}, index=[5,7,9])
```

	A	B
5	1	4
7	2	5
9	3	6

Transposing DataFrames

Transposing a DataFrame swaps the rows and the columns



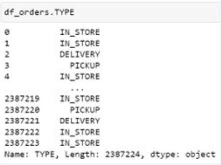
```
pd.DataFrame({'A':[1,2,3], 'B':[4,5,6]}).transpose()
```

	0	1	2
A	1	2	3
B	4	5	6

Accessing and modifying data in a Pandas DataFrame

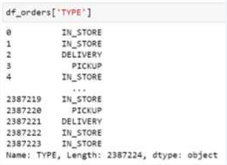
Accessing columns as series

There are two ways to access a column in a Pandas DataFrame



```
df_orders.TYPE
```

```
0          IN_STORE
1          IN_STORE
2          DELIVERY
3            PICKUP
4          IN_STORE
             ...
2387219    IN_STORE
2387220      PICKUP
2387221    DELIVERY
2387222    IN_STORE
2387223    IN_STORE
Name: TYPE, Length: 2387224, dtype: object
```



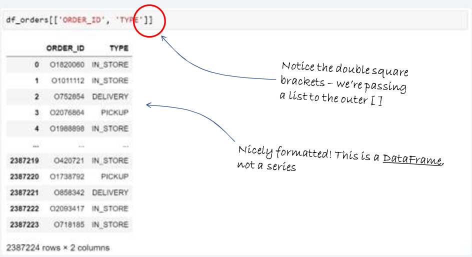
```
df_orders['TYPE']
```

```
0          IN_STORE
1          IN_STORE
2          DELIVERY
3            PICKUP
4          IN_STORE
             ...
2387219    IN_STORE
2387220      PICKUP
2387221    DELIVERY
2387222    IN_STORE
2387223    IN_STORE
Name: TYPE, Length: 2387224, dtype: object
```

Only works if the column name doesn't have spaces, doesn't start with a number, etc...

Notice the output isn't formatted – this is a tell-tale sign it's a series, not a DataFrame

Accessing a subset of columns as a DataFrame



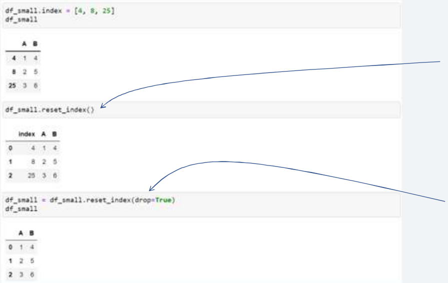
```
df_orders[['ORDER_ID', 'TYPE']]
```

```
         ORDER_ID      TYPE
      0  O1820060  IN_STORE
      1  O1011112  IN_STORE
      2   O752854  DELIVERY
      3  O2076864    PICKUP
      4  O1988898  IN_STORE
    ...       ...       ...
2387219   O420721  IN_STORE
2387220  O1738792    PICKUP
2387221   O858342  DELIVERY
2387222  O2093417  IN_STORE
2387223   O718185  IN_STORE

2387224 rows × 2 columns
```

Notice the double square brackets – we're passing a list to the outer [ ]

Nicely formatted! This is a DataFrame, not a series

Changing and resetting the index



```
df_small.index = [4, 8, 25]
df_small
```

```
    A  B
 4  1  4
 8  2  5
25  3  6
```

```
df_small.reset_index()
```

```
   index  A  B
0      4  1  4
1      8  2  5
2     25  3  6
```

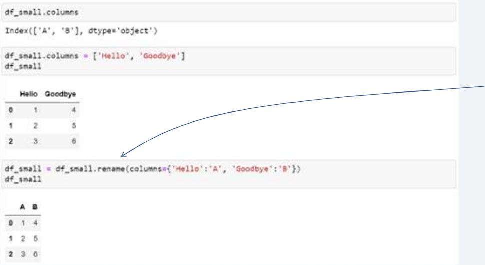
```
df_small = df_small.reset_index(drop=True)
df_small
```

```
   A  B
0  1  4
1  2  5
2  3  6
```

Notice that reset_index doesn't change the DataFrame; it simply returns a modified DataFrame

If you don't include this, the old index is kept as a new column

Changing column names – two ways



```
df_small.columns
```

```
Index(['A', 'B'], dtype='object')
```

```
df_small.columns = ['Hello', 'Goodbye']
df_small
```

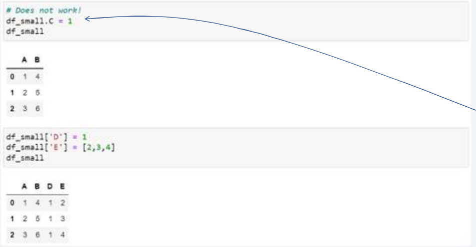
```
   Hello  Goodbye
0      1        4
1      2        5
2      3        6
```

```
df_small = df_small.rename(columns={'Hello':'A', 'Goodbye':'B'})
df_small
```

```
   A  B
0  1  4
1  2  5
2  3  6
```

Notice that rename doesn't change the DataFrame; it simply returns a modified DataFrame

Adding a column



```
# Does not work!
df_small.C = 1
df_small
```

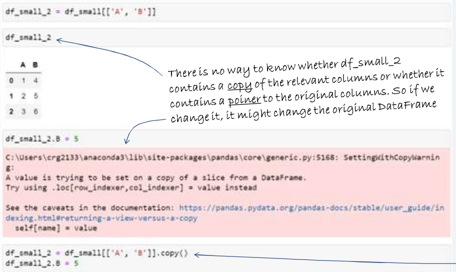
```
   A  B
0  1  4
1  2  5
2  3  6
```

```
df_small['D'] = 1
df_small['E'] = [2,3,4]
df_small
```

```
   A  B  D  E
0  1  4  1  2
1  2  5  1  3
2  3  6  1  4
```

The dot technique does not work! You must use the square bracket technique!

A warning: when to use `.copy()`



```
df_small_2 = df_small[['A', 'B']]
```

```
df_small_2
```

```
   A  B
0  1  4
1  2  5
2  3  6
```

```
df_small_2.B = 5
```

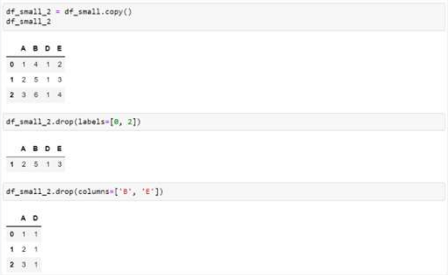
```
C:\Users\crg2133\anaconda3\lib\site-packages\pandas\core\generic.py:5168: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy
  self[name] = value
```

```
df_small_2 = df_small[['A', 'B']].copy()
df_small_2.B = 5
```

There is no way to know whether df_small_2 contains a copy of the relevant columns or whether it contains a poiner to the original columns. So if we change it, it might change the original DataFrame

Tell pandas we want a copy specifically

Deleting rows and columns



```
df_small_2 = df_small.copy()
df_small_2
```

```
   A  B  D  E
0  1  4  1  2
1  2  5  1  3
2  3  6  1  4
```

```
df_small_2.drop(labels=[0, 2])
```

```
   A  B  D  E
1  2  5  1  3
```

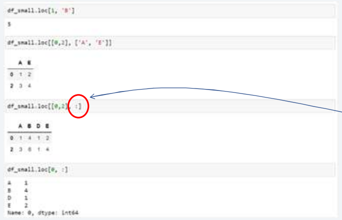
```
df_small_2.drop(columns=['B', 'E'])
```

```
   A  D
0  1  1
1  2  1
2  3  1
```

.loc

	A	B	D	E
0	1	4	1	2
1	2	5	1	3
2	3	6	1	4

`.loc` allows you to access specific parts of a DataFrame using the column names and the index



```
df_small.loc[1, 'B']
```

```
5
```

```
df_small.loc[[0,2], ['A', 'E']]
```

	A	E
0	1	2
2	3	4

```
df_small.loc[[0,2], :]
```

	A	B	D	E
0	1	4	1	2
2	3	6	1	4

```
df_small.loc[0, :]
```

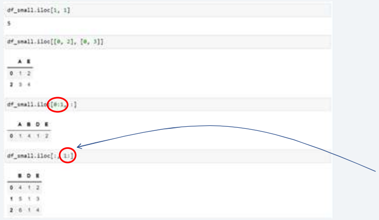
```
A    1
B    4
D    1
E    2
Name: 0, dtype: int64
```

This means "include everything"

.iloc

	A	B	D	E
0	1	4	1	2
1	2	5	1	3
2	3	6	1	4

`.loc` allows you to access specific parts of a DataFrame using the column and row **numbers**



```
df_small.iloc[1, 1]
```

```
5
```

```
df_small.iloc[[0, 2], [0, 3]]
```

	A	E
0	1	2
2	3	4

```
df_small.iloc[0:1, :]
```

	A	B	D	E
0	1	4	1	2

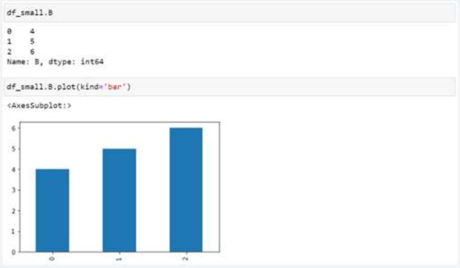
```
df_small.iloc[:, 1:]
```

	B	D	E
0	4	1	2
1	5	1	3
2	6	1	4

a: goes from a all the way till the end

Plotting directly from Pandas

Plotting a series



```
df_small.B
```

```
0    4
1    5
2    6
Name: B, dtype: int64
```

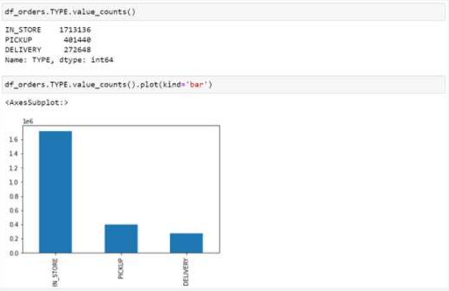
```
df_small.B.plot(kind='bar')
```

```
<AxesSubplot:>
```

The index of the series is used as the x-axis

Operations on Columns

Exploring discrete columns



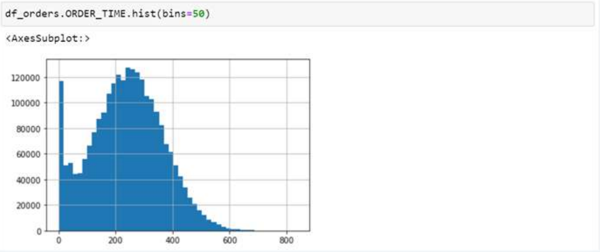
```
df_orders.TYPE.value_counts()
```

```
IN_STORE    1713136
PICKUP       401440
DELIVERY     272648
Name: TYPE, dtype: int64
```

```
df_orders.TYPE.value_counts().plot(kind='bar')
```

```
<AxesSubplot:>
```

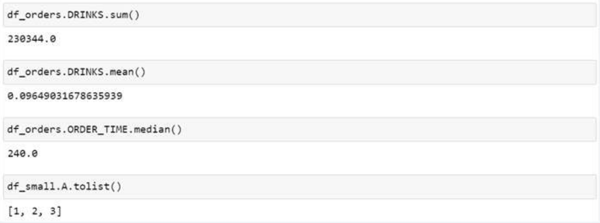
Exploring continuous columns



```
df_orders.ORDER_TIME.hist(bins=50)
```

```
<AxesSubplot:>
```

Aggregations



```
df_orders.DRINKS.sum()
```

```
230344.0
```

```
df_orders.DRINKS.mean()
```

```
0.09649031678635939
```

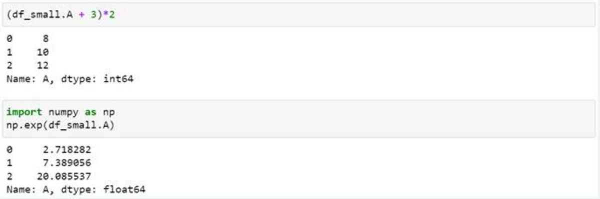
```
df_orders.ORDER_TIME.median()
```

```
240.0
```

```
df_small.A.tolist()
```

```
[1, 2, 3]
```

Arithmetic



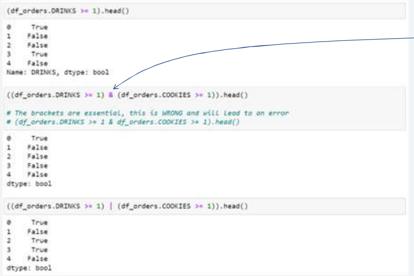
```
(df_small.A + 3)*2
```

```
0     8
1    10
2    12
Name: A, dtype: int64
```

```
import numpy as np
np.exp(df_small.A)
```

```
0     2.718282
1     7.389056
2    20.085537
Name: A, dtype: float64
```

Logic



```
(df_orders.DRINKS >= 1).head()
```

```
0     True
1    False
2    False
3     True
4    False
Name: DRINKS, dtype: bool
```

```
((df_orders.DRINKS >= 1) & (df_orders.COOKIES >= 1)).head()

# The brackets are essential, this is WRONG and will lead to an error
# (df_orders.DRINKS >= 1 & df_orders.COOKIES >= 1).head()
```

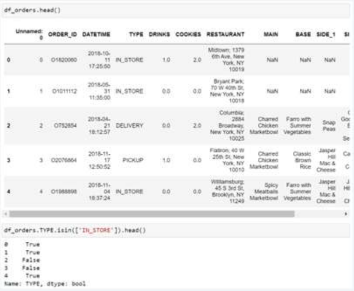
```
0     True
1    False
2    False
3    False
4    False
dtype: bool
```

```
((df_orders.DRINKS >= 1) | (df_orders.COOKIES >= 1)).head()
```

```
0     True
1    False
2     True
3     True
4    False
dtype: bool
```

Notice "&" and not "and"

isin



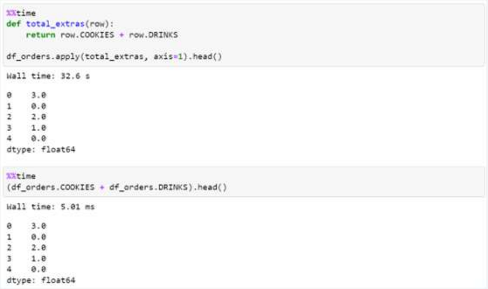
```
df_orders.head()
```

	Unnamed: 0	ORDER_ID	DATETIME	TYPE	DRINKS	COOKIES	RESTAURANT	MAIN	BASE	SIDE_1	SI
0	0	O1820060	2018-10-11 17:25:50	IN_STORE	1.0	2.0	Midtown; 1379 6th Ave, New York, NY 10019	NaN	NaN	NaN	
1	1	O1011112	2018-05-31 11:35:00	IN_STORE	0.0	0.0	Bryant Park; 70 W 40th St, New York, NY 10018	NaN	NaN	NaN	
2	2	O752854	2018-04-21 18:12:57	DELIVERY	0.0	2.0	Columbia; 2884 Broadway, New York, NY 10025	Charred Chicken Marketbowl	Farro with Summer Vegetables	Snap Peas	(Go(E Se
3	3	O2076864	2018-11-17 12:50:52	PICKUP	1.0	0.0	Flatiron; 40 W 25th St, New York, NY 10010	Charred Chicken Marketbowl	Classic Brown Rice	Jasper Hill Mac & Cheese	Ca C
4	4	O1988898	2018-11-04 18:37:24	IN_STORE	0.0	0.0	Williamsburg; 45 S 3rd St, Brooklyn, NY 11249	Spicy Meatballs Marketbowl	Farro with Summer Vegetables	Jasper Hill Mac & Cheese	J Hi Cl

```
df_orders.TYPE.isin(['IN_STORE']).head()
```

```
0     True
1     True
2    False
3    False
4     True
Name: TYPE, dtype: bool
```

apply



```
%%time
def total_extras(row):
    return row.COOKIES + row.DRINKS

df_orders.apply(total_extras, axis=1).head()
```

```
Wall time: 32.6 s

0    3.0
1    0.0
2    2.0
3    1.0
4    0.0
dtype: float64
```

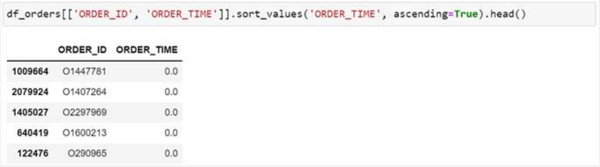
```
%%time
(df_orders.COOKIES + df_orders.DRINKS).head()
```

```
Wall time: 5.01 ms

0    3.0
1    0.0
2    2.0
3    1.0
4    0.0
dtype: float64
```

Columbia Business School
AT THE VERY CENTER OF BUSINESS

Filtering DataFrames

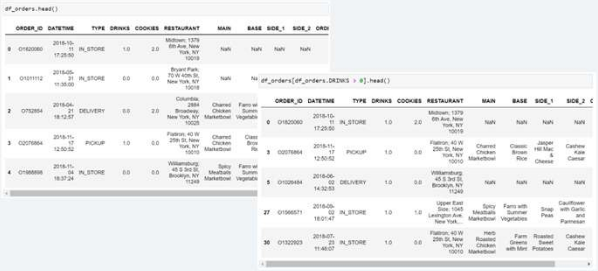
Sorting DataFrames



```
df_orders[['ORDER_ID', 'ORDER_TIME']].sort_values('ORDER_TIME', ascending=True).head()
```

	ORDER_ID	ORDER_TIME
1009664	O1447781	0.0
2079924	O1407264	0.0
1405027	O2297969	0.0
640419	O1600213	0.0
122476	O290965	0.0

Filtering DataFrames



```
df_orders.head()
```

	ORDER_ID	DATETIME	TYPE	DRINKS	COOKIES	RESTAURANT	MAIN	BASE	SIDE_1	SIDE_2	ORDI
0	O1820060	2018-10-11 17:25:50	IN_STORE	1.0	2.0	Midtown; 1379 6th Ave, New York, NY 10019	NaN	NaN	NaN	NaN	
1	O1011112	2018-05-31 11:35:00	IN_STORE	0.0	0.0	Bryant Park; 70 W 40th St, New York, NY 10018	NaN	Na			
2	O752854	2018-04-21 18:12:57	DELIVERY	0.0	2.0	Columbia; 2884 Broadway, New York, NY 10025	Charred Chicken Marketbowl	Farro wi Summ Vegetabl			
3	O2076864	2018-11-17 12:50:52	PICKUP	1.0	0.0	Flatiron; 40 W 25th St, New York, NY 10010	Charred Chicken Marketbowl	Class Brow Ric			
4	O1988898	2018-11-04 18:37:24	IN_STORE	0.0	0.0	Williamsburg; 45 S 3rd St, Brooklyn, NY 11249	Spicy Meatballs Marketbowl	Farro wi Summ Vegetabl			

```
df_orders[df_orders.DRINKS > 0].head()
```

	ORDER_ID	DATETIME	TYPE	DRINKS	COOKIES	RESTAURANT	MAIN	BASE	SIDE_1	SIDE_2	C
0	O1820060	2018-10-11 17:25:50	IN_STORE	1.0	2.0	Midtown; 1379 6th Ave, New York, NY 10019	NaN	NaN	NaN	NaN	
3	O2076864	2018-11-17 12:50:52	PICKUP	1.0	0.0	Flatiron; 40 W 25th St, New York, NY 10010	Charred Chicken Marketbowl	Classic Brown Rice	Jasper Hill Mac & Cheese	Cashew Kale Caesar	
5	O1026484	2018-06-02 14:32:53	DELIVERY	1.0	0.0	Williamsburg; 45 S 3rd St, Brooklyn, NY 11249	NaN	NaN	NaN	NaN	
27	O1566571	2018-09-02 18:01:47	IN_STORE	1.0	1.0	Upper East Side; 1045 Lexington Ave, New York,...	Spicy Meatballs Marketbowl	Farro with Summer Vegetables	Snap Peas	Cauliflower with Garlic and Parmesan	
30	O1322923	2018-07-23 11:46:07	IN_STORE	1.0	0.0	Flatiron; 40 W 25th St, New York, NY 10010	Herb Roasted Chicken Marketbowl	Farm Greens with Mint	Roasted Sweet Potatoes	Cashew Kale Caesar	

Reviewing square brackets

`df_orders[ ]`

What's in the [ ]	What happens
A string	A series is returned containing the column with the name in the string
A list	A DataFrame is returned, containing the subset of columns named in the list
A series of True/False values	A DataFrame is returned, containing

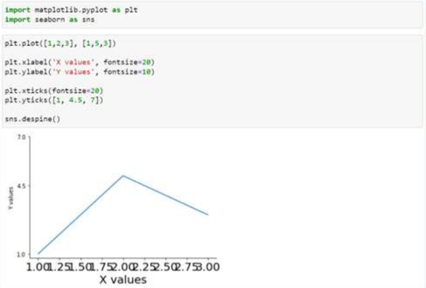
Columbia Business School
AT THE VERY CENTER OF BUSINESS

Plotting in `matplotlib`

`matplotlib`

- `matplotlib` is Python's most popular plotting library
- It was designed to emulate Matlab's plotting capability
- A sometimes less well-known fact is that there are **two** very different **ways** to use the library
  - **The state based/pyplot interface**, which is great for creating quick-and-easy plots, but gives you much less control over the finer aspects of the plot
  - **The object oriented interface**, which gives far finer control over every aspect of the plot

The pyplot interface



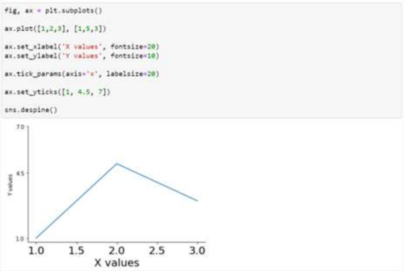
```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
plt.plot([1,2,3], [1,5,3])

plt.xlabel('X values', fontsize=20)
plt.ylabel('Y values', fontsize=10)

plt.xticks(fontsize=20)
plt.yticks([1, 4.5, 7])

sns.despine()
```

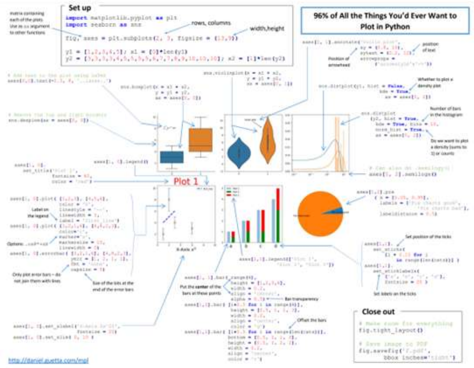
The object-oriented interface

- Every Python plot comprises a **figure**, on which one or more **axes** are plotted. Various **artist** elements (lines, labels, etc…) are then plotted on top of that axis
- The object-oriented interface creates these elements manually, and allows you to manipulate them one by one
- It also allows you to create a figure with multiple axes; there are two reasons you might want to do this
  - Include a “secondary axis” with a different scale
  - Ceate multiple plots in one figure

The object-oriented interface



```
fig, ax = plt.subplots()

ax.plot([1,2,3], [1,5,3])

ax.set_xlabel('X values', fontsize=20)
ax.set_ylabel('Y values', fontsize=10)

ax.tick_params(axis='x', labelsize=20)

ax.set_yticks([1, 4.5, 7])

sns.despine()
```



96% of All the Things You'd Ever Want to Plot in Python

Set up

```
import matplotlib.pyplot as plt
import seaborn as sns

fig, axes = plt.subplots(2, 3, figsize = (12,9))

y1 = [1,2,3,4,5]; x1 = [0]*len(y1)
y2 = [illegible]; x2 = [1]*len(y2)
```

Close out

```
# Make room for everything
fig.tight_layout()

# Save image to PDF
fig.savefig('F.pdf',
        bbox_inches='tight')
```

matrix containing each of the plots. Use as ax argument to other functions
rows, columns
width,height
Position of arrowhead
position of text
Whether to plot a density plot
Number of bars in the histogram
Do we want to plot a density (sums to 1) or counts
Plot 1
Label on the legend
Options: ...
Only plot error bars – do not join them with lines
Size of the ticks at the end of the error bars
Put the center of the bars at these points
Bar transparency
Offset the bars
Set position of the ticks
Set labels on the ticks
X-Axis x^2

http://daniel.guetta.com/mpl

Fall 2024

Linear Regression

Module 3

Professor Daniel Guetta

This Module

- Simple linear regression
- Multiple linear regression
- The R^2
- Dummy variables
- Variable selection
- Making predictions
- Interpreting regression output
- Advanced regression: nonlinearities, interactions, penalties…

Regression analysis: the big picture

Regression analysis: the big picture

- Regression is used to describe the relationship between two or more variables
- There are two main purposes of a regression
  - Quantifying causality (**explain**)
    - What is the effect of smoking on the likelihood of cardiovascular disease?
    - Do mask mandates reduce COVID transmission rates?
  - Prediction and forecasting (**predict**)
    - Predict home sales for December given an interest rate
    - Predict the price of wine given its acidity

Example 1: the wine equation

https://www.nytimes.com/1990/03/04/us/wine-equation-puts-some-noses-out-of-joint.html

Calculate the winter rain and the harvest rain (in millimeters). Add summer heat in the vineyard (in degrees centigrade). Subtract 12.145. And what do you have? A very, very passionate argument over wine.

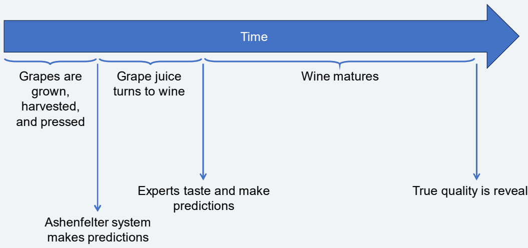
Prof. Orley Ashenfelter, a Princeton economist, has devised a mathematical formula for predicting the quality of red wine vintages in France. And the guardians of tradition are fuming.

Robert M. Parker Jr., generally regarded as the most influential wine critic in America, calls Professor Ashenfelter's research "ludicrous and absurd.‘’

NYT, March 4th 1990

$$\boxed{\log(\text{Quality}) = -12.145 + \beta_{\text{wr}}(\text{WinterRain}) + \beta_{\text{hr}}(\text{HarvestRain}) + \beta_{\text{sh}}(\text{SummerHeat})}$$

Why do we even need a prediction?



Time

Grapes are grown, harvested, and pressed

Grape juice turns to wine

Wine matures

Ashenfelter system makes predictions

Experts taste and make predictions

True quality is revealed

Example 1: the wine equation

Mr. Parker rates the 1986's as "very good and sometimes exceptional." Peter A. Sichel, author of the influential Bordeaux Vintage and Market Report, said the 1986's have "elegance and classic Bordeaux structure." New York stores, brimming with the vintage, are pricing the wines in the same range as the much-praised 1985's.

But according to the Ashenfelter system, below-average growing season temperatures and above-average harvest rainfall doom the 1986 Bordeaux to mediocrity. When the dust settles, he predicts, it will be judged the worst vintage of the 1980's, and no better than the unmemorable 1974's or 1969's.

Perhaps the most dramatic Ashenfelter prediction, the one likely to vault the ratings system into prominence or doom it to obscurity, is for the 1989 vintage.

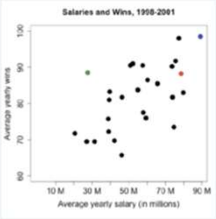
These wines are barely three months in the cask and have yet to be tasted by critics. By Professor Ashenfelter's calculations, the hottest growing season in memory, combined with a very dry harvest, all but guarantee that the 1989 Bordeaux will be stunningly good. Adjusted for age, he predicts, these wines will eventually sell for a substantial premium over the great 1961 vintage.

What part of this example quantifies causality? What part does prediction and forecasting?

Example 1: the verdict

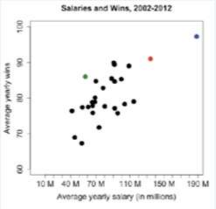
- "1986 was largely OK, but stopped short of excellent."
- "1989 was a fantastic vintage year. Bordeaux, particularly, had virtually no faults with red, whites, and dessert wines all performing exceptionally well."

Example 2: the baseball equation



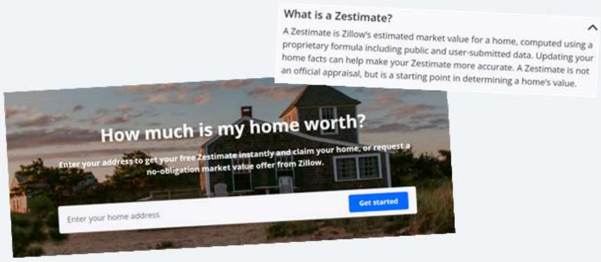
$$\boxed{\text{RunScored} = \beta_0 + \beta_{\text{obp}}(\text{OnBasePer}) + \beta_{\text{slp}}(\text{SluggingPer}) + \beta_{\text{ba}}(\text{BattingAvg}) + \cdots}$$

Example 2: once everyone catches on…



$$\boxed{\text{RunScored} = \beta_0 + \beta_{\text{obp}}(\text{OnBasePer}) + \beta_{\text{slp}}(\text{SluggingPer}) + \beta_{\text{ba}}(\text{BattingAvg}) + \cdots}$$

Example 3: the zestimate



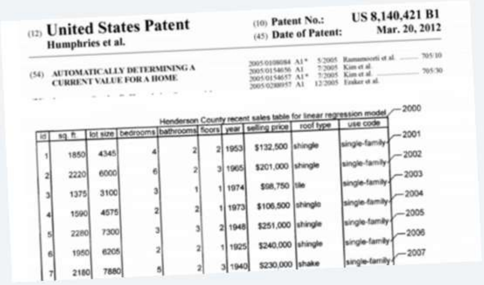
What is a Zestimate?

A Zestimate is Zillow's estimated market value for a home, computed using a proprietary formula including public and user-submitted data. Updating your home facts can help make your Zestimate more accurate. A Zestimate is not an official appraisal, but is a starting point in determining a home's value.

How much is my home worth?

Enter your address to get your free Zestimate instantly and claim your home, or request a no-obligation market value offer from Zillow.

Example 3: the zestimate



(12) **United States Patent**
Humphries et al.

(10) **Patent No.:** US 8,140,421 B1
(45) **Date of Patent:** Mar. 20, 2012

(54) AUTOMATICALLY DETERMINING A CURRENT VALUE FOR A HOME

2005/0108084 A1* 5/2005 Ramamoorti et al. 705/10
2005/0154656 A1 7/2005 Kim et al.
2005/0154657 A1* 7/2005 Kim et al. 705/30
2005/0288957 A1 12/2005 Eraker et al.

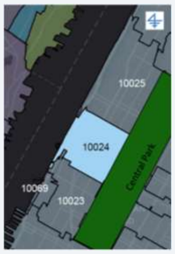
Henderson County recent sales table for linear regression model —2000

id	sq. ft.	lot size	bedrooms	bathrooms	floors	year	selling price	roof type	use code
1	1850	4345	4	2	2	1953	$132,500	shingle	single-family —2001
2	2220	6000	6	2	3	1965	$201,000	shingle	single-family —2002
3	1375	3100	3	1	1	1974	$98,750	tile	single-family —2003
4	1590	4575	2	2	1	1973	$106,500	shingle	single-family —2004
5	2280	7300	3	3	2	1948	$251,000	shingle	single-family —2005
6	1950	6205	2	2	1	1925	$240,000	shingle	single-family —2006
7	2180	7880	5	2	3	1940	$230,000	shake	single-family —2007

Columbia Business School
AT THE VERY CENTER OF BUSINESS

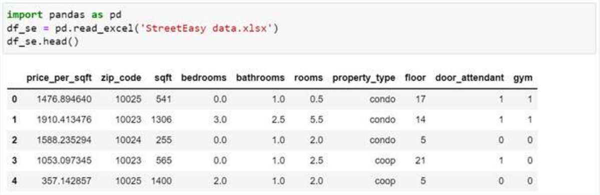
The CBSTimate

The CBSTimate



- We will create a mini version of the Zestimate
- We'll be using data from the UWS – specifically, the following four zip codes
- Our data comprises 1,464 apartments, the price per square foot they brought in when sold, and several apartment characteristics we'll discuss shortly

Loading the StreetEasy data

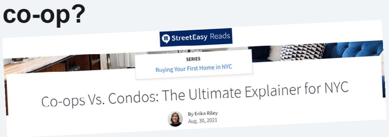


```
import pandas as pd
df_se = pd.read_excel('StreetEasy data.xlsx')
df_se.head()
```

	price_per_sqft	zip_code	sqft	bedrooms	bathrooms	rooms	property_type	floor	door_attendant	gym
0	1476.894640	10025	541	0.0	1.0	0.5	condo	17	1	1
1	1910.413476	10023	1306	3.0	2.5	5.5	condo	14	1	1
2	1588.235294	10024	255	0.0	1.0	2.0	condo	5	0	0
3	1053.097345	10023	565	0.0	1.0	2.5	coop	21	1	0
4	357.142857	10025	1400	2.0	1.0	2.0	coop	5	0	0

Real estate agent definitions, lol

Condo or co-op?



StreetEasy Reads

SERIES
Buying Your First Home in NYC

Co-ops Vs. Condos: The Ultimate Explainer for NYC

By Erika Riley
Aug. 30, 2021

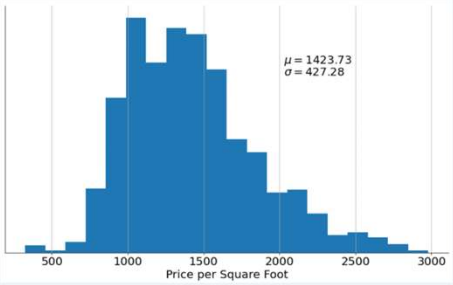
Condos

- Traditional real estate investment (own the apartment)
- Fewer restrictions (on renting for eg)
- Often newer
- Often more amenities

Co-ops

- Owns a share in the building
- Sale and rentals require board approval
- Often older
- Often fewer amenities
- Board can block rentals and purchases

Price per square foot



What “explain” and “predict” questions might we ask using this data?

Beginning with univariate regression

Important note

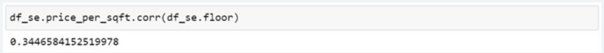
To focus on the insights behind linear regression, we are going to be a little sloppy with the distinction between random variables and specific deterministic values these random variables can take, and various other mathematical details; a more rigorous, mathematical class would make that distinction more carefully. Those interested can refer to any advanced text on linear regression, or my notes here.

Let’s begin with two simple questions:

1. Does floor affect price?
2. Given the floor, can I predict price?

How might we begin answering these questions?

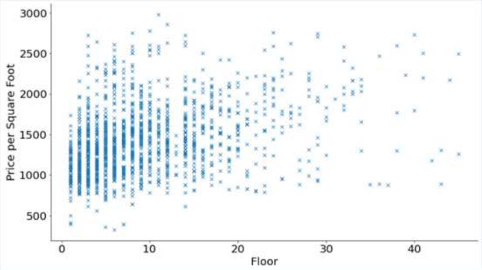
Correlation

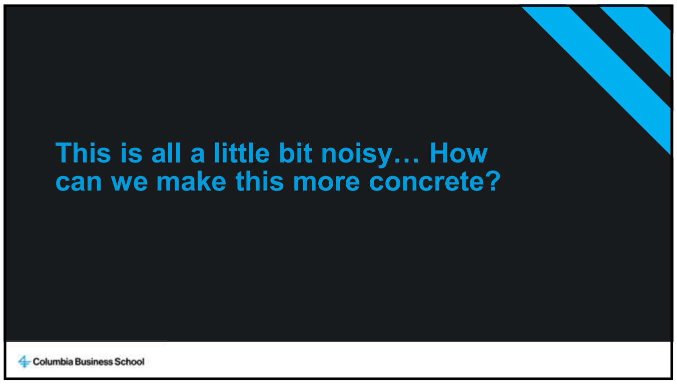


```
df_se.price_per_sqft.corr(df_se.floor)
```

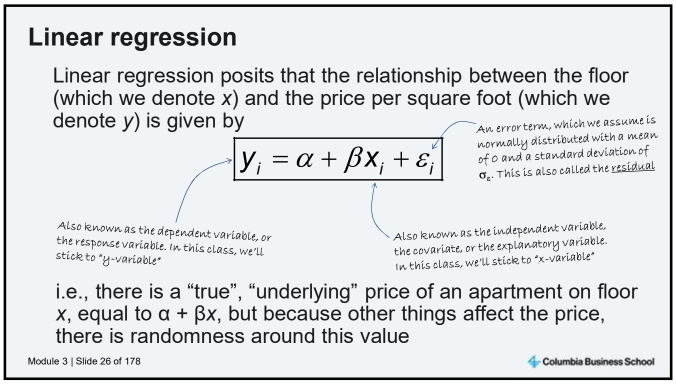
```
0.3446584152519978
```

Visualization





This is all a little bit noisy… How can we make this more concrete?



Linear regression

Linear regression posits that the relationship between the floor (which we denote *x*) and the price per square foot (which we denote *y*) is given by

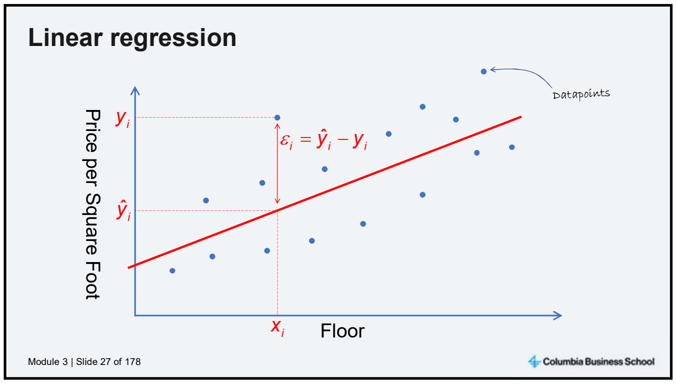
$$\boxed{y_i = \alpha + \beta x_i + \varepsilon_i}$$

An error term, which we assume is normally distributed with a mean of 0 and a standard deviation of σ_ε. This is also called the residual

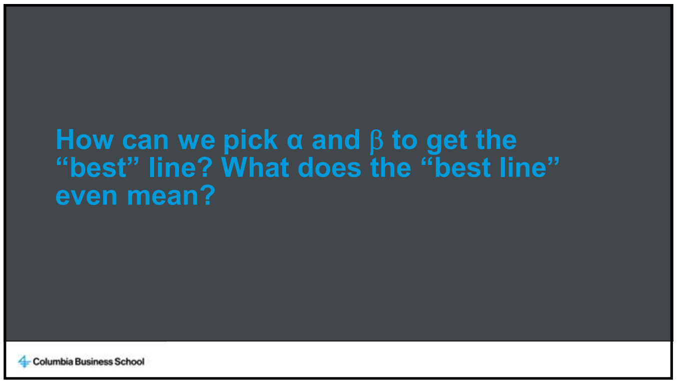
Also known as the dependent variable, or the response variable. In this class, we'll stick to "y-variable"

Also known as the independent variable, the covariate, or the explanatory variable. In this class, we'll stick to "x-variable"

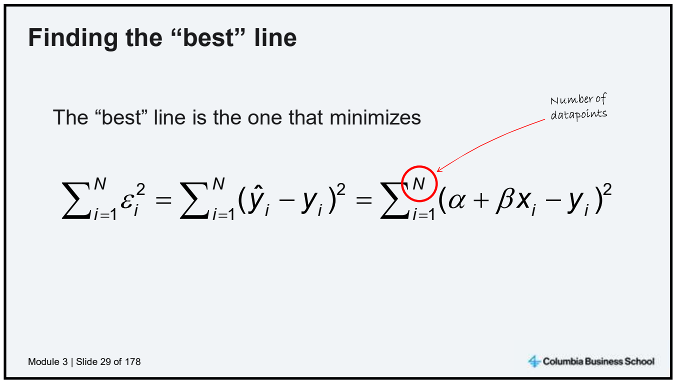
i.e., there is a "true", "underlying" price of an apartment on floor *x*, equal to α + β*x*, but because other things affect the price, there is randomness around this value



Linear regression



How can we pick α and β to get the "best" line? What does the "best line" even mean?

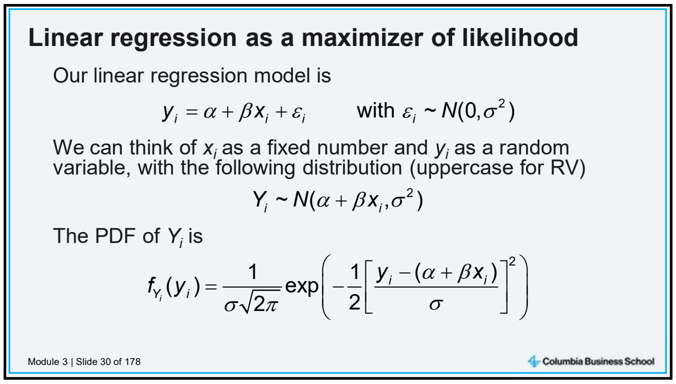


Finding the "best" line

The "best" line is the one that minimizes

Number of datapoints

$$\sum_{i=1}^{N} \varepsilon_i^2 = \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 = \sum_{i=1}^{N} (\alpha + \beta x_i - y_i)^2$$



Linear regression as a maximizer of likelihood

Our linear regression model is

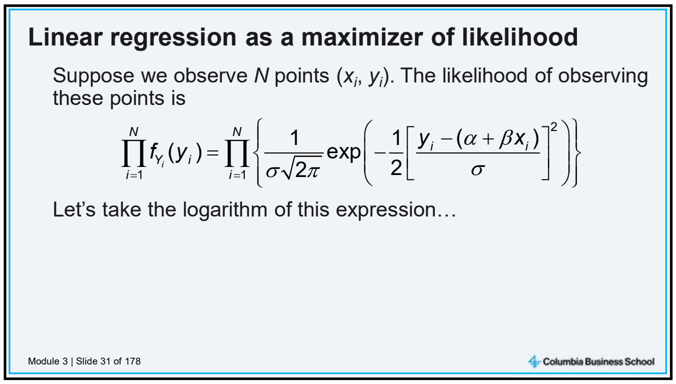
$$y_i = \alpha + \beta x_i + \varepsilon_i \qquad \text{with } \varepsilon_i \sim N(0, \sigma^2)$$

We can think of x_i as a fixed number and y_i as a random variable, with the following distribution (uppercase for RV)

$$Y_i \sim N(\alpha + \beta x_i, \sigma^2)$$

The PDF of Y_i is

$$f_{Y_i}(y_i) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left[\frac{y_i - (\alpha + \beta x_i)}{\sigma}\right]^2\right)$$

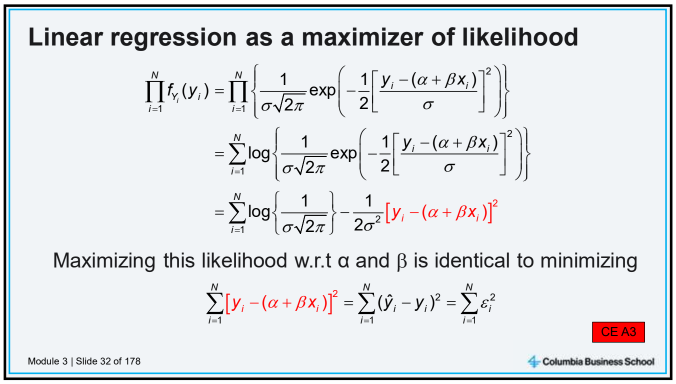


Linear regression as a maximizer of likelihood

Suppose we observe N points (x_i, y_i). The likelihood of observing these points is

$$\prod_{i=1}^{N} f_{Y_i}(y_i) = \prod_{i=1}^{N}\left\{\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left[\frac{y_i-(\alpha+\beta x_i)}{\sigma}\right]^2\right)\right\}$$

Let's take the logarithm of this expression…



Linear regression as a maximizer of likelihood

$$\prod_{i=1}^{N} f_{Y_i}(y_i) = \prod_{i=1}^{N}\left\{\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left[\frac{y_i-(\alpha+\beta x_i)}{\sigma}\right]^2\right)\right\}$$

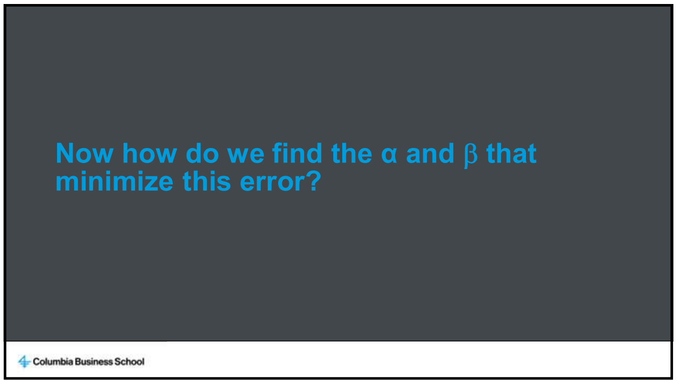
$$= \sum_{i=1}^{N}\log\left\{\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left[\frac{y_i-(\alpha+\beta x_i)}{\sigma}\right]^2\right)\right\}$$

$$= \sum_{i=1}^{N}\log\left\{\frac{1}{\sigma\sqrt{2\pi}}\right\} - \frac{1}{2\sigma^2}[y_i-(\alpha+\beta x_i)]^2$$

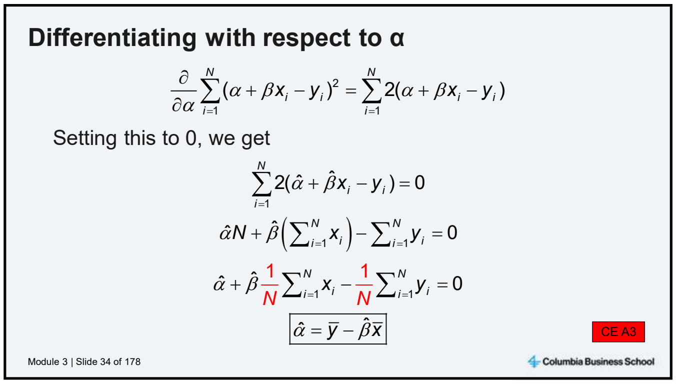
Maximizing this likelihood w.r.t α and β is identical to minimizing

$$\sum_{i=1}^{N}[y_i-(\alpha+\beta x_i)]^2 = \sum_{i=1}^{N}(\hat{y}_i-y_i)^2 = \sum_{i=1}^{N}\varepsilon_i^2$$

CE A3



Now how do we find the α and β that minimize this error?



Differentiating with respect to α

$$\frac{\partial}{\partial\alpha}\sum_{i=1}^{N}(\alpha+\beta x_i-y_i)^2 = \sum_{i=1}^{N}2(\alpha+\beta x_i-y_i)$$

Setting this to 0, we get

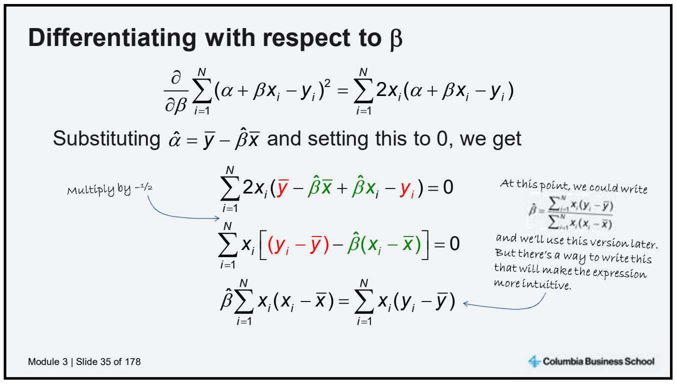
$$\sum_{i=1}^{N}2(\hat{\alpha}+\hat{\beta}x_i-y_i)=0$$

$$\hat{\alpha}N+\hat{\beta}\left(\sum_{i=1}^{N}x_i\right)-\sum_{i=1}^{N}y_i=0$$

$$\hat{\alpha}+\hat{\beta}\frac{1}{N}\sum_{i=1}^{N}x_i-\frac{1}{N}\sum_{i=1}^{N}y_i=0$$

$$\boxed{\hat{\alpha}=\bar{y}-\hat{\beta}\bar{x}}$$

CE A3



Differentiating with respect to β

$$\frac{\partial}{\partial\beta}\sum_{i=1}^{N}(\alpha+\beta x_i-y_i)^2 = \sum_{i=1}^{N}2x_i(\alpha+\beta x_i-y_i)$$

Substituting $\hat{\alpha}=\bar{y}-\hat{\beta}\bar{x}$ and setting this to 0, we get

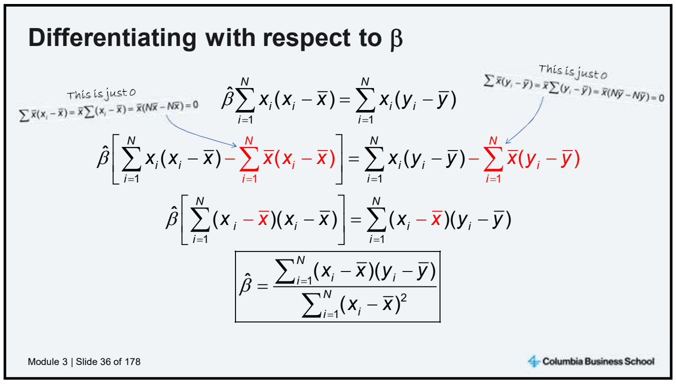
Multiply by -½

$$\sum_{i=1}^{N}2x_i(\bar{y}-\hat{\beta}\bar{x}+\hat{\beta}x_i-y_i)=0$$

$$\sum_{i=1}^{N}x_i\left[(y_i-\bar{y})-\hat{\beta}(x_i-\bar{x})\right]=0$$

$$\hat{\beta}\sum_{i=1}^{N}x_i(x_i-\bar{x})=\sum_{i=1}^{N}x_i(y_i-\bar{y})$$

At this point, we could write $\hat{\beta}=\frac{\sum_{i=1}^{N}x_i(y_i-\bar{y})}{\sum_{i=1}^{N}x_i(x_i-\bar{x})}$ and we'll use this version later. But there's a way to write this that will make the expression more intuitive.



Differentiating with respect to β

This is just 0 $\sum\bar{x}(x_i-\bar{x})=\bar{x}\sum(x_i-\bar{x})=\bar{x}(N\bar{x}-N\bar{x})=0$

This is just 0 $\sum\bar{x}(y_i-\bar{y})=\bar{x}\sum(y_i-\bar{y})=\bar{x}(N\bar{y}-N\bar{y})=0$

$$\hat{\beta}\sum_{i=1}^{N}x_i(x_i-\bar{x})=\sum_{i=1}^{N}x_i(y_i-\bar{y})$$

$$\hat{\beta}\left[\sum_{i=1}^{N}x_i(x_i-\bar{x})-\sum_{i=1}^{N}\bar{x}(x_i-\bar{x})\right]=\sum_{i=1}^{N}x_i(y_i-\bar{y})-\sum_{i=1}^{N}\bar{x}(y_i-\bar{y})$$

$$\hat{\beta}\left[\sum_{i=1}^{N}(x_i-\bar{x})(x_i-\bar{x})\right]=\sum_{i=1}^{N}(x_i-\bar{x})(y_i-\bar{y})$$

$$\boxed{\hat{\beta}=\frac{\sum_{i=1}^{N}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{N}(x_i-\bar{x})^2}}$$

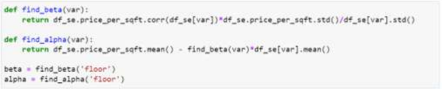
A more intuitive explanation for β

Note that we can write

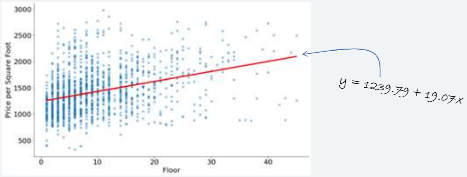
$$
\begin{aligned}
\hat{\beta} &= \frac{\sum_{i=1}^{N}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{N}(x_i-\bar{x})^2} \\
&= \frac{N\text{Cov}(X,Y)}{N\text{Std}(X)}\frac{1}{\text{Std}(X)}\cdot\frac{\text{Std}(Y)}{\text{Std}(Y)} \\
&= \frac{\text{Cov}(X,Y)}{\text{Std}(X)\text{Std}(Y)}\frac{\text{Std}(Y)}{\text{Std}(X)} \\
&= \text{Corr}(X,Y)\frac{\text{Std}(Y)}{\text{Std}(X)}
\end{aligned}
$$

In other words, the gradient is just the correlation, corrected for the variance of each column!

Univariate regression in Python



```python
def find_beta(var):
    return df_se.price_per_sqft.corr(df_se[var])*df_se.price_per_sqft.std()/df_se[var].std()

def find_alpha(var):
    return df_se.price_per_sqft.mean() - find_beta(var)*df_se[var].mean()

beta = find_beta('floor')
alpha = find_alpha('floor')
```



y = 1239.79 + 19.07x

How can this be used for “predict” and “explain” purposes?

The “predict” vs. “explain”

Explain

Price per square foot =
1239.79 + 19.07 × Floor

An apartment on “floor 0” costs $1239.79 per square foot

Every extra floor leads to an extra $19.07 per square foot

Predict

Price per square foot =
1239.79 + 19.07 × Floor

Given the floor of an apartment, I can predict the price per square foot for that apartment

How does this relate to the concept of population parameter/sample statistic from our first lecture?

Population parameters and statistics

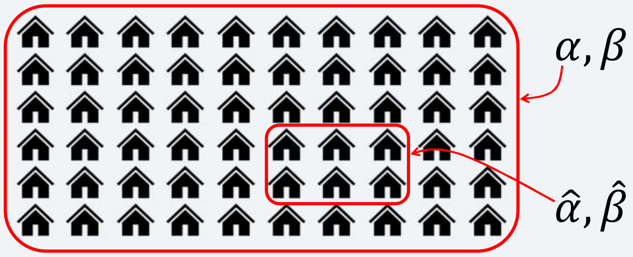
α, β

$\hat{\alpha}, \hat{\beta}$

These are the **population parameters** – the **true** impact of floor on the price per square foot

These are the **statistics**, which are **random variables** – we derive these from our sample, which is random (why?) We will later find the **distribution** of these statistics

Population parameter vs. statistics



(Optional) Better understanding the regression coefficients

Understanding the coefficients – part 1

Feeding $\hat{\alpha}$ and this value of $\hat{\beta}$ into $\hat{y} = \hat{\alpha} + \hat{\beta}x$, we get

$$\hat{y} = \hat{\alpha} + \hat{\beta}x$$

$$\hat{y} = (\bar{y} - \hat{\beta}\bar{x}) + \hat{\beta}x$$

$$\hat{y} - \bar{y} = \hat{\beta}(x - \bar{x})$$

$$\hat{y} - \bar{y} = \text{Corr}(X,Y)\frac{\text{Std}(Y)}{\text{Std}(X)}(x - \bar{x})$$

$$\boxed{\frac{\hat{y} - \bar{y}}{\text{Std}(Y)} = \text{Corr}(X,Y)\frac{x - \bar{x}}{\text{Std}(X)}}$$

If the variables are standardized, the intercept is 0 and the gradient is just the correlation!

Understanding the coefficients – part 2

We saw that

$$\frac{\hat{y} - \bar{y}}{\text{Std}(Y)} = \text{Corr}(X,Y)\frac{x - \bar{x}}{\text{Std}(X)}$$

Suppose we have a datapoint with x just equal to the mean. For example, suppose an apartment is on the 9.6th floor (the average). Then

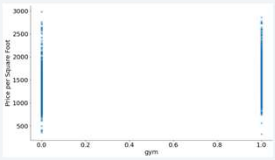
$$\frac{\hat{y} - \bar{y}}{\text{Std}(Y)} = 0$$

$$\hat{y} = \bar{y}$$

We just predict the average price per square feet. If the apartment is average, why predict anything else?!

Understanding the coefficients – part 3

To get an even deeper understanding of the slope and intercept, let’s consider an example in which x only takes two values (0 and 1). For example, a regression of `price_per_sqft` against `gym`



$$\begin{aligned}
N_0 &\equiv (\#\text{ points with } x = 0)\\
N_1 &\equiv (\#\text{ points with } x = 1)\\
\bar{y}_0 &\equiv \frac{1}{N_0}\left(\sum_{\text{points with } x_1=0} y_i\right)\\
\bar{y}_1 &\equiv \frac{1}{N_1}\left(\sum_{\text{points with } x_1=1} y_i\right)\\
\bar{y} &= \bar{y}_1 P(X=1) + \bar{y}_0 P(X=0)\\
&= \bar{y}_1\bar{x} + \bar{y}_0(1-\bar{x})
\end{aligned}$$

Understanding the coefficients – part 3

Let’s now go back to our original expression for β

$$\hat{\beta} = \frac{\sum_{i=1}^{N} x_i(y_i - \bar{y})}{\sum_{i=1}^{N} x_i(x_i - \bar{x})}$$

Now split it into points with x = 0 and x = 1

$$\hat{\beta} = \frac{\sum_{\text{points with } x_i=0} x_i(y_i - \bar{y}) + \sum_{\text{points with } x_i=1} x_i(y_i - \bar{y})}{\sum_{\text{points with } x_i=0} x_i(x_i - \bar{x}) + \sum_{\text{points with } x_i=1} x_i(x_i - \bar{x})}$$

$$\hat{\beta} = \frac{\sum_{\text{points with } x_i=1} y_i - \bar{y}}{\sum_{\text{points with } x_i=1} 1 - \bar{x}}$$

Understanding the coefficients – part 4

$$\hat{\beta} = \frac{\sum_{\text{points with } x_i=1} y_i - \bar{y}}{\sum_{\text{points with } x_i=1} 1-\bar{x}}$$

$$= \frac{N_1(\bar{y}_1 - \bar{y})}{N_1(1-\bar{x})}$$

$$= \frac{\bar{y}_1 - [\bar{y}_1\bar{x} + \bar{y}_0(1-\bar{x})]}{1-\bar{x}}$$

$$= \frac{(\bar{y}_1 - \bar{y}_0)(1-\bar{x})}{1-\bar{x}}$$

$$= \bar{y}_1 - \bar{y}_0$$

$$\hat{\alpha} = \bar{y} - \beta\bar{x}$$

$$= \bar{y}_1\bar{x} + \bar{y}_0(1-\bar{x}) - (\bar{y}_1 - \bar{y}_0)\bar{x}$$

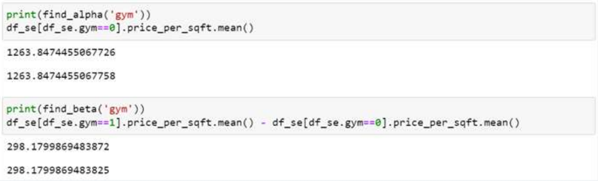
$$= \bar{y}_0$$

We showed earlier that $\bar{y} = \bar{y}_1\bar{x} + \bar{y}_0(1-\bar{x})$

This is the difference between the average price per sqft with and without a gym; in other words, the "gym premium"

This is the average price per sqft for apartments without a gym

Understanding the coefficients – part 3



```
print(find_alpha('gym'))
df_se[df_se.gym==0].price_per_sqft.mean()
```

```
1263.8474455067726

1263.8474455067758
```

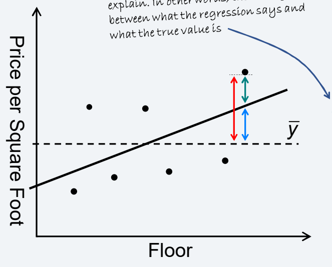
```
print(find_beta('gym'))
df_se[df_se.gym==1].price_per_sqft.mean() - df_se[df_se.gym==0].price_per_sqft.mean()
```

```
298.1799869483872

298.1799869483825
```

Errors

Errors in linear regression

The variation that the regression doesn't explain. In other words, the average error between what the regression says and what the true value is



The total amount of variation in the data – looking at all the prices, how much do they vary?

$$\text{SST} = \sum_{i=1}^{N}(y_i - \bar{y})^2 = N\sigma_Y^2$$

$$\text{SSE} = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{N}\varepsilon_i^2 = N\hat{\sigma}_\varepsilon^2$$

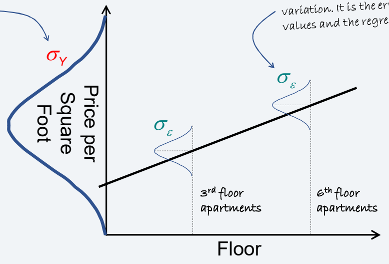
$$\text{SSR} = \sum_{i=1}^{N}(\hat{y}_i - \bar{y})^2$$

The variation the regression does explain; in other words, when we see all the prices vary, this is the amount of that variation that the regression can explain using the x-variables

Errors in linear regression

Total variation in price, which comes from the fact different prices correspond to different floors and that there is variation within floors

This is the total variation within each floor; the regression only gets to use floor as a variable, so it can't possibly capture this variation. It is the error between the true values and the regression prediction



Estimating the residual error σ_ε

Estimating σ_ε

- Estimating σ_ε from data proceeds just as you'd expect – you find the average error the regression makes
- However, we are estimating this from **limited data**
- Recall that when we found an estimate of the standard deviation from data, we had to divide by $N – 1$ to ensure our estimate was **unbiased**
- The same applies here, except we need to divide by $N – 2$

$$s_\varepsilon^2 = \frac{1}{N-2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

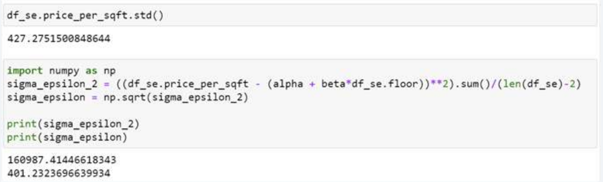
Why *N* – 2

- Fundamentally, the reason we divide by $N – 2$ is because

$$E\left(\frac{1}{N-2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2\right) = \sigma_\varepsilon^2$$

- This is – unfortunately – quite hard to show (see here – footnote 9 and the proof of Cochran's Theorem on page 27)
- One common explanation goes as follows
  - When estimating the standard deviation, we are already estimating the mean which removes **1** degree of freedom, and so we divide by ***N* – 1**
  - When estimating a regression, we are estimating **2** parameters, which removes **2** degrees of freedom, and so we divide by ***N* – 2**.
- I personally loathe this "logic", for reasons we'll discuss in class; but if it helps you remember the formula, it works well enough

Finding the standard error in our regression



```
df_se.price_per_sqft.std()
```

```
427.2751500848644
```

```
import numpy as np
sigma_epsilon_2 = ((df_se.price_per_sqft - (alpha + beta*df_se.floor))**2).sum()/(len(df_se)-2)
sigma_epsilon = np.sqrt(sigma_epsilon_2)

print(sigma_epsilon_2)
print(sigma_epsilon)
```

```
160987.41446618343
401.2323696639934
```

Side note; back to the likelihood…

$$\text{log likelihood } = \sum_{i=1}^{N}\log\left\{\frac{1}{\sigma\sqrt{2\pi}}\right\} - \frac{1}{2\sigma^2}\left[y_i - (\alpha + \beta x_i)\right]^2$$

$$= -\frac{N}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{N}\left[y_i - (\alpha + \beta x_i)\right]^2 + \text{constant}$$

Suppose we want to maximize this with respect to σ; let's differentiate this with respect to σ^2 and set to 0

$$-\frac{N}{2\hat{\sigma}^2} + \frac{1}{2\hat{\sigma}^4}\sum_{i=1}^{N}\left[y_i - (\hat{\alpha} + \hat{\beta}x_i)\right]^2 = 0$$

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}\left[y_i - (\hat{\alpha} + \hat{\beta}x_i)\right]^2 = \frac{1}{N}\sum_{i=1}^{N}\varepsilon_i^2$$

Properties of residuals

Residuals and predicted values

We can prove some important properties of residuals. Recall that linear regression solves the problem

$$\min_{\hat{\alpha},\hat{\beta}}\sum_{i=1}^{N}\varepsilon_i^2 \qquad \text{where } \varepsilon_i = (\hat{\alpha} + \hat{\beta}x_i - y_i)$$

When we differentiated with respect to $\hat{\alpha}$ and $\hat{\beta}$ and set them to 0, we found that

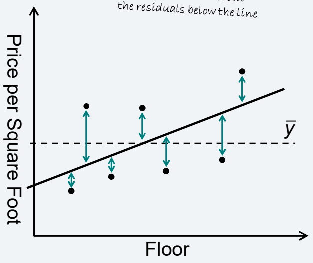
$$\frac{\partial}{\partial\alpha} = \boxed{\sum_{i=1}^{N}\varepsilon_i = 0} \qquad \frac{\partial}{\partial\beta} = \boxed{\sum_{i=1}^{N}x_i\varepsilon_i = 0}$$

If this weren't true, we would just change **α** until it becomes true

If this weren't true, we would just change **β** until it becomes true

Fact 1: mean residual is 0

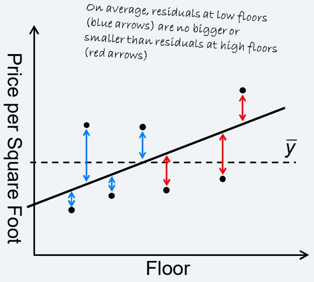
$$\bar{\varepsilon} = \frac{1}{N}\sum_{i=1}^{N}\varepsilon_i = 0$$



On average, the residuals above the line cancel out the residuals below the line

Fact 2: residuals uncorrelated to *x*-values

$$\begin{aligned}\mathrm{Corr}(x,\varepsilon) &\propto \sum_{i=1}^{N}(x_i-\bar{x})(\varepsilon_i-\bar{\varepsilon})\\ &= \sum_{i=1}^{N}(x_i-\bar{x})\varepsilon_i\\ &= \sum_{i=1}^{N}x_i\varepsilon_i - \bar{x}\sum_{i=1}^{N}\varepsilon_i\\ &= 0\end{aligned}$$

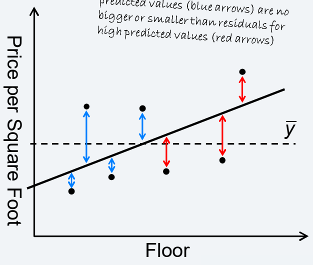


On average, residuals at low floors (blue arrows) are no bigger or smaller than residuals at high floors (red arrows)

Fact 3: residuals uncorrelated to predicted values

$E(\hat{y}_i) = E(\hat{\alpha}+\hat{\beta}x_i) = E(\bar{y}-\hat{\beta}\bar{x}+\hat{\beta}x_i) = \bar{y}$

$$\begin{aligned}\mathrm{Corr}(\hat{y},\varepsilon) &\propto \sum_{i=1}^{N}(\hat{y}_i-\bar{y})(\varepsilon_i-\bar{\varepsilon})\\ &= \sum_{i=1}^{N}\hat{y}_i\varepsilon_i\\ &= \sum_{i=1}^{N}(\hat{\alpha}+\hat{\beta}x_i)\varepsilon_i\\ &= \hat{\alpha}\sum_{i=1}^{N}\varepsilon_i + \hat{\beta}\sum_{i=1}^{N}x_i\varepsilon_i\\ &= 0\end{aligned}$$



On average, residuals for low predicted values (blue arrows) are no bigger or smaller than residuals for high predicted values (red arrows)

Fact 4: the beauty

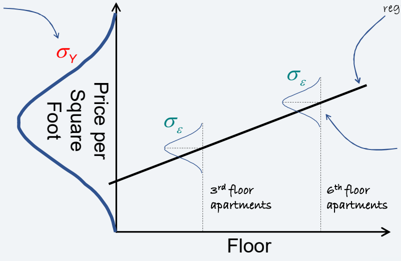
$$\begin{aligned}\mathrm{SST} &= \textstyle\sum_{i=1}^{N}(y_i-\bar{y})^2\\ &= \textstyle\sum_{i=1}^{N}(y_i-\hat{y}_i+\hat{y}_i-\bar{y})^2\\ &= \textstyle\sum_{i=1}^{N}(y_i-\hat{y}_i)^2 + \sum_{i=1}^{N}(\hat{y}_i-\bar{y})^2 + 2\sum_{i=1}^{N}(y_i-\hat{y}_i)(\hat{y}_i-\bar{y})\\ &= \mathrm{SSE}+\mathrm{SSR}+2\textstyle\sum_{i=1}^{N}(\hat{y}_i-\bar{y})\varepsilon_i\\ &= \mathrm{SSE}+\mathrm{SSR}\end{aligned}$$

We conclude

$$\boxed{\mathrm{SST} = \mathrm{SSE} + \mathrm{SSR}}$$

The beauty

This total variation in price per squared foot is equal to the sum of…

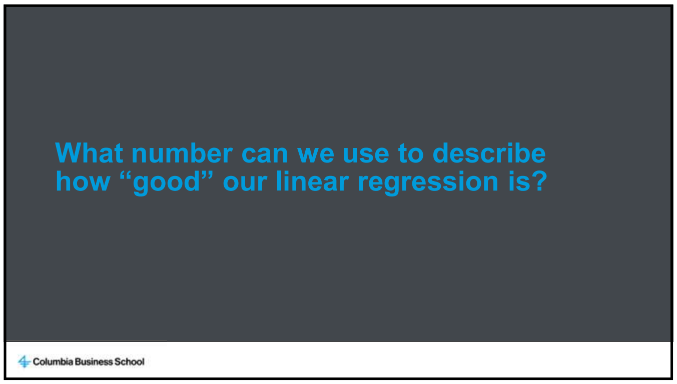


…the variation due to floor that is explained by our regression line…

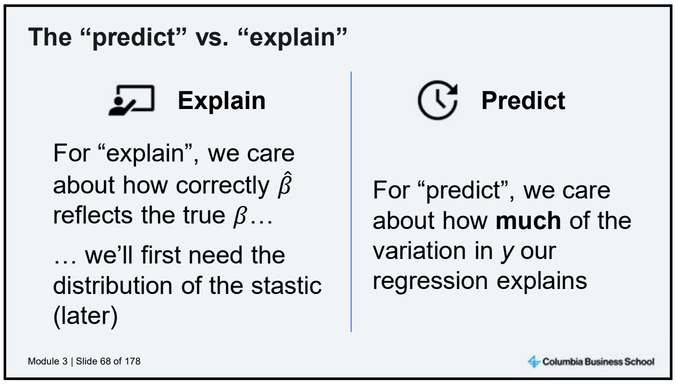
…and the residual variance left over within each floor

Columbia Business School
AT THE VERY CENTER OF BUSINESS

How “good” is a regression?



What number can we use to describe how “good” our linear regression is?



The “predict” vs. “explain”

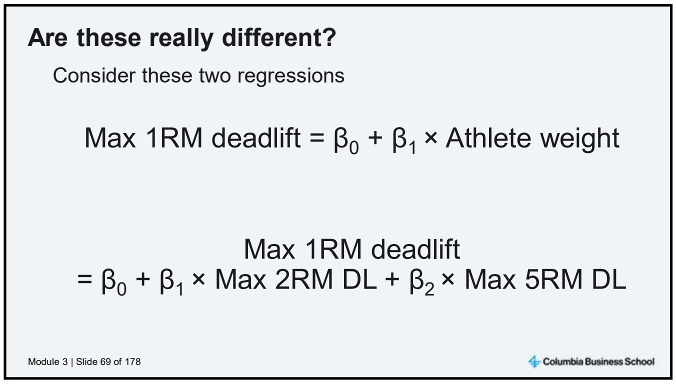
Explain

For “explain”, we care about how correctly $\hat{\beta}$ reflects the true β…

… we’ll first need the distribution of the stastic (later)

Predict

For “predict”, we care about how **much** of the variation in *y* our regression explains

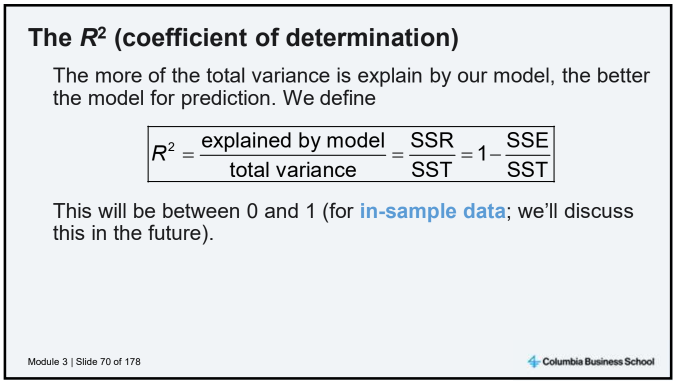


Are these really different?

Consider these two regressions

$$\text{Max 1RM deadlift} = \beta_0 + \beta_1 \times \text{Athlete weight}$$

$$\text{Max 1RM deadlift} = \beta_0 + \beta_1 \times \text{Max 2RM DL} + \beta_2 \times \text{Max 5RM DL}$$

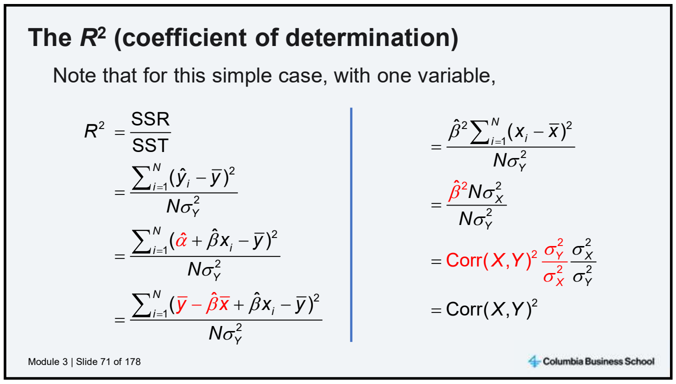


The *R*² (coefficient of determination)

The more of the total variance is explain by our model, the better the model for prediction. We define

$$\boxed{R^2 = \frac{\text{explained by model}}{\text{total variance}} = \frac{\text{SSR}}{\text{SST}} = 1 - \frac{\text{SSE}}{\text{SST}}}$$

This will be between 0 and 1 (for **in-sample data**; we’ll discuss this in the future).

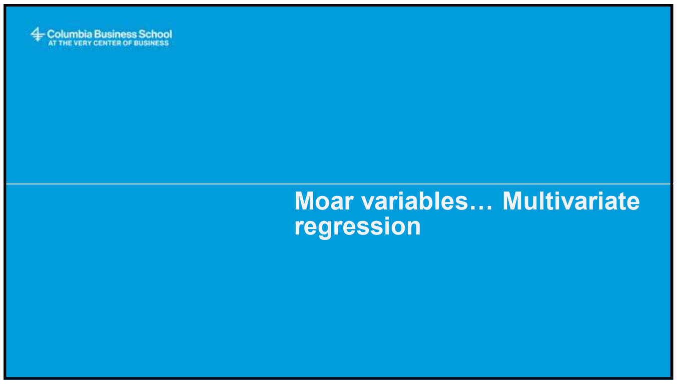


The *R*² (coefficient of determination)

Note that for this simple case, with one variable,

$$\begin{aligned} R^2 &= \frac{\text{SSR}}{\text{SST}} \\ &= \frac{\sum_{i=1}^{N}(\hat{y}_i - \overline{y})^2}{N\sigma_Y^2} \\ &= \frac{\sum_{i=1}^{N}(\hat{\alpha} + \hat{\beta}x_i - \overline{y})^2}{N\sigma_Y^2} \\ &= \frac{\sum_{i=1}^{N}(\overline{y} - \hat{\beta}\overline{x} + \hat{\beta}x_i - \overline{y})^2}{N\sigma_Y^2} \end{aligned}$$

$$\begin{aligned} &= \frac{\hat{\beta}^2\sum_{i=1}^{N}(x_i - \overline{x})^2}{N\sigma_Y^2} \\ &= \frac{\hat{\beta}^2 N\sigma_X^2}{N\sigma_Y^2} \\ &= \text{Corr}(X,Y)^2 \frac{\sigma_Y^2}{\sigma_X^2}\frac{\sigma_X^2}{\sigma_Y^2} \\ &= \text{Corr}(X,Y)^2 \end{aligned}$$



Columbia Business School
AT THE VERY CENTER OF BUSINESS

Moar variables… Multivariate regression

The “predict” vs. “explain”

Explain

Price per square foot =
1239.79 + 19.07 × Floor

Do higher floors really get a higher price per square foot? Or is it because apartments on higher floors have more bathrooms, making them more desirable?

Adding “number of bathrooms” to our regression can control for the number off bathrooms and reveal the true effect of the floor

Predict

Price per square foot =
1239.79 + 19.07 × Floor

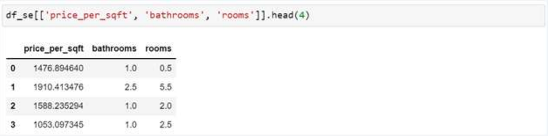
If our regression gets to use more characteristics of the apartment, the predictions are likely to be more accurate

Multivariate regression

- We have thus far been using **one** independent variable in our analysis. Multivariate regression uses **many** variables.
- With more variables, everything is more difficult
  - We can’t display things on a simple diagram
  - The proofs become more difficult; this isn’t a math class, so we won’t focus on these, but the intuition transfers from the univariate case
- With more difficulty comes a great reward!

Multivariate linear regression; matrix notation

When working with multivariate linear regression, it is simplest to work in matrix notation. As a simple example, let’s consider two variables only; `rooms` (the number of rooms in the apartment) and `bathrooms` (the number of bathrooms in the apartment). We’ll consider four rows only:



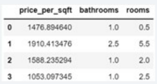
```
df_se[['price_per_sqft', 'bathrooms', 'rooms']].head(4)
```

```
   price_per_sqft  bathrooms  rooms
0     1476.894640        1.0    0.5
1     1910.413476        2.5    5.5
2     1588.235294        1.0    2.0
3     1053.097345        1.0    2.5
```

Multivariate linear regression; classical notation

$$y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \ldots$$

Multivariate linear regression; matrix notation



```
   price_per_sqft  bathrooms  rooms
0     1476.894640        1.0    0.5
1     1910.413476        2.5    5.5
2     1588.235294        1.0    2.0
3     1053.097345        1.0    2.5
```

We can express the regression as follows:

$$\boldsymbol{Y} = X\beta + \varepsilon$$

Vector of y-variables

Matrix of X variables (the first column represents the intercept)

Vector of coefficients

Vector of errors

$$\begin{pmatrix}1477\\1910\\1588\\1053\end{pmatrix} \quad \begin{pmatrix}1 & 1 & 0.5\\1 & 2.5 & 5.5\\1 & 1 & 2\\1 & 1 & 2.5\end{pmatrix} \quad \begin{pmatrix}\beta_0\\\beta_1\\\beta_2\end{pmatrix} \quad \begin{pmatrix}\varepsilon_0\\\varepsilon_1\\\varepsilon_2\\\varepsilon_3\end{pmatrix}$$

Intercept

Bathrooms

Rooms

Some matrix reminders (from pre-class note!)

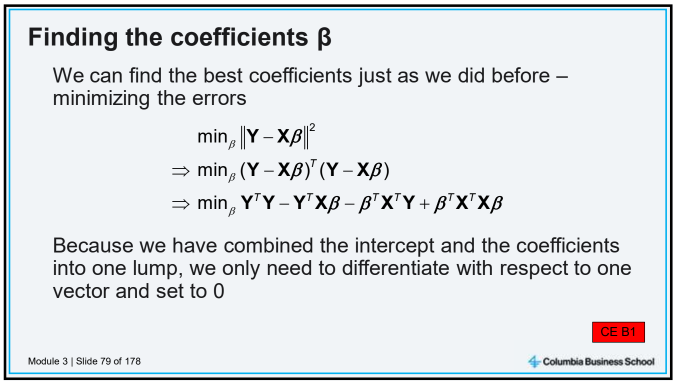
$$(\mathbf{X}^T)^T = \mathbf{X}$$

$$(\mathbf{XY})^T = \mathbf{Y}^T\mathbf{X}^T$$

$$\frac{\partial}{\partial \mathbf{X}}(\mathbf{XY}) = \frac{\partial}{\partial \mathbf{X}}(\mathbf{YX}) = \mathbf{Y}$$

$$\frac{\partial}{\partial \mathbf{X}}(\mathbf{X}^T\mathbf{Y}) = \frac{\partial}{\partial \mathbf{X}}(\mathbf{YX}^T) = \mathbf{Y}^T$$

$$\frac{\partial}{\partial \mathbf{X}}(\mathbf{X}^T\mathbf{YX}) = \mathbf{X}^T(\mathbf{Y}^T + \mathbf{Y})$$



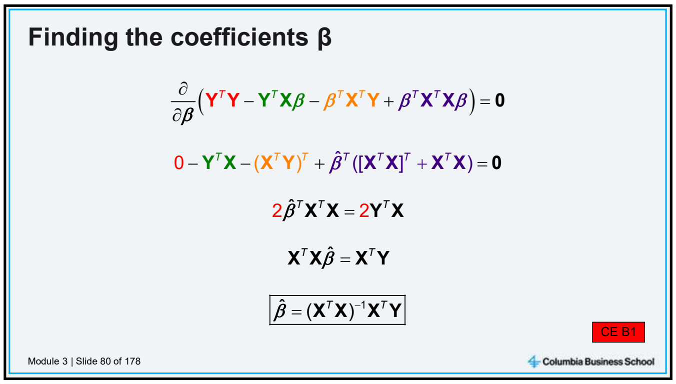
Finding the coefficients β

We can find the best coefficients just as we did before – minimizing the errors

$$\min_\beta \|\mathbf{Y}-\mathbf{X}\beta\|^2$$
$$\Rightarrow \min_\beta (\mathbf{Y}-\mathbf{X}\beta)^T(\mathbf{Y}-\mathbf{X}\beta)$$
$$\Rightarrow \min_\beta \mathbf{Y}^T\mathbf{Y}-\mathbf{Y}^T\mathbf{X}\beta-\beta^T\mathbf{X}^T\mathbf{Y}+\beta^T\mathbf{X}^T\mathbf{X}\beta$$

Because we have combined the intercept and the coefficients into one lump, we only need to differentiate with respect to one vector and set to 0

CE B1



Finding the coefficients β

$$\frac{\partial}{\partial\beta}\left(\mathbf{Y}^T\mathbf{Y}-\mathbf{Y}^T\mathbf{X}\beta-\beta^T\mathbf{X}^T\mathbf{Y}+\beta^T\mathbf{X}^T\mathbf{X}\beta\right)=\mathbf{0}$$

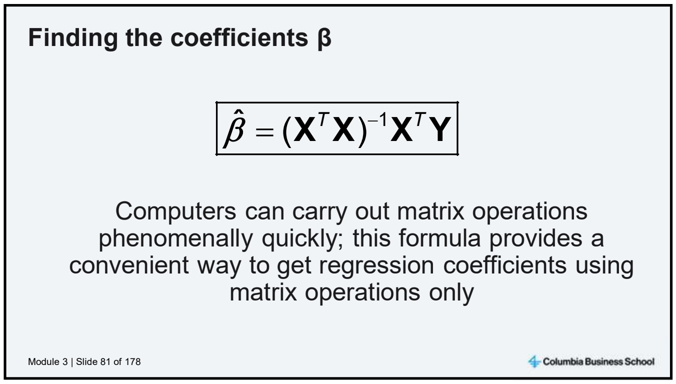
$$0-\mathbf{Y}^T\mathbf{X}-(\mathbf{X}^T\mathbf{Y})^T+\hat\beta^T([\mathbf{X}^T\mathbf{X}]^T+\mathbf{X}^T\mathbf{X})=\mathbf{0}$$

$$2\hat\beta^T\mathbf{X}^T\mathbf{X}=2\mathbf{Y}^T\mathbf{X}$$

$$\mathbf{X}^T\mathbf{X}\hat\beta=\mathbf{X}^T\mathbf{Y}$$

$$\boxed{\hat\beta=(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}}$$

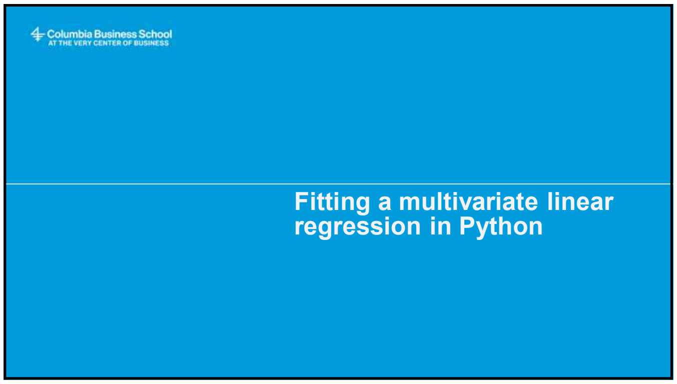
CE B1



Finding the coefficients β

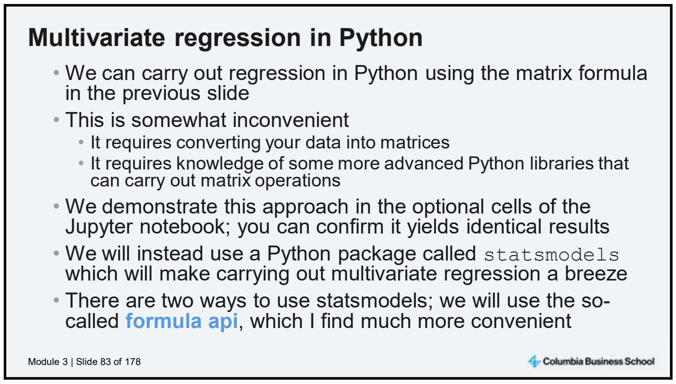
$$\boxed{\hat\beta=(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}}$$

Computers can carry out matrix operations phenomenally quickly; this formula provides a convenient way to get regression coefficients using matrix operations only



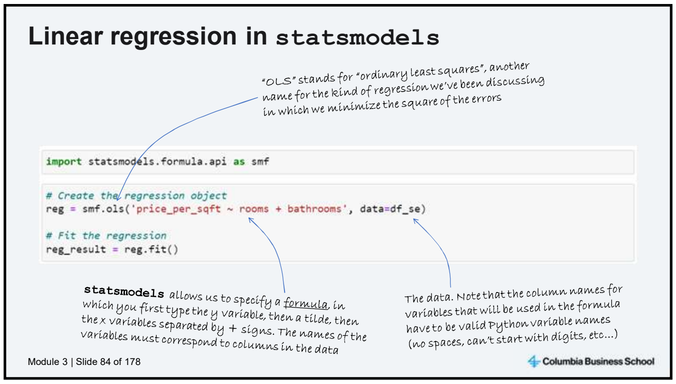
Columbia Business School
AT THE VERY CENTER OF BUSINESS

Fitting a multivariate linear regression in Python



Multivariate regression in Python

- We can carry out regression in Python using the matrix formula in the previous slide
- This is somewhat inconvenient
  - It requires converting your data into matrices
  - It requires knowledge of some more advanced Python libraries that can carry out matrix operations
- We demonstrate this approach in the optional cells of the Jupyter notebook; you can confirm it yields identical results
- We will instead use a Python package called `statsmodels` which will make carrying out multivariate regression a breeze
- There are two ways to use statsmodels; we will use the so-called **formula api**, which I find much more convenient



Linear regression in `statsmodels`

"OLS" stands for "ordinary least squares", another name for the kind of regression we've been discussing in which we minimize the square of the errors

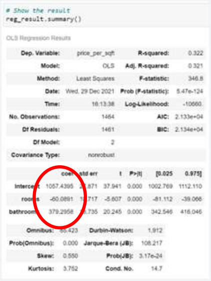
```
import statsmodels.formula.api as smf
```

```
# Create the regression object
reg = smf.ols('price_per_sqft ~ rooms + bathrooms', data=df_se)

# Fit the regression
reg_result = reg.fit()
```

`statsmodels` allows us to specify a formula, in which you first type the y variable, then a tilde, then the x variables separated by + signs. The names of the variables must correspond to columns in the data

The data. Note that the column names for variables that will be used in the formula have to be valid Python variable names (no spaces, can't start with digits, etc...)

View the regression results



```
# Show the result
reg_result.summary()
```

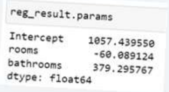
OLS Regression Results

Dep. Variable:	price_per_sqft	R-squared:	0.322
Model:	OLS	Adj. R-squared:	0.321
Method:	Least Squares	F-statistic:	346.8
Date:	Wed, 29 Dec 2021	Prob (F-statistic):	5.47e-124
Time:	16:13:38	Log-Likelihood:	-10660.
No. Observations:	1464	AIC:	2.133e+04
Df Residuals:	1461	BIC:	2.134e+04
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1057.4395	27.871	37.941	0.000	1002.769	1112.110
rooms	-60.0891	10.717	-5.607	0.000	-81.112	-39.066
bathrooms	379.2958	18.735	20.245	0.000	342.546	416.046

Omnibus:	65.423	Durbin-Watson:	1.912
Prob(Omnibus):	0.000	Jarque-Bera (JB):	108.217
Skew:	0.550	Prob(JB):	3.17e-24
Kurtosis:	3.752	Cond. No.	14.7

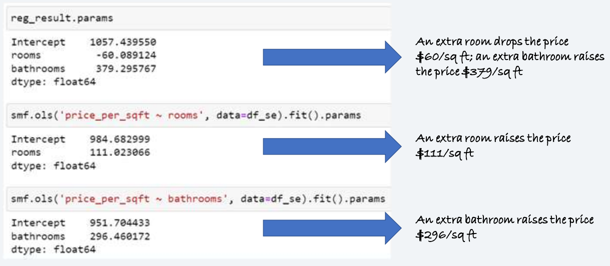
Price per square foot
= 1057
– 60 × rooms
+ 379 × bathrooms



```
reg_result.params

Intercept    1057.439550
rooms         -60.089124
bathrooms     379.295767
dtype: float64
```

Using multivariate linear regression to “explain”; controlling for other variables

Three regressions



```
reg_result.params

Intercept    1057.439550
rooms         -60.089124
bathrooms     379.295767
dtype: float64
```

An extra room drops the price $60/sq ft; an extra bathroom raises the price $379/sq ft

```
smf.ols('price_per_sqft ~ rooms', data=df_se).fit().params

Intercept    984.682999
rooms        111.023066
dtype: float64
```

An extra room raises the price $111/sq ft

```
smf.ols('price_per_sqft ~ bathrooms', data=df_se).fit().params

Intercept    951.704433
bathrooms    296.460172
dtype: float64
```

An extra bathroom raises the price $296/sq ft

How do we explain these seemingly contradictory conclusions?

Controlling for other variables

- Apartments with **more rooms** are **more expensive**, per sqft
- Apartments with **more bathrooms** are **more expensive**, per sqft
- **BUT**, apartments with more rooms have more bathrooms (the **correlation** between the two variables is **0.81**)
  - So maybe the only reason it looks like more rooms = more expensive is because of more bathrooms, or vice-versa
- When both variables are included, the regression figures out how much of the effect is due to each variable
- To be able to do this, the regression needs examples where one variable is high and the other is low
  - If the correlation between variables is too high, there won’t be such cases and the regression won’t be able to do its job – more on that later

Multivariate linear regression can disentangle the impact of multiple variables on the outcome. In other words, it can find the impact of one variable controlling for the effect of another

Are we now sure that the results of the larger regression are reliable? Are there any other variable that might change the picture?

Using multivariate linear regression to “predict”

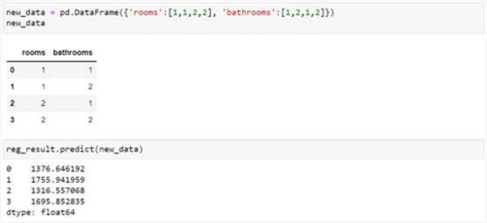
Predicted values

Suppose we have new values of the *x*-values, say $\mathbf{X}_{\text{new}}$. We can find an expression for the predicted values for these values of *x* from our multivariate regression

$$\hat{\mathbf{Y}} = \mathbf{X}_{\text{new}}\hat{\beta} = \mathbf{X}_{\text{new}}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

Predicted values from `statsmodels`

`statsmodels` can easily make predictions on new data



```
new_data = pd.DataFrame({'rooms':[1,1,2,2], 'bathrooms':[1,2,1,2]})
new_data
```

	rooms	bathrooms
0	1	1
1	1	2
2	2	1
3	2	2

```
reg_result.predict(new_data)
```

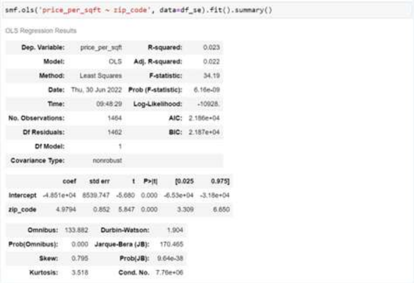
```
0    1376.646192
1    1755.941959
2    1316.557068
3    1695.852835
dtype: float64
```

Dealing with categorical variables

Categorical variables

- The regressions we have fit so far have all used **continuous** variables
- Our dataset contains some **categorical variables** – variables that can only take one of a few values, and that might not even be numeric
  - Property type (condo/co-op)
  - Zip code
  - etc…
- How can we use these in a regression? How do we get them to fit in an **X** matrix?
- There are a number of ways to do this – we’ll cover the **dummy variable encoding** or **one hot encoding**

Why can we not just do this?



```
smf.ols('price_per_sqft ~ zip_code', data=df_se).fit().summary()
```

```
OLS Regression Results

    Dep. Variable:   price_per_sqft        R-squared:     0.023
           Model:              OLS    Adj. R-squared:     0.022
          Method:    Least Squares        F-statistic:     34.19
            Date: Thu, 30 Jun 2022  Prob (F-statistic):  6.16e-09
            Time:         09:48:29    Log-Likelihood:    -10928.
No. Observations:             1464               AIC:  2.186e+04
    Df Residuals:             1462               BIC:  2.187e+04
        Df Model:                1
 Covariance Type:        nonrobust

                coef    std err       t   P>|t|     [0.025     0.975]
Intercept  -4.851e+04  8539.747  -5.680   0.000  -6.53e+04  -3.18e+04
zip_code      4.9794     0.852   5.847   0.000      3.309      6.650

      Omnibus: 133.882     Durbin-Watson:     1.904
Prob(Omnibus):   0.000  Jarque-Bera (JB):   170.465
         Skew:   0.795          Prob(JB):  9.64e-38
     Kurtosis:   3.518          Cond. No.  7.76e+06
```

Creating dummy variables

In the dummy variable approach, we create one column for every category of the variable:

Original data

Prop. Type
condo
condo
coop
coop
coop
condo

Dummy variables

Type_condo	Type_coop
1	0
1	0
0	1
0	1
0	1
1	0

We're now almost ready to fit our regression using these **new** variables

What if we have a categorical with > 2 values

Original data

ZIP
10023
10024
10023
10023
10025
10025

Dummy variables

ZIP_10023	ZIP_10024	ZIP_10025
1	0	0
0	1	0
1	0	0
1	0	0
0	0	1
0	0	1

Why can we not use these new variables directly in a regression?

The redundant dummy

- Remember regression disentangles the impact of various variables on the outcome
- If we fit a regression with both dummies, it's equivalent to disentangling the impact of
  - The property being a condo and not a co-op
  - The property being a co-op and not a condo
- But these are the **same thing** – the two columns basically contain **exactly the same data**, and have a correlation of 1
- So it's pointless to include both, and the regression won't be able to disentangle them

What if we have a categorical with > 2 values

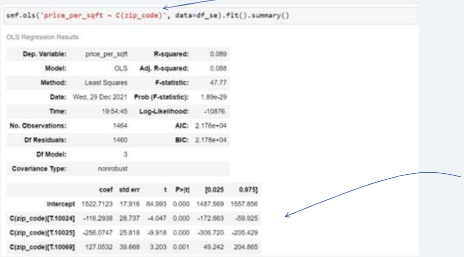
- The solution is to pick **one possible value** of the categorical variable as a **baseline**
- We then create dummy variables for **every other category**
- And finally, we fit the regression normally

When a categorical variable has *m* possible values, we pick one as the baseline, and we create dummies for the remaining *m* – 1 values

Dummy variables in Python

- Luckily, `statsmodels` will create dummy variables for us automatically – there's no need to do all of this manually
- The key is to surround the categorical variable with the `C()` keyword
- Let's look at an example with zip codes; the zip codes in the data are 10023, 10024, 10025, 10069

Dummy variables in Python



```
smf.ols('price_per_sqft ~ C(zip_code)', data=df_se).fit().summary()
```

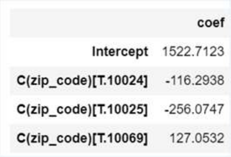
```
OLS Regression Results
     Dep. Variable:     price_per_sqft         R-squared:     0.089
             Model:                OLS     Adj. R-squared:    0.088
            Method:      Least Squares        F-statistic:    47.77
              Date:   Wed, 29 Dec 2021  Prob (F-statistic):  1.89e-29
              Time:           19:54:45     Log-Likelihood:   -10876.
  No. Observations:              1464                 AIC:  2.176e+04
      Df Residuals:              1460                 BIC:  2.178e+04
          Df Model:                 3
   Covariance Type:         nonrobust

                        coef   std err       t    P>|t|    [0.025    0.975]
         Intercept  1522.7123   17.916   84.993   0.000  1487.569  1557.856
C(zip_code)[T.10024] -116.2938  28.737   -4.047   0.000  -172.663   -59.925
C(zip_code)[T.10025] -256.0747  25.818   -9.918   0.000  -306.720  -205.429
C(zip_code)[T.10069]  127.0532  39.668    3.203   0.001    49.242   204.865
```

The C() keyword ensures the zip code is turned into a dummy

There are four different zip codes, but only three variables – 10023 is dropped as the baseline

How do we interpret these coefficients?

Interpreting the coefficients



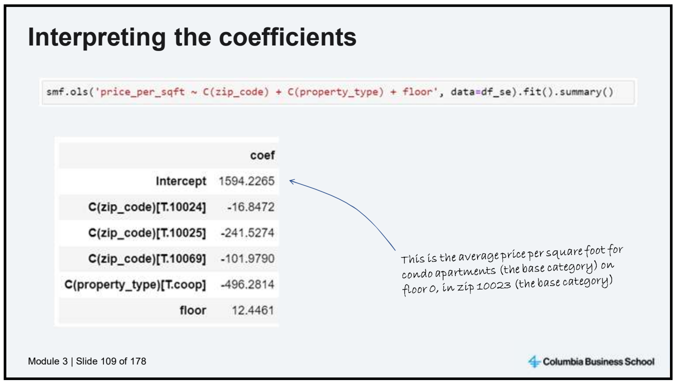
	coef
Intercept	1522.7123
C(zip_code)[T.10024]	-116.2938
C(zip_code)[T.10025]	-256.0747
C(zip_code)[T.10069]	127.0532

This is the average price per square foot for the baseline category (10023); properties in 10023 on average cost $1,523 per square foot

This is the "premium" for properties in 10024; on average, properties in that zip code cost $1,523 – $116 = $1,407 per square foot

This is the "premium" for properties in 10069; on average, properties in that zip code cost $1,523 + $127 = $1,650 per square foot

How do we interpret coefficients when there are multiple dummy variables and continuous variables

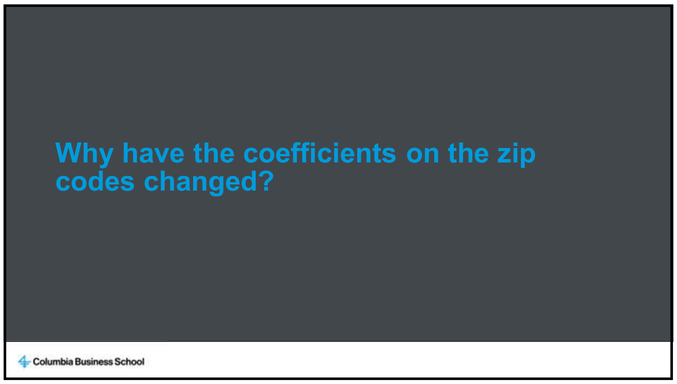


Interpreting the coefficients

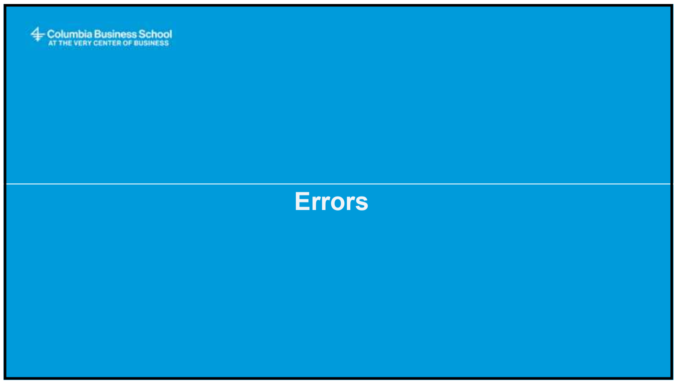
```
smf.ols('price_per_sqft ~ C(zip_code) + C(property_type) + floor', data=df_se).fit().summary()
```

	coef
Intercept	1594.2265
C(zip_code)[T.10024]	-16.8472
C(zip_code)[T.10025]	-241.5274
C(zip_code)[T.10069]	-101.9790
C(property_type)[T.coop]	-496.2814
floor	12.4461

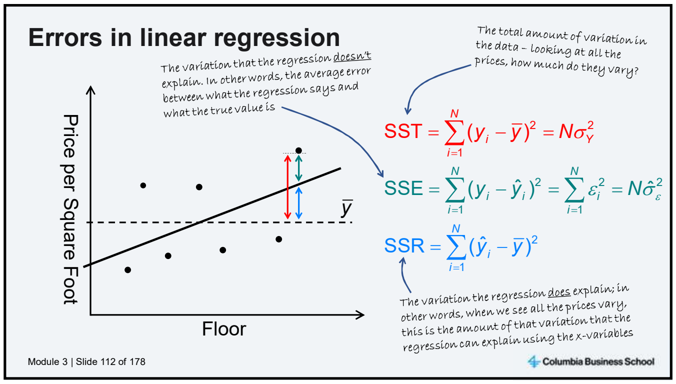
This is the average price per square foot for condo apartments (the base category) on floor 0, in zip 10023 (the base category)



Why have the coefficients on the zip codes changed?



Errors



Errors in linear regression

The variation that the regression doesn't explain. In other words, the average error between what the regression says and what the true value is

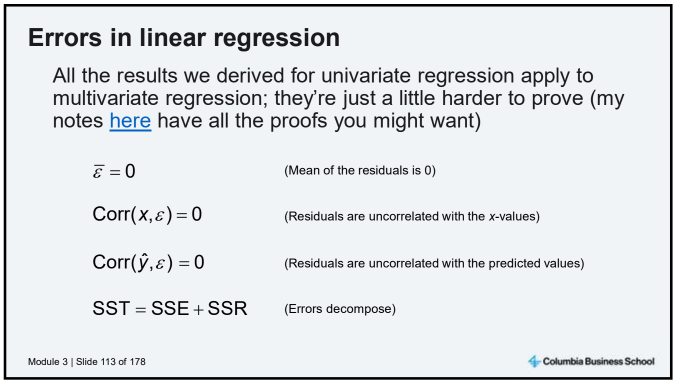
The total amount of variation in the data – looking at all the prices, how much do they vary?

$$\text{SST} = \sum_{i=1}^{N}(y_i - \bar{y})^2 = N\sigma_Y^2$$

$$\text{SSE} = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{N}\varepsilon_i^2 = N\hat{\sigma}_\varepsilon^2$$

$$\text{SSR} = \sum_{i=1}^{N}(\hat{y}_i - \bar{y})^2$$

The variation the regression does explain; in other words, when we see all the prices vary, this is the amount of that variation that the regression can explain using the x-variables



Errors in linear regression

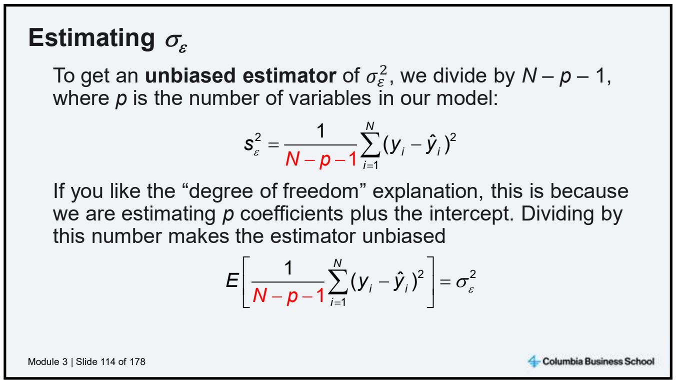
All the results we derived for univariate regression apply to multivariate regression; they're just a little harder to prove (my notes here have all the proofs you might want)

$\bar{\varepsilon} = 0$ (Mean of the residuals is 0)

$\text{Corr}(x, \varepsilon) = 0$ (Residuals are uncorrelated with the *x*-values)

$\text{Corr}(\hat{y}, \varepsilon) = 0$ (Residuals are uncorrelated with the predicted values)

$\text{SST} = \text{SSE} + \text{SSR}$ (Errors decompose)



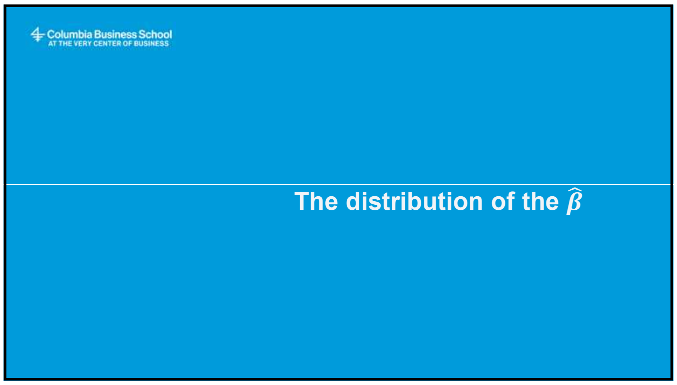
Estimating σ_ε

To get an **unbiased estimator** of σ_ε^2, we divide by $N - p - 1$, where p is the number of variables in our model:

$$s_\varepsilon^2 = \frac{1}{N - p - 1}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

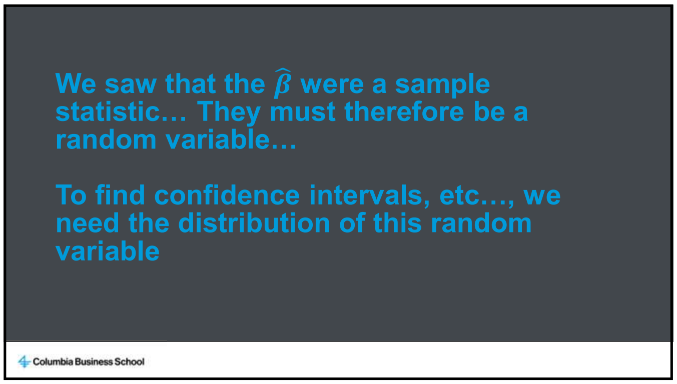
If you like the "degree of freedom" explanation, this is because we are estimating p coefficients plus the intercept. Dividing by this number makes the estimator unbiased

$$E\left[\frac{1}{N - p - 1}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2\right] = \sigma_\varepsilon^2$$



Columbia Business School
AT THE VERY CENTER OF BUSINESS

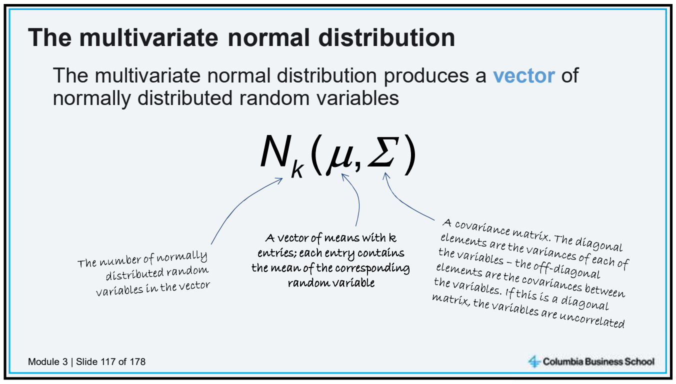
The distribution of the $\hat{\beta}$



We saw that the $\hat{\beta}$ were a sample statistic… They must therefore be a random variable…

To find confidence intervals, etc…, we need the distribution of this random variable

Columbia Business School



The multivariate normal distribution

The multivariate normal distribution produces a **vector** of normally distributed random variables

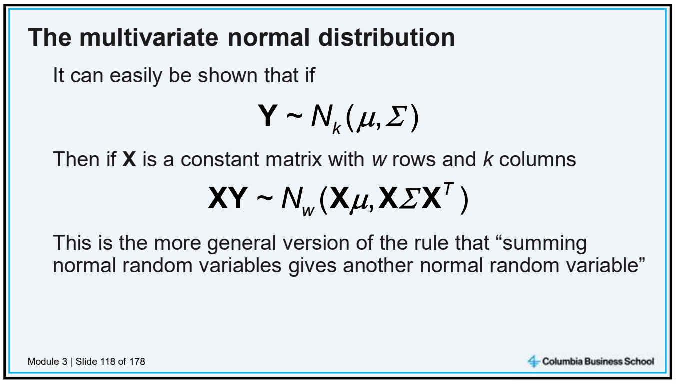
$$N_k(\mu, \Sigma)$$

The number of normally distributed random variables in the vector

A vector of means with k entries; each entry contains the mean of the corresponding random variable

A covariance matrix. The diagonal elements are the variances of each of the variables – the off-diagonal elements are the covariances between the variables. If this is a diagonal matrix, the variables are uncorrelated

Module 3 | Slide 117 of 178 Columbia Business School



The multivariate normal distribution

It can easily be shown that if

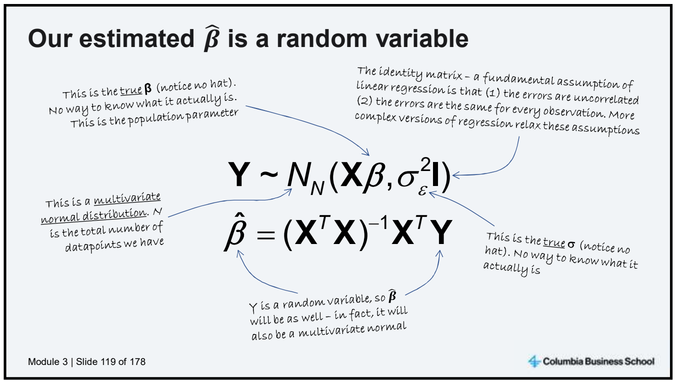
$$\mathbf{Y} \sim N_k(\mu, \Sigma)$$

Then if **X** is a constant matrix with *w* rows and *k* columns

$$\mathbf{XY} \sim N_w(\mathbf{X}\mu, \mathbf{X}\Sigma\mathbf{X}^T)$$

This is the more general version of the rule that "summing normal random variables gives another normal random variable"

Module 3 | Slide 118 of 178 Columbia Business School



Our estimated $\hat{\beta}$ is a random variable

This is the <u>true</u> **β** (notice no hat). No way to know what it actually is. This is the population parameter

The identity matrix – a fundamental assumption of linear regression is that (1) the errors are uncorrelated (2) the errors are the same for every observation. More complex versions of regression relax these assumptions

$$\mathbf{Y} \sim N_N(\mathbf{X}\beta, \sigma_\varepsilon^2\mathbf{I})$$

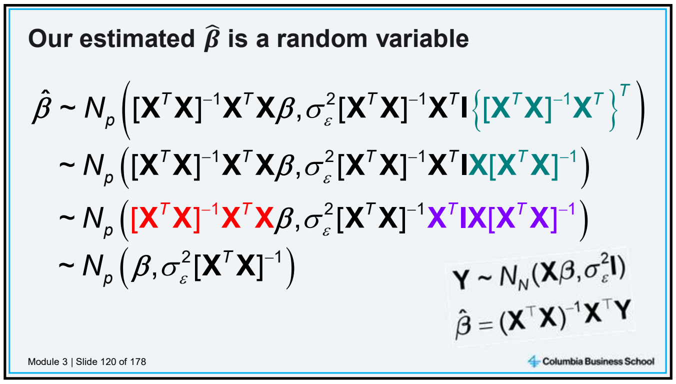
$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

This is a <u>multivariate normal distribution</u>. N is the total number of datapoints we have

This is the <u>true</u> **σ** (notice no hat). No way to know what it actually is

Y is a random variable, so $\hat{\boldsymbol{\beta}}$ will be as well – in fact, it will also be a multivariate normal

Module 3 | Slide 119 of 178 Columbia Business School



Our estimated $\hat{\beta}$ is a random variable

$$\hat{\beta} \sim N_p\left([\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{X}\beta, \sigma_\varepsilon^2[\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{I}\left\{[\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\right\}^T\right)$$

$$\sim N_p\left([\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{X}\beta, \sigma_\varepsilon^2[\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{I}\mathbf{X}[\mathbf{X}^T\mathbf{X}]^{-1}\right)$$

$$\sim N_p\left([\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{X}\beta, \sigma_\varepsilon^2[\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{I}\mathbf{X}[\mathbf{X}^T\mathbf{X}]^{-1}\right)$$

$$\sim N_p\left(\beta, \sigma_\varepsilon^2[\mathbf{X}^T\mathbf{X}]^{-1}\right)$$

$$\mathbf{Y} \sim N_N(\mathbf{X}\beta, \sigma_\varepsilon^2\mathbf{I})$$

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

Module 3 | Slide 120 of 178 Columbia Business School

Our estimated $\hat{\beta}$ is a random variable

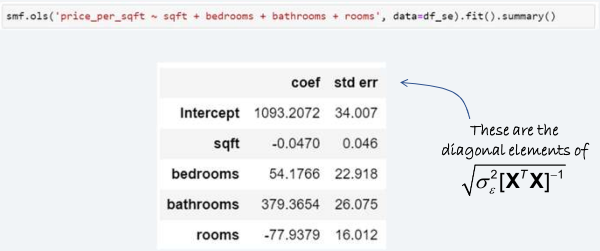
$$\hat{\beta} \sim N_p\left(\beta, \sigma_\varepsilon^2[\mathbf{X}^T\mathbf{X}]^{-1}\right)$$

We have shown that $\hat{\beta}$ is a normally distributed random variable. The mean is the true β which is fantastic news, but there's some variance around it, which comes from the errors in the data. Because there's some noise in the data, there will also be some noise in the β.

Finding these variances in practice

- We can find the variances manually
  - Estimate σ_ε^2 using s_ε^2.
  - Calculate $(\mathbf{X}^T\mathbf{X})^{-1}$
- We take this approach in the optional cell of the Jupyter notebook, but it quires some more advanced Python functionality
- Luckily, `statsmodels` can calculate these variances for us

Finding these variances in practice



```
smf.ols('price_per_sqft ~ sqft + bedrooms + bathrooms + rooms', data=df_se).fit().summary()
```

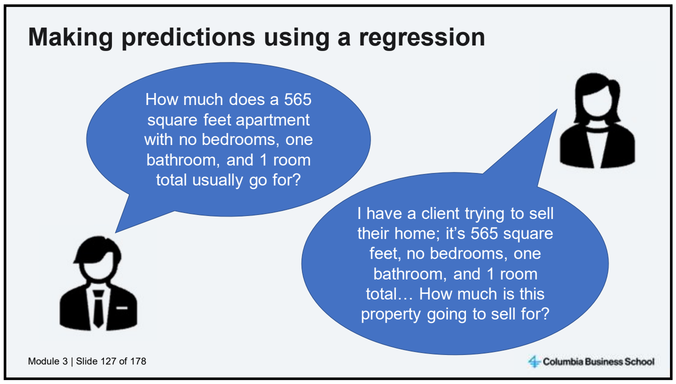
	coef	std err
Intercept	1093.2072	34.007
sqft	-0.0470	0.046
bedrooms	54.1766	22.918
bathrooms	379.3654	26.075
rooms	-77.9379	16.012

These are the diagonal elements of $\sqrt{\sigma_\varepsilon^2[\mathbf{X}^T\mathbf{X}]^{-1}}$

k. This is lovely, but why do I care?

It turns out the distribution on these $\hat{\beta}$ plays an essential role both in "explain" and "predict" regressions

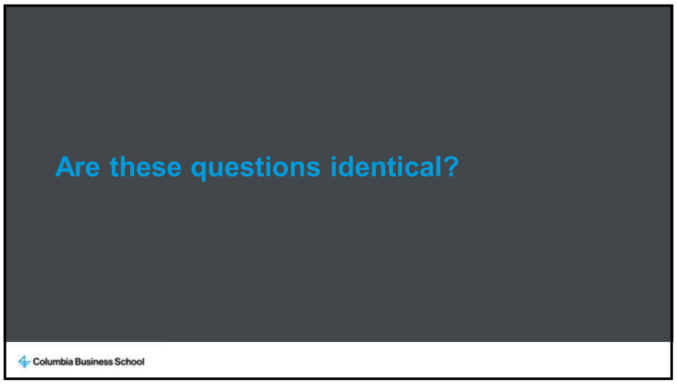
Errors in "predict" regressions: prediction and confidence intervals



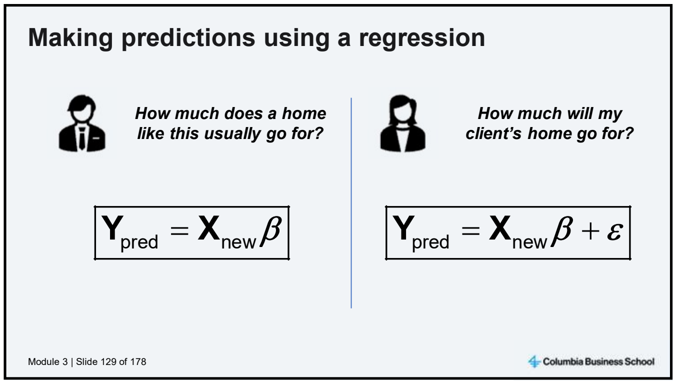
Making predictions using a regression

How much does a 565 square feet apartment with no bedrooms, one bathroom, and 1 room total usually go for?

I have a client trying to sell their home; it's 565 square feet, no bedrooms, one bathroom, and 1 room total… How much is this property going to sell for?



Are these questions identical?



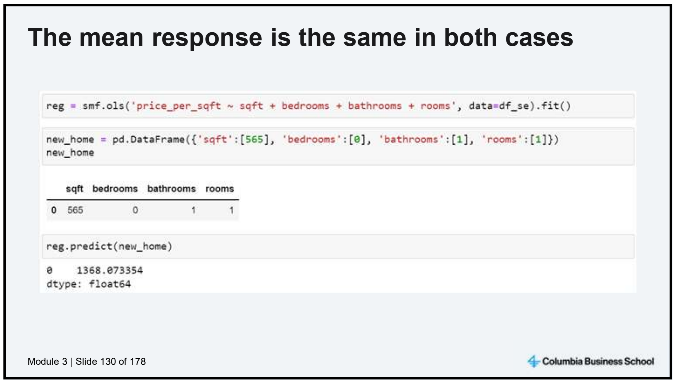
Making predictions using a regression

How much does a home like this usually go for?

$$\boxed{\mathbf{Y}_{\text{pred}} = \mathbf{X}_{\text{new}}\beta}$$

How much will my client's home go for?

$$\boxed{\mathbf{Y}_{\text{pred}} = \mathbf{X}_{\text{new}}\beta + \varepsilon}$$



The mean response is the same in both cases

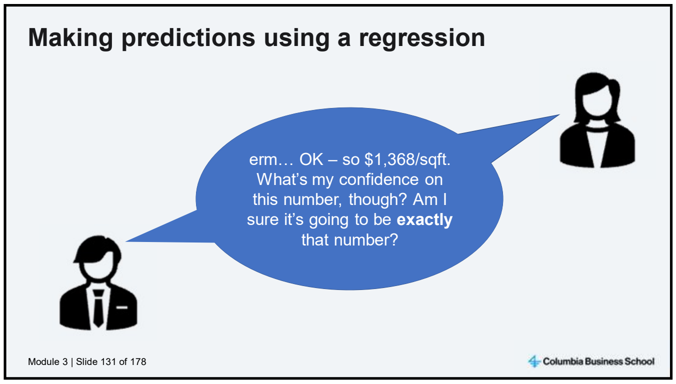
```
reg = smf.ols('price_per_sqft ~ sqft + bedrooms + bathrooms + rooms', data=df_se).fit()
```

```
new_home = pd.DataFrame({'sqft':[565], 'bedrooms':[0], 'bathrooms':[1], 'rooms':[1]})
new_home
```

	sqft	bedrooms	bathrooms	rooms
0	565	0	1	1

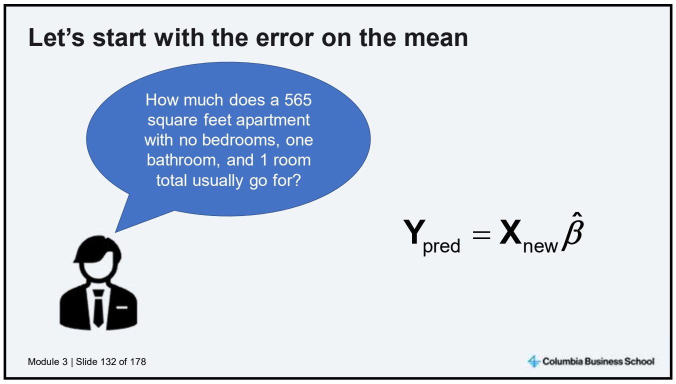
```
reg.predict(new_home)
```

```
0    1368.073354
dtype: float64
```



Making predictions using a regression

erm… OK – so $1,368/sqft. What's my confidence on this number, though? Am I sure it's going to be **exactly** that number?



Let's start with the error on the mean

How much does a 565 square feet apartment with no bedrooms, one bathroom, and 1 room total usually go for?

$$\mathbf{Y}_{\text{pred}} = \mathbf{X}_{\text{new}}\hat{\beta}$$

Where does uncertainty come from in this prediction?

Let’s start with the error on the mean

$$\mathbf{Y}_{\text{pred}} = \mathbf{X}_{\text{new}}\hat{\beta}$$

This is not the real $\boldsymbol{\beta}$. It’s $\hat{\boldsymbol{\beta}}$, a multivariate normal random variable! We can use the rules of multivariate normal to calculate the distribution of our predictions. Recall $\hat{\beta} \sim N_p\left(\beta, \sigma_\varepsilon^2[\mathbf{X}^T\mathbf{X}]^{-1}\right)$

$$\mathbf{Y}_{\text{pred}} \sim N(\mathbf{X}_{\text{new}}\beta, \sigma_\varepsilon^2 \mathbf{X}_{\text{new}}[\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}_{\text{new}}^T)$$

What about the error on a specific observation

$$\mathbf{Y}_{\text{pred}} = \mathbf{X}_{\text{new}}\hat{\beta} + \varepsilon$$

I have a client trying to sell their home; it’s 565 square feet, no bedrooms, one bathroom, and 1 room total… How much is this property going to sell for?

What about the error on a specific observation

$$\mathbf{Y}_{\text{pred}} = \mathbf{X}_{\text{new}}\hat{\beta} + \varepsilon$$

$N(\mathbf{0}, \sigma_\varepsilon^2\mathbf{I})$

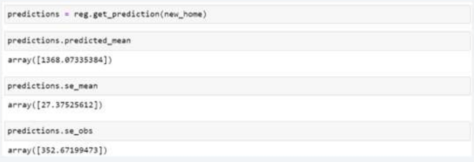
$$\mathbf{Y}_{\text{pred}} \sim N\left(\mathbf{X}_{\text{new}}\beta, \sigma_\varepsilon^2\left[1 + \mathbf{X}_{\text{new}}[\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}_{\text{new}}^T\right]\right)$$

Calculating these numbers in Python

- As before, we can calculate these numbers directly in Python
  - We first need to estimate σ_ε using s_ε
  - Then, we use the formula to calculate the covariance matrices
- Again, we do this in the optional cells of our Jupyter notebook, but it requires a little more Python than we’ve covered
- Instead, `statsmodels` can do this for us automatically!

Getting prediction variances in `statsmodels`

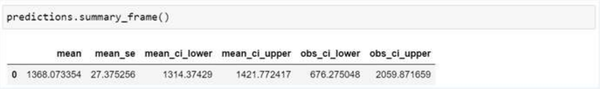
The standard error on the mean



```
predictions = reg.get_prediction(new_home)

predictions.predicted_mean
array([1368.07335384])

predictions.se_mean
array([27.37525612])

predictions.se_obs
array([352.67199473])
```

The standard error on a single observation

Confidence intervals in `statsmodels`



```
predictions.summary_frame()
```

	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	1368.073354	27.375256	1314.37429	1421.772417	676.275048	2059.871659

95% confidence interval on the mean

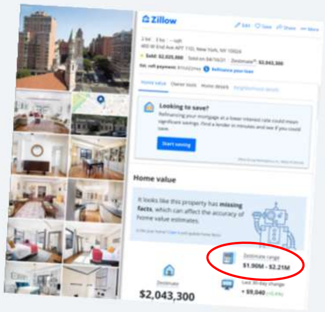
95% confidence interval on a single observation

Confidence intervals on these means

- Note that in theory, if we **knew** the **true** σ_ε^2, we'd be able to calculate these confidence intervals using a normal distribution
- Unfortunately, we don't – instead, we know s_ε^2 estimated from the errors, which isn't quite the same thing
- For that reason, we need a ***t-distribution***, not a normal distribution – and the details are beyond what we'll have time to cover
- Thankfully, `statsmodels` does it all for us

Predictions

400 West End Avenue
Apartment 11D



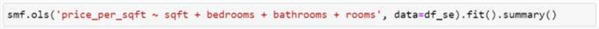
Columbia Business School
AT THE VERY CENTER OF BUSINESS

Errors in "explain" regressions: confidence intervals on the coefficients

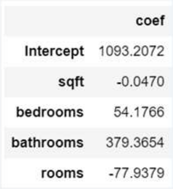
An "explain" regression

I'm going to build a building of 1 bed, 1 bath, 3 room total apartments; how should I use my space? More apartments but keep them smaller, or fewer larger apartments?

An "explain" regression



```
smf.ols('price_per_sqft ~ sqft + bedrooms + bathrooms + rooms', data=df_se).fit().summary()
```

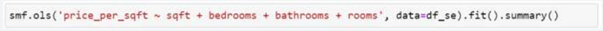


	coef
Intercept	1093.2072
sqft	-0.0470
bedrooms	54.1766
bathrooms	379.3654
rooms	-77.9379

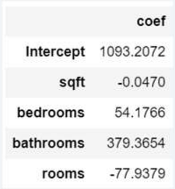
At first glance, this coefficient is negative; this means the smaller the apartment, the more expensive it is per square foot! So we should definitely build smaller apartments...

Does anything cause you to doubt that conclusion?

An “explain” regression



```
smf.ols('price_per_sqft ~ sqft + bedrooms + bathrooms + rooms', data=df_se).fit().summary()
```



	coef
Intercept	1093.2072
sqft	-0.0470
bedrooms	54.1766
bathrooms	379.3654
rooms	-77.9379

This number is a single draw from the distribution

$$\hat{\beta} \sim N_p\left(\beta, \sigma_\varepsilon^2[\mathbf{X}^T\mathbf{X}]^{-1}\right)$$

and what we care about isn’t the draw; it’s the true **β**. Does observing one single negative draw from this distribution tell you for sure that the true **β** is negative??

An analogy: suppose you flip a coin 20 times and it comes up heads 12 times; do you immediately conclude the coin is biased with *p* = P(head) = 0.6 ? In other words, does the single draw $\hat{p} = 0.6$ convince you the true $p \neq 0.5$?

A hypothesis test on $\hat{\beta}$

$$\hat{\beta} \sim N_p\left(\beta, \sigma_\varepsilon^2[\mathbf{X}^T\mathbf{X}]^{-1}\right)$$

- We observe a single draw from $\hat{\beta}_{\text{sqft}}$; in this case, –0.0470
- We want to carry out the following hypothesis test
  - **Null hypothesis H_0**: $\beta_{\text{sqft}} = 0$
  - **Alternative hypothesis H_1**: $\beta_{\text{sqft}} \neq 0$
- If we knew σ_ε^2 exactly, then we could say that under the null hypothesis,

$$\hat{\beta}_{\text{sqft}} \sim N\left(0, \sigma_\varepsilon^2[\mathbf{X}^T\mathbf{X}]^{-1}_{\text{sqft,sqft}}\right) \Rightarrow \frac{\hat{\beta}_{\text{sqft}}}{\sqrt{\sigma_\varepsilon^2[\mathbf{X}^T\mathbf{X}]^{-1}_{\text{sqft,sqft}}}} \sim N(0,1)$$

A hypothesis test on β

- Unfortunately, we do not know σ_ε^2 exactly. Instead, we have to use s_ε^2.
- It turns out, for reason that go beyond what we cover in this class, that under the null hypothesis,

$$\frac{\hat{\beta}_{\text{sqft}}}{\sqrt{s_\varepsilon^2[\mathbf{X}^T\mathbf{X}]^{-1}_{\text{sqft,sqft}}}} \sim t_{N-p}$$

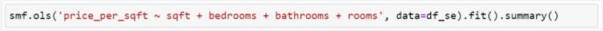
Student’s t distribution with N – p degrees of freedom

Number of variables in the regression

Number of data points

- Luckily, `statsmodels` will handle all the details for us!

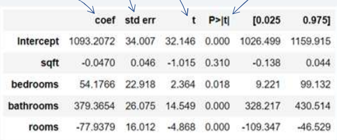
Hypothesis tests with `statsmodels`



```
smf.ols('price_per_sqft ~ sqft + bedrooms + bathrooms + rooms', data=df_se).fit().summary()
```

$\hat{\beta}_i$ $\sqrt{s_\varepsilon^2[\mathbf{X}^T\mathbf{X}]^{-1}_{i,i}}$ $\frac{\hat{\beta}_{\text{sqft}}}{\sqrt{s_\varepsilon^2[\mathbf{X}^T\mathbf{X}]^{-1}_{i,i}}}$

The probability of getting this draw from $\hat{\beta}_i$ assuming the null hypothesis $\beta_i = 0$ is true. If this is smaller than 5%, we reject the null hypothesis

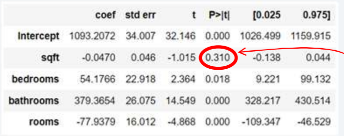


	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1093.2072	34.007	32.146	0.000	1026.499	1159.915
sqft	-0.0470	0.046	-1.015	0.310	-0.138	0.044
bedrooms	54.1766	22.918	2.364	0.018	9.221	99.132
bathrooms	379.3654	26.075	14.549	0.000	328.217	430.514
rooms	-77.9379	16.012	-4.868	0.000	-109.347	-46.529

The 95% confidence interval on $\hat{\beta}_i$. If the null hypothesis is rejected, this won’t include 0

Our conclusion

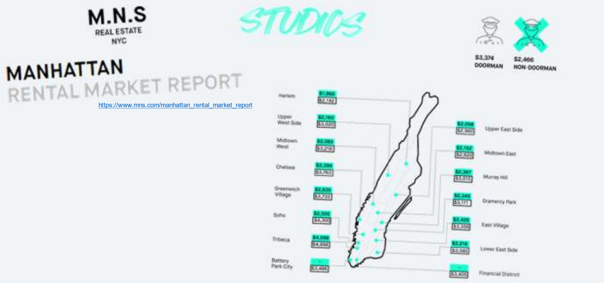
```
smf.ols('price_per_sqft ~ sqft + bedrooms + bathrooms + rooms', data=df_se).fit().summary()
```



	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1093.2072	34.007	32.146	0.000	1026.499	1159.915
sqft	-0.0470	0.046	-1.015	0.310	-0.138	0.044
bedrooms	54.1766	22.918	2.364	0.018	9.221	99.132
bathrooms	379.3654	26.075	14.549	0.000	328.217	430.514
rooms	-77.9379	16.012	-4.868	0.000	-109.347	-46.529

It looks like in this instance, we cannot reject the null hypothesis – there is not enough evidence to show that everything else being equal, the size of the apartment affects the price of the apartment per square feet

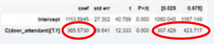
Regression is the right way to "explain"



https://www.mns.com/manhattan_rental_market_report

Regression is the right way to "explain"

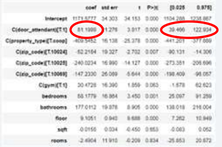
```
smf.ols('price_per_sqft ~ C(door_attendant)', data=df_se).fit().summary()
```



	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1113.5945	27.302	40.789	0.000	1060.040	1167.149
C(door_attendant)[T.1]	365.5730	29.641	12.333	0.000	307.429	423.717

"A doorman adds between $307 and $424 per sq foot"

```
smf.ols('''price_per_sqft ~ bedrooms + bathrooms + floor
                          + C(door_attendant) + C(property_type) + C(zip_code)
                          + sqft + rooms + C(gym)''', data=df_se).fit().summary()
```



	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1171.5777	34.303	34.153	0.000	1104.288	1238.867
C(door_attendant)[T.1]	81.1999	21.276	3.817	0.000	39.466	122.934
C(property_type)[T.coop]	-409.5452	16.138	-25.378	0.000	-441.201	-377.889
C(zip_code)[T.10024]	-52.2184	19.327	-2.702	0.007	-90.131	-14.306
C(zip_code)[T.10025]	-240.0234	16.990	-14.127	0.000	-273.351	-206.696
C(zip_code)[T.10069]	-147.2330	26.089	-5.644	0.000	-198.409	-96.057
C(gym)[T.1]	30.4726	16.390	1.859	0.063	-1.678	62.623
bedrooms	58.1779	16.864	3.450	0.001	25.097	91.259
bathrooms	177.0112	19.876	8.905	0.000	138.018	216.004
floor	9.1051	0.940	9.688	0.000	7.262	10.949
sqft	-0.0156	0.034	-0.450	0.653	-0.083	0.052
rooms	-2.4904	11.910	-0.209	0.834	-25.853	20.872

"A doorman adds between $39 and $123 per sq foot"

Columbia Business School
AT THE VERY CENTER OF BUSINESS

Multicollinearity

Multicollinearity

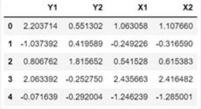
- **Multicollinearity** refers to the fact some variables in the data might be highly correlated
- This makes the regression much less reliable
  - Shows up as **broader confidence intervals**
- Two ways of thinking about why
  - If two variables are highly correlated, it's difficult to know which one causes variations in the outcome (eg: predicting
  - If two variables are highly correlated $\mathbf{X}^T\mathbf{X}$ is very hard to invert
- A common misconception I've seen is people getting scared when there is **any** correlation between variables. Wrong. Separating between correlated variables is precisely what linear regression is about! Trouble only arises when variables are **highly correlated**.

Columbia Business School
AT THE VERY CENTER OF BUSINESS

The F-test (optional)

Multicollinearity

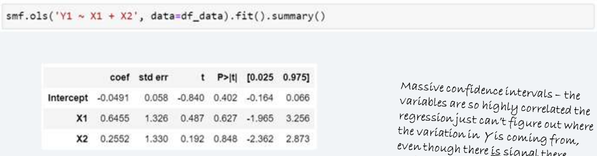
- We can demonstrate this using some **synthetic data**
- The notebook contains optional code that generates a data frame with four columns
  - Two variables **X1**, and **X2**, designed so that **Corr(X1, X2) = 0.999**
  - A variable **Y1**, generated so that **Y1 = X1 + ε**
  - A variable **Y2**, generated so that **Y2 = ε** (i.e., Y2 has no relationship to the X variables)



	Y1	Y2	X1	X2
0	2.203714	0.551302	1.063058	1.107660
1	-1.037392	0.419589	-0.249226	-0.316590
2	0.806762	1.815652	0.541528	0.615383
3	2.063392	-0.252750	2.435663	2.416482
4	-0.071639	-0.292004	-1.246239	-1.285001

Multicollinearity

Let's fit a first regression that tries to predict **Y1** using **X1** and **X2**



```
smf.ols('Y1 ~ X1 + X2', data=df_data).fit().summary()
```

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.0491	0.058	-0.840	0.402	-0.164	0.066
X1	0.6455	1.326	0.487	0.627	-1.965	3.256
X2	0.2552	1.330	0.192	0.848	-2.362	2.873

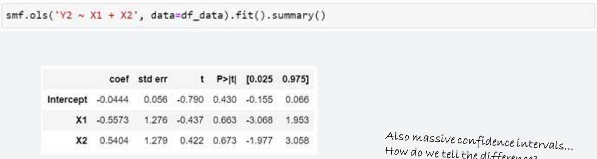
Massive confidence intervals – the variables are so highly correlated the regression just can't figure out where the variation in Y is coming from, even though there is signal there

In this case, there really is a signal in the data (Y = X1 + ε), we just can't find it.

How do we distinguish this from a situation in which there is truly no signal in the data at all?

No signal at all

Let's try and predict **Y2** using **X1** and **X2**; there's no signal there at all, Y2 is completely random



```
smf.ols('Y2 ~ X1 + X2', data=df_data).fit().summary()
```

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.0444	0.056	-0.790	0.430	-0.155	0.066
X1	-0.5573	1.276	-0.437	0.663	-3.068	1.953
X2	0.5404	1.279	0.422	0.673	-1.977	3.058

Also massive confidence intervals...
How do we tell the difference?

The F-test

- The *F*-test tests the regression **as a whole**
  - **Null hypothesis**: every β = 0
  - **Alternative hypothesis**: one or more β > 0
- Under the null hypothesis, it can be shown that

$$\frac{\frac{1}{p}\hat{\beta}^T\mathbf{X}^T\mathbf{X}\hat{\beta}}{s_\varepsilon^2} \sim F_{p,N-p}$$

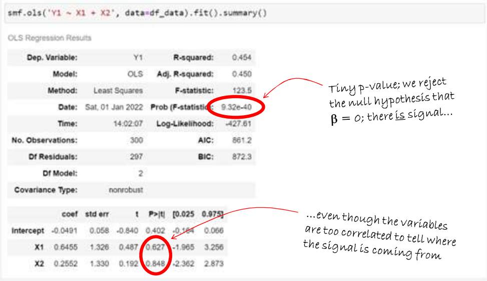
F distribution with p and N – p degrees of freedom

Number of variables in the regression

Number of data points

- As ever, `statsmodels` will handle all the gory details of the computation for us

A model with signal



```
smf.ols('Y1 ~ X1 + X2', data=df_data).fit().summary()
```

OLS Regression Results

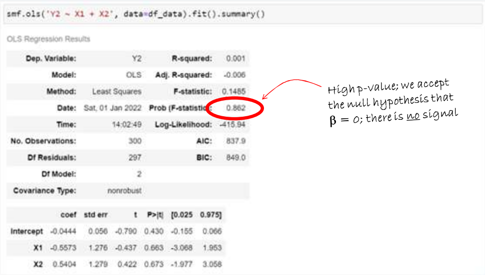
Dep. Variable:	Y1	R-squared:	0.454
Model:	OLS	Adj. R-squared:	0.450
Method:	Least Squares	F-statistic:	123.5
Date:	Sat, 01 Jan 2022	Prob (F-statistic):	9.32e-40
Time:	14:02:07	Log-Likelihood:	-427.61
No. Observations:	300	AIC:	861.2
Df Residuals:	297	BIC:	872.3
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.0491	0.058	-0.840	0.402	-0.164	0.066
X1	0.6455	1.326	0.487	0.627	-1.965	3.256
X2	0.2552	1.330	0.192	0.848	-2.362	2.873

Tiny p-value; we reject the null hypothesis that β = 0; there is signal...

...even though the variables are too correlated to tell where the signal is coming from

A model with no signal



```
smf.ols('Y2 ~ X1 + X2', data=df_data).fit().summary()
```

```
OLS Regression Results

   Dep. Variable:              Y2        R-squared:   0.001
           Model:             OLS   Adj. R-squared:  -0.006
          Method:   Least Squares      F-statistic:  0.1485
            Date: Sat, 01 Jan 2022 Prob (F-statistic):  0.862
            Time:        14:02:49   Log-Likelihood: -415.94
No. Observations:             300              AIC:   837.9
    Df Residuals:             297              BIC:   849.0
        Df Model:               2
 Covariance Type:       nonrobust

             coef  std err       t  P>|t|  [0.025  0.975]
Intercept -0.0444    0.056  -0.790  0.430  -0.155   0.066
       X1 -0.5573    1.276  -0.437  0.663  -3.068   1.953
       X2  0.5404    1.279   0.422  0.673  -1.977   3.058
```

High p-value; we accept the null hypothesis that $\beta = 0$; there is no signal

Columbia Business School
AT THE VERY CENTER OF BUSINESS

Variable selection

Variable selection

- Throughout this lecture, we have been fitting a variety of regressions, with a variety of variables
- We've seen that adding or removing one variable can have a **massive effect** on the coefficients (and its confidence intervals and *p*-values)
- This begs the question – when we have a lot of variables, **which should we include**?
- This is called **variable selection**
- Variable selection is an enormously complex topic – we'll scratch the surface here; more in **Applied Regression Analysis**, and **BA2**

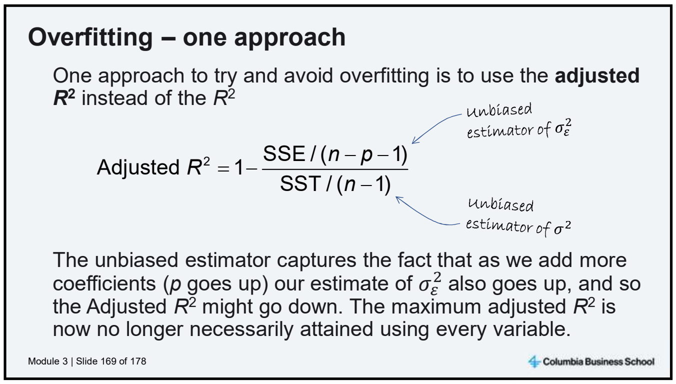
Why not include every variable

Why not include every variable?

- Suppose this apartment sold for an unusually high price
- What would happen in our regression if we added a dummy variable for leopard print?
- Should we add the variable?

Overfitting

- Every **extra variable** can only help **reduce SSE**, and make the **R^2 higher**
- However, with too many variables, the regression will start capturing some **spurious correlations** in the data
- As such, we'd like to include **just enough variables** to **capture the signal**, but **not so many** that we start capturing **noise**

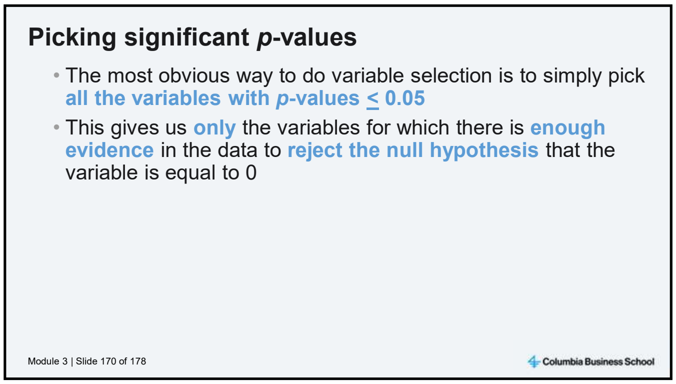


Overfitting – one approach

One approach to try and avoid overfitting is to use the **adjusted *R*²** instead of the R^2

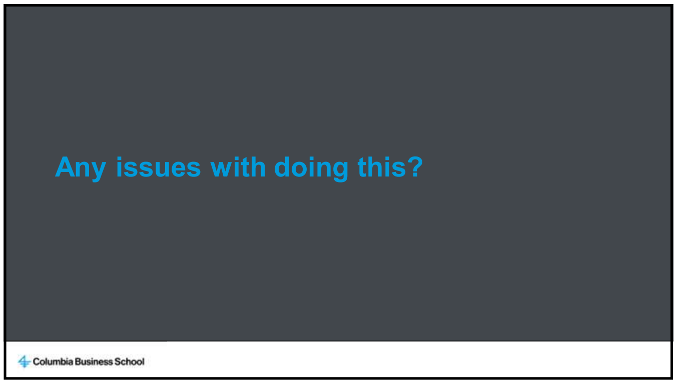
$$\text{Adjusted } R^2 = 1 - \frac{\overset{\text{unbiased estimator of } \sigma_\varepsilon^2}{\text{SSE}/(n-p-1)}}{\underset{\text{unbiased estimator of } \sigma^2}{\text{SST}/(n-1)}}$$

The unbiased estimator captures the fact that as we add more coefficients (*p* goes up) our estimate of σ_ε^2 also goes up, and so the Adjusted R^2 might go down. The maximum adjusted R^2 is now no longer necessarily attained using every variable.

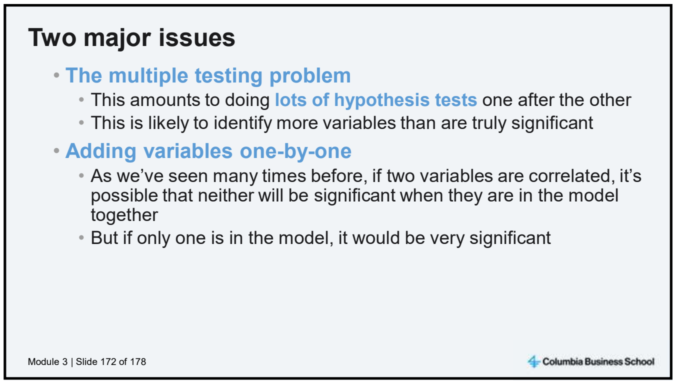


Picking significant *p*-values

- The most obvious way to do variable selection is to simply pick **all the variables with *p*-values ≤ 0.05**
- This gives us **only** the variables for which there is **enough evidence** in the data to **reject the null hypothesis** that the variable is equal to 0

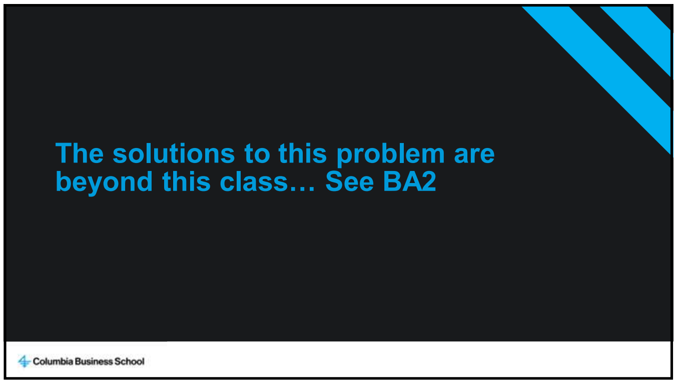


Any issues with doing this?

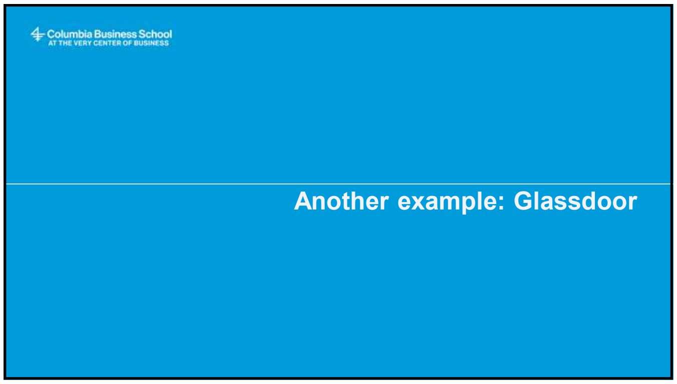


Two major issues

- **The multiple testing problem**
  - This amounts to doing **lots of hypothesis tests** one after the other
  - This is likely to identify more variables than are truly significant
- **Adding variables one-by-one**
  - As we've seen many times before, if two variables are correlated, it's possible that neither will be significant when they are in the model together
  - But if only one is in the model, it would be very significant



The solutions to this problem are beyond this class… See BA2



Columbia Business School
AT THE VERY CENTER OF BUSINESS

Another example: Glassdoor

Glassdoor jobs report

glassdoor | ECONOMIC RESEARCH

Why Do Candidates Reject Job Offers?

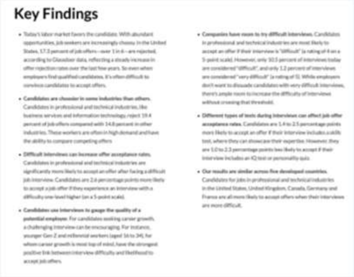
An Analysis of International Data from Glassdoor

Jason Sockin
Economic Research Fellow

Daniel Zhao
Senior Economist & Data Scientist

https://www.glassdoor.com/research/app/uploads/sites/2/2020/02/20-02-27-Glassdoor-Econ-JobRejection-Report.pdf

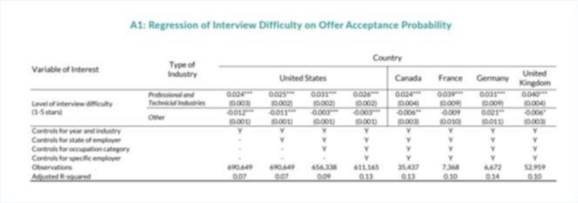
Glassdoor jobs report



Key Findings

- Today's labor market favors the candidate. With abundant opportunities, job seekers are increasingly choosy. In the United States, 17.2 percent of job offers—over 1 in 6—are rejected, according to Glassdoor data, reflecting a steady increase in offer rejection rates over the last few years. So even when employers find qualified candidates, it's often difficult to convince candidates to accept offers.
- **Candidates are choosier in some industries than others.** Candidates in professional and technical industries, like business services and information technology, reject 19.4 percent of job offers compared with 14.8 percent in other industries. These workers are often in high demand and have the ability to compare competing offers.
- **Difficult interviews can increase offer acceptance rates.** Candidates in professional and technical industries are significantly more likely to accept an offer after facing a difficult job interview. Candidates are 2.6 percentage points more likely to accept a job offer if they experience an interview with a difficulty one level higher (on a 5-point scale).
- **Candidates use interviews to gauge the quality of a potential employer.** For candidates seeking career growth, a challenging interview can be encouraging. For instance, younger Gen Z and millennial workers (aged 16 to 34), for whom career growth is most top of mind, have the strongest positive link between interview difficulty and likelihood to accept job offers.
- **Companies have room to try difficult interviews.** Candidates in professional and technical industries are most likely to accept an offer if their interview is "difficult" (a rating of 4 on a 5-point scale). However, only 10.5 percent of interviews today are considered "difficult", and only 1.2 percent of interviews are considered "very difficult" (a rating of 5). While employers don't want to dissuade candidates with very difficult interviews, there's ample room to increase the difficulty of interviews without crossing that threshold.
- **Different types of tests during interviews can affect job offer acceptance rates.** Candidates are 1.4 to 2.5 percentage points more likely to accept an offer if their interview includes a skills test, where they can showcase their expertise. However, they are 1.0 to 2.3 percentage points less likely to accept if their interview includes an IQ test or personality quiz.
- **Our results are similar across five developed countries.** Candidates for jobs in professional and technical industries in the United States, United Kingdom, Canada, Germany and France are all more likely to accept offers when their interviews are more difficult.

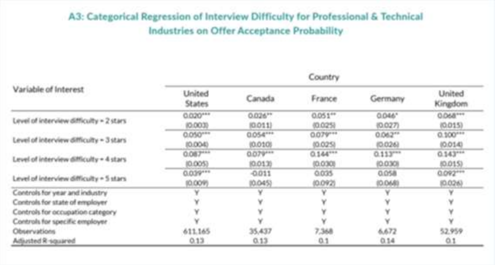
Glassdoor jobs report



A1: Regression of Interview Difficulty on Offer Acceptance Probability

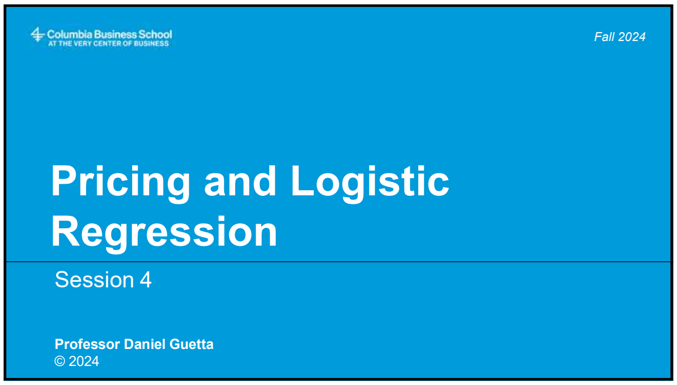
Variable of Interest	Type of Industry	Country							
		United States				Canada	France	Germany	United Kingdom
Level of interview difficulty (1-5 stars)	*Professional and Technicial Industries*	0.024*** (0.003)	0.025*** (0.002)	0.031*** (0.002)	0.026*** (0.002)	0.024*** (0.004)	0.039*** (0.009)	0.031*** (0.009)	0.040*** (0.004)
	Other	-0.012*** (0.001)	-0.011*** (0.001)	-0.003*** (0.001)	-0.003*** (0.001)	-0.006** (0.003)	-0.009 (0.010)	0.021** (0.011)	-0.006* (0.003)
Controls for year and industry		Y	Y	Y	Y	Y	Y	Y	Y
Controls for state of employer		-	Y	Y	Y	Y	Y	Y	Y
Controls for occupation category		-	-	Y	Y	Y	Y	Y	Y
Controls for specific employer		-	-	-	Y	Y	Y	Y	Y
Observations		690,649	690,649	656,338	611,165	35,437	7,368	6,672	52,959
Adjusted R-squared		0.07	0.07	0.09	0.13	0.13	0.10	0.14	0.10

Glassdoor jobs report



A3: Categorical Regression of Interview Difficulty for Professional & Technical Industries on Offer Acceptance Probability

Variable of Interest	Country				
	United States	Canada	France	Germany	United Kingdom
Level of interview difficulty = 2 stars	0.020*** (0.003)	0.026** (0.011)	0.051** (0.025)	0.046* (0.027)	0.068*** (0.015)
Level of interview difficulty = 3 stars	0.050*** (0.004)	0.054*** (0.010)	0.079*** (0.025)	0.062** (0.026)	0.100*** (0.014)
Level of interview difficulty = 4 stars	0.087*** (0.005)	0.079*** (0.013)	0.144*** (0.030)	0.113*** (0.030)	0.143*** (0.015)
Level of interview difficulty = 5 stars	0.039*** (0.009)	-0.011 (0.045)	0.035 (0.092)	0.058 (0.068)	0.092*** (0.026)
Controls for year and industry	Y	Y	Y	Y	Y
Controls for state of employer	Y	Y	Y	Y	Y
Controls for occupation category	Y	Y	Y	Y	Y
Controls for specific employer	Y	Y	Y	Y	Y
Observations	611,165	35,437	7,368	6,672	52,959
Adjusted R-squared	0.13	0.13	0.1	0.14	0.1

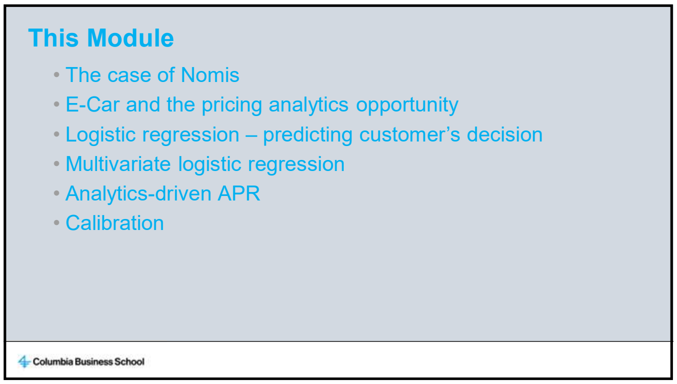


Fall 2024

Pricing and Logistic Regression

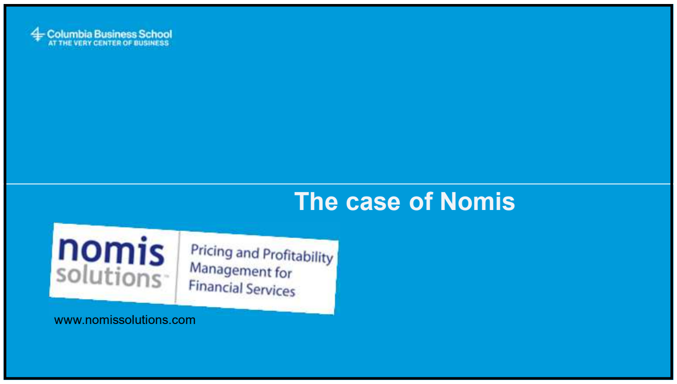
Session 4

Professor Daniel Guetta
© 2024



This Module

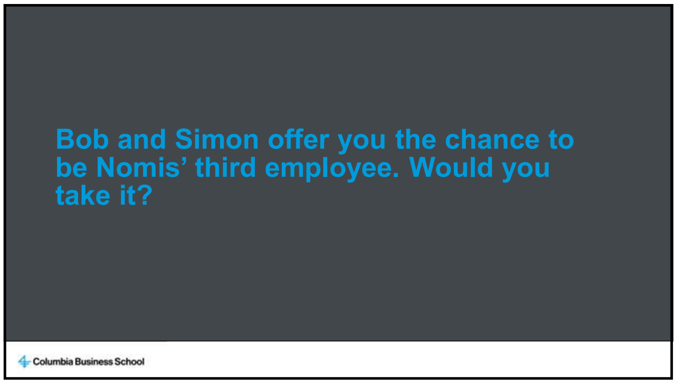
- The case of Nomis
- E-Car and the pricing analytics opportunity
- Logistic regression – predicting customer's decision
- Multivariate logistic regression
- Analytics-driven APR
- Calibration



The case of Nomis

nomis solutions
Pricing and Profitability Management for Financial Services

www.nomissolutions.com



Bob and Simon offer you the chance to be Nomis' third employee. Would you take it?

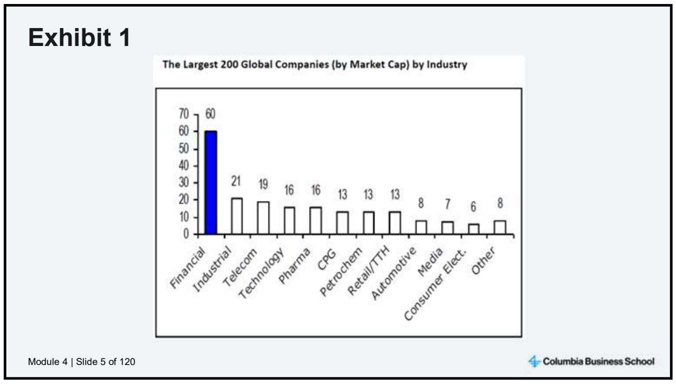


Exhibit 1

The Largest 200 Global Companies (by Market Cap) by Industry

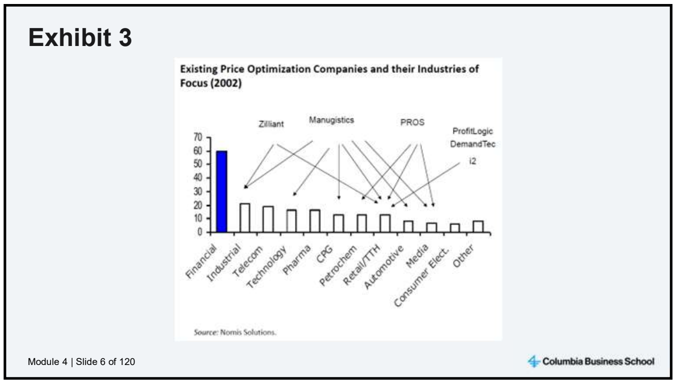
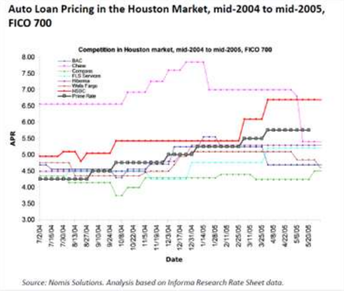


Exhibit 3

Existing Price Optimization Companies and their Industries of Focus (2002)

Source: Nomis Solutions.

Exhibit 6

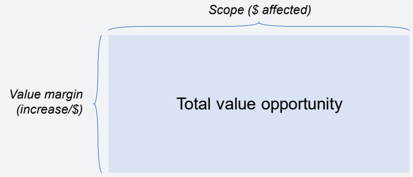


Auto Loan Pricing in the Houston Market, mid-2004 to mid-2005, FICO 700

Source: Nomis Solutions. Analysis based on Informa Research Rate Sheet data.

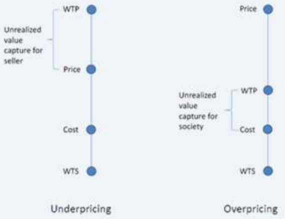
Assessing the size of an opportunity

- **Value margin**: how much can analytics improve each transaction or decision
- **Scope**: how many transactions and decisions can we improve?



Where does the analytic value come from?

The value stick

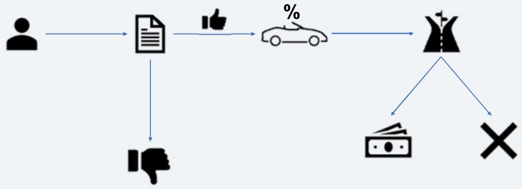


WTP
Unrealized value capture for seller
Price
Cost
WTS
Underpricing
Price
Unrealized value capture for society
WTP
Cost
WTS
Overpricing

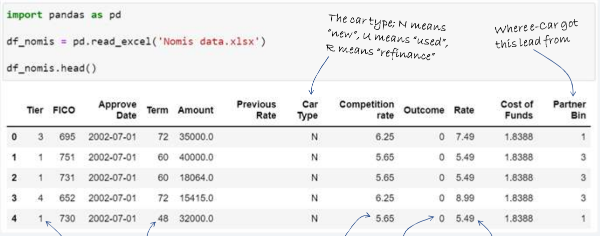
Business analytics can help firms capture unrealized value from customers willing to pay more by raising prices, or increase their customer base by lowering prices

e-Car

What is the process of getting a loan at e-Car?



The e-Car data (208,085 rows)



```
import pandas as pd

df_nomis = pd.read_excel('Nomis data.xlsx')

df_nomis.head()
```

	Tier	FICO	Approve Date	Term	Amount	Previous Rate	Car Type	Competition rate	Outcome	Rate	Cost of Funds	Partner Bin
0	3	695	2002-07-01	72	35000.0		N	6.25	0	7.49	1.8388	1
1	1	751	2002-07-01	60	40000.0		N	5.65	0	5.49	1.8388	3
2	1	731	2002-07-01	60	18064.0		N	5.65	0	5.49	1.8388	3
3	4	652	2002-07-01	72	15415.0		N	6.25	0	8.99	1.8388	3
4	1	730	2002-07-01	48	32000.0		N	5.65	0	5.49	1.8388	1

The car type; N means "new", U means "used", R means "refinance"

Where e-Car got this lead from

How good is this person's FICO score?

How long is the loan (months)

Rate offered by competitors around that time

Did the customer accept the loan (1) or not (0)

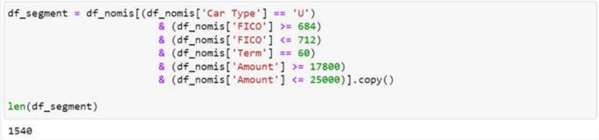
Rate offered to the consumer

Let's start by focusing on a single segment

Starting with an easier problem

- With any problem like this one, it's helpful to begin with a smaller, simpler segment of the data to understand what's happening
- We will use
  - Used cars
  - Borrowers with FICO scoes between 684 and 712
  - Loans with a term of 60 months
  - Loan amounts between 17.8K and 25K
- How could we determine whether e-Car is mispricing loans in this segment?

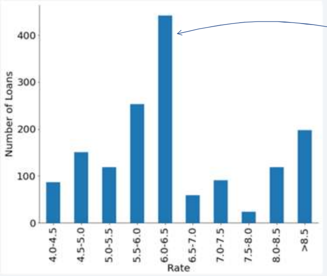
Starting with an easier problem



```
df_segment = df_nomis[(df_nomis['Car Type'] == 'U')
                      & (df_nomis['FICO'] >= 684)
                      & (df_nomis['FICO'] <= 712)
                      & (df_nomis['Term'] == 60)
                      & (df_nomis['Amount'] >= 17800)
                      & (df_nomis['Amount'] <= 25000)].copy()

len(df_segment)
```

```
1540
```

What is e-Car doing in this segment?

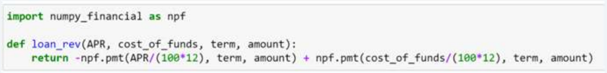


e-Car mostly prices loans between 6% and 6.5% in this segment

How do we determine e-Car's revenue in each segment? We want to check whether the segment they use is the best one…

Revenue per *accepted* quote

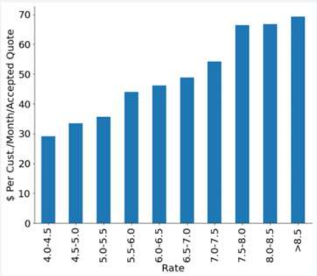
- Revenue per client is the money received from the client minus the cost of funds
- Both can be calculated using the `numpy_financial.pmt` function, equivalent to the Excel `PMT` function



```
import numpy_financial as npf

def loan_rev(APR, cost_of_funds, term, amount):
    return -npf.pmt(APR/(100*12), term, amount) + npf.pmt(cost_of_funds/(100*12), term, amount)
```

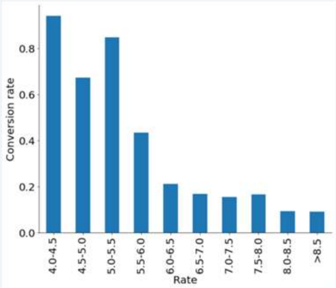
Important note: the “cost of funds” includes the probability of default

Revenue per *accepted* quote

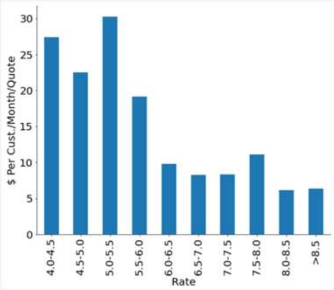


Why not just price at the highest rate all the time? It gives the highest revenue…

Conversion rate



Revenue per quote



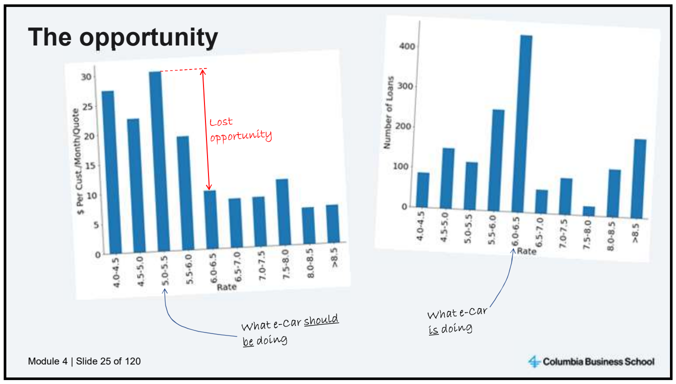
Revenue per quote

=

Revenue per *accepted* quote

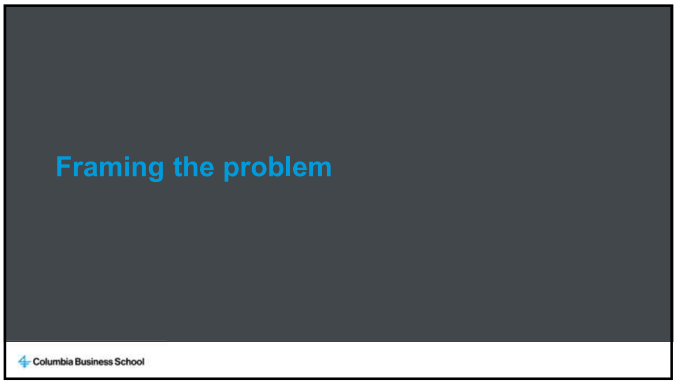
×

Acceptance rate



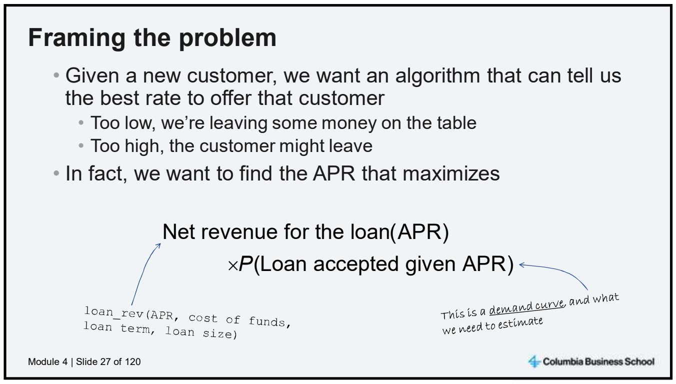
The opportunity

Columbia Business School



Framing the problem

Columbia Business School



Framing the problem

- Given a new customer, we want an algorithm that can tell us the best rate to offer that customer
  - Too low, we're leaving some money on the table
  - Too high, the customer might leave
- In fact, we want to find the APR that maximizes

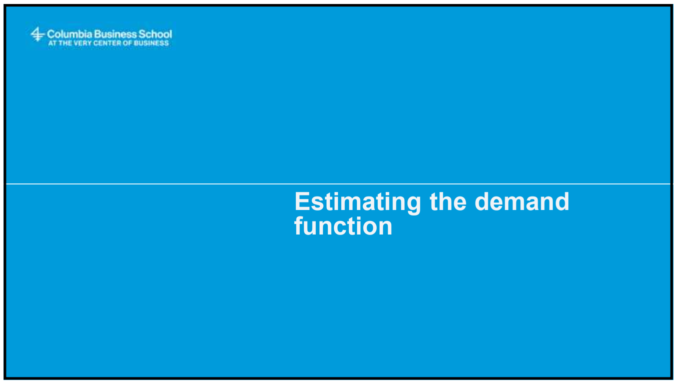
Net revenue for the loan(APR)

$\times P$(Loan accepted given APR)

```
loan_rev(APR, cost of funds,
loan term, loan size)
```

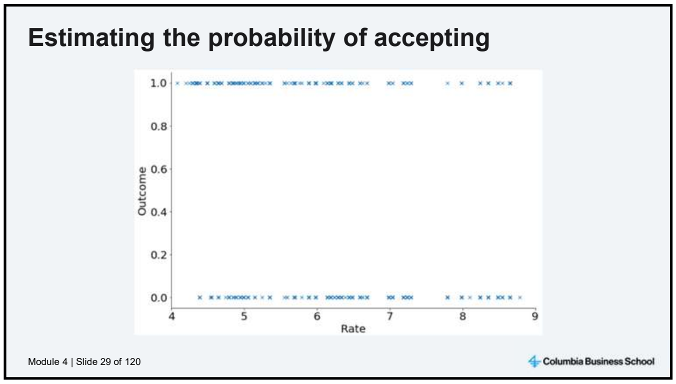
This is a demand curve, and what we need to estimate

Columbia Business School



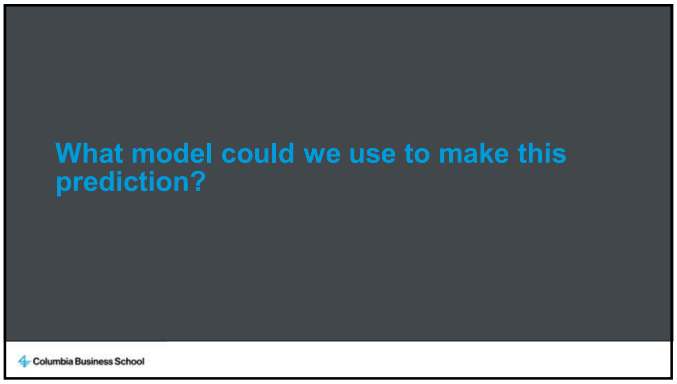
Columbia Business School
AT THE VERY CENTER OF BUSINESS

Estimating the demand function



Estimating the probability of accepting

Columbia Business School



What model could we use to make this prediction?

Columbia Business School

Estimating the probability of accepting

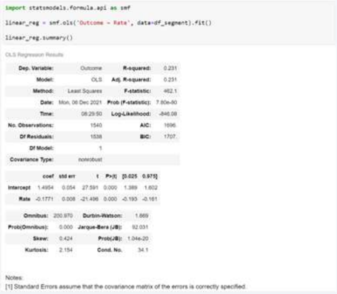
Linear regression

$$\text{Outcome}_j = a + b\,\text{APR}_j + \varepsilon_j$$

Where APR$_j$ is the APR quoted to customer j, and:

$$\text{Outcome}_j = \begin{cases} 1 & \text{if customer } j \text{ accepted the quote} \\ 0 & \text{otherwise} \end{cases}$$

Linear regression



```
import statsmodels.formula.api as smf

linear_reg = smf.ols('Outcome ~ Rate', data=df_segment).fit()

linear_reg.summary()
```

OLS Regression Results

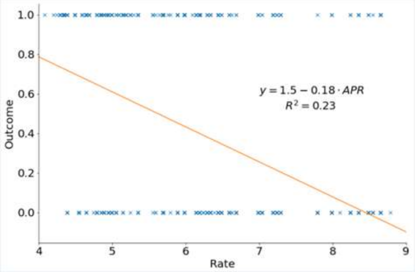
Dep. Variable:	Outcome	**R-squared:**	0.231
Model:	OLS	**Adj. R-squared:**	0.231
Method:	Least Squares	**F-statistic:**	462.1
Date:	Mon, 06 Dec 2021	**Prob (F-statistic):**	7.80e-90
Time:	08:29:50	**Log-Likelihood:**	-846.08
No. Observations:	1540	**AIC:**	1696.
Df Residuals:	1538	**BIC:**	1707.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	1.4954	0.054	27.591	0.000	1.389	1.602
Rate	-0.1771	0.008	-21.496	0.000	-0.193	-0.161

Omnibus:	200.970	**Durbin-Watson:**	1.869
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	92.031
Skew:	0.424	**Prob(JB):**	1.04e-20
Kurtosis:	2.154	**Cond. No.**	34.1

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Linear regression



Does this seem reasonable?

Issues with linear regression

- Predictions need to be probabilities (between 0 and 1) but linear regression might predict numbers smaller than 0 or larger than 1
- The “normal errors”/“errors independent of *x*” assumptions of linear regression are violated

Logistic regression

Logistic regression

- **Logistic regression** is a technique for fitting a curve to data in which the **dependent variable is binary**
- Applications
  - Response to a medical treatment: worked (coded as 1) or did not work (coded as 0)
  - Customized pricing: bought (1) or not (0)
  - Sponsored search: user clicked (1) or not (0)

Logistic regression

$$P(\text{Accepting given APR}) = \text{Logit}^{-1}(a + b \cdot \text{APR})$$

$$\text{Logit}(p) = \ln\left(\frac{p}{1-p}\right) \qquad \text{Logit}^{-1}(w) = \frac{1}{1+e^{-w}}$$

`np.exp(-w)`

- The Logit function *squeezes* the results of the linear regression to the range [0, 1]
- The responses are always between 0 and 1
- Allows for flexible nonlinear shapes
- Parameters *a* and *b* need to be chosen to fit the data "best"; more on that later

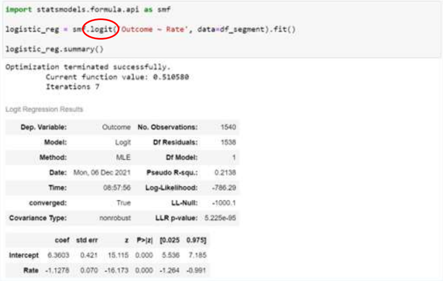
Differing conventions

Note that

$$\begin{aligned}\text{Logit}^{-1}(w) &= \frac{1}{1+e^{-w}} \\ &= \frac{1}{1+e^{-w}} \times 1 \\ &= \frac{1}{1+e^{-w}} \times \frac{e^{w}}{e^{w}} \\ &= \frac{e^{w}}{1+e^{w}}\end{aligned}$$

Some texts you will read will use the second form of this function – they are identical.

Logistic regression in Python



```
import statsmodels.formula.api as smf

logistic_reg = smf.logit('Outcome ~ Rate', data=df_segment).fit()

logistic_reg.summary()
```

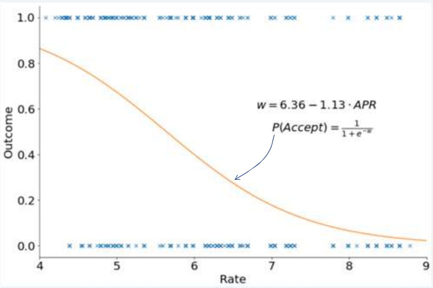
```
Optimization terminated successfully.
         Current function value: 0.510580
         Iterations 7
```

Logit Regression Results

Dep. Variable:	Outcome	No. Observations:	1540
Model:	Logit	Df Residuals:	1538
Method:	MLE	Df Model:	1
Date:	Mon, 06 Dec 2021	Pseudo R-squ.:	0.2138
Time:	08:57:56	Log-Likelihood:	-786.29
converged:	True	LL-Null:	-1000.1
Covariance Type:	nonrobust	LLR p-value:	5.225e-95

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	6.3603	0.421	15.115	0.000	5.536	7.185
Rate	-1.1278	0.070	-16.173	0.000	-1.264	-0.991

Logistic regression in Python

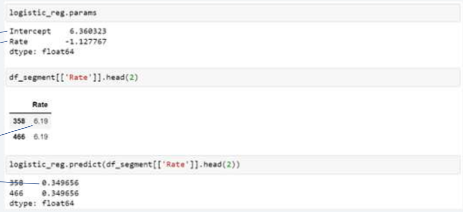


Understanding logistic regression

$$\frac{1}{1+e^{-(a+b\cdot\text{APR})}}$$

$$= \frac{1}{1+e^{-(6.3603-1.1278\times 6.19)}}$$

$$= 0.3496$$



```
logistic_reg.params
```

```
Intercept    6.360323
Rate        -1.127767
dtype: float64
```

```
df_segment[['Rate']].head(2)
```

	Rate
358	6.19
466	6.19

```
logistic_reg.predict(df_segment[['Rate']].head(2))
```

```
358    0.349656
466    0.349656
dtype: float64
```

Interpreting coefficients

- Coefficients are harder to interpret in a logistic regression
- If *w* goes from 1 to 2, it has a different impact on the predicted probability than if it goes from 10 to 11
- The **sign** of the coefficient, however, can easily be interpreted; the **negative coefficient** here means that **as the APR increases, the probability of acceptance goes down**

A deeper dive into logistic regression (optional)

Where does logistic regression come from?

- There are many ways to motivate the exact form of logistic regression
- Many of them are summarized surprisingly well at https://en.wikipedia.org/wiki/Logistic_regression
- We're going to focus on one specific interpretation that is particularly well-suited to the problem at hand

A latent-variable model of logistic regression

- The theory of **discrete choice models** tries to explain how consumers make purchasing decisions
- The idea is that when we decide to buy something, we weigh up the pros and cons
  - Getting the item is a pro (positive utility)
  - Having to pay for it is a con (negative utility) – the more expensive, the worse (more negative) the con
  - There might be some randomness (positive or negative) based on who the consumer is exactly
- If the total utility is positive, the consumer gets *more* out of buying the item than not and buys it. Otherwise, they don't

A latent-variable model of logistic regression

The utility a certain consumer would get out of accepting the car loan

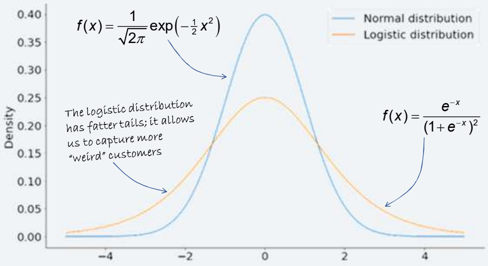
The disutility from having to pay for the loan (b will be negative). The more negative b is, the more customers care about price

$$\text{Utility}_j = a + \overbrace{b\,\text{APR}_j} - \varepsilon_j$$

The base utility everyone gets from getting a car loan

The random component of the customer's utility; across the whole population, $E[\varepsilon_j] = 0$

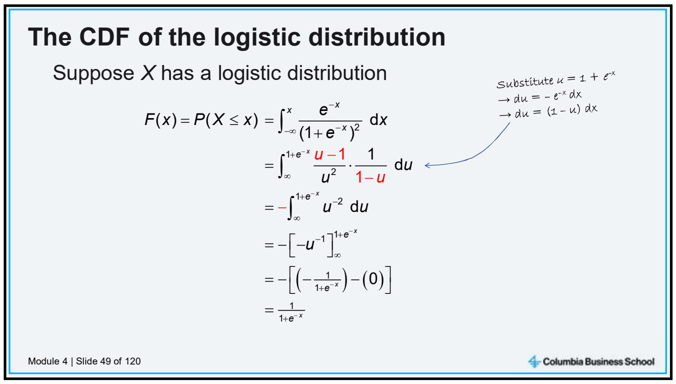
The distribution of ε_j



$$f(x) = \frac{1}{\sqrt{2\pi}}\exp\left(-\tfrac{1}{2}x^2\right)$$

The logistic distribution has fatter tails; it allows us to capture more "weird" customers

$$f(x) = \frac{e^{-x}}{(1+e^{-x})^2}$$

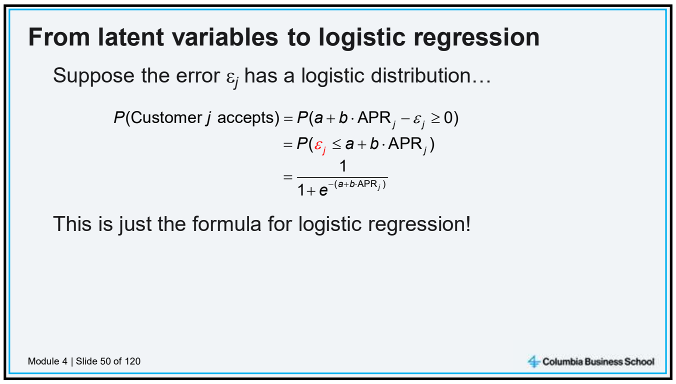


The CDF of the logistic distribution

Suppose X has a logistic distribution

$$F(x) = P(X \le x) = \int_{-\infty}^{x} \frac{e^{-x}}{(1+e^{-x})^2}\ dx$$
$$= \int_{\infty}^{1+e^{-x}} \frac{u-1}{u^2} \cdot \frac{1}{1-u}\ du$$
$$= -\int_{\infty}^{1+e^{-x}} u^{-2}\ du$$
$$= -\left[-u^{-1}\right]_{\infty}^{1+e^{-x}}$$
$$= -\left[\left(-\tfrac{1}{1+e^{-x}}\right) - (0)\right]$$
$$= \tfrac{1}{1+e^{-x}}$$

Substitute $u = 1 + e^{-x}$
$\rightarrow du = -e^{-x}\,dx$
$\rightarrow du = (1-u)\,dx$

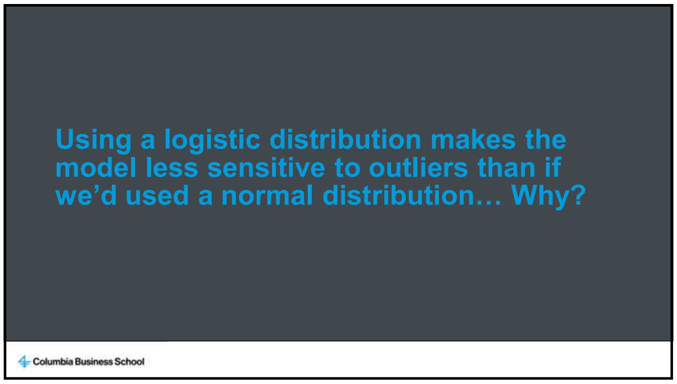


From latent variables to logistic regression

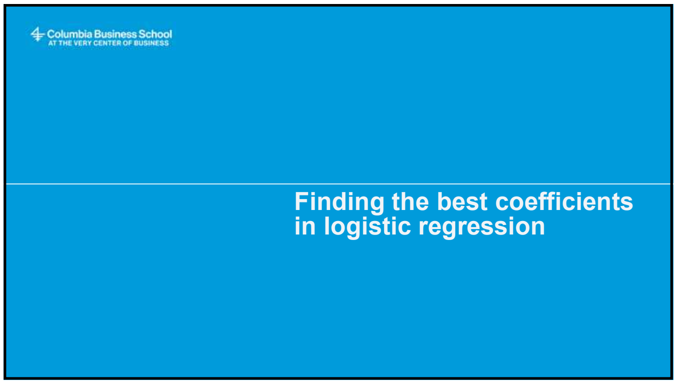
Suppose the error ε_j has a logistic distribution…

$$P(\text{Customer } j \text{ accepts}) = P(a + b \cdot \text{APR}_j - \varepsilon_j \ge 0)$$
$$= P(\varepsilon_j \le a + b \cdot \text{APR}_j)$$
$$= \frac{1}{1 + e^{-(a + b \cdot \text{APR}_j)}}$$

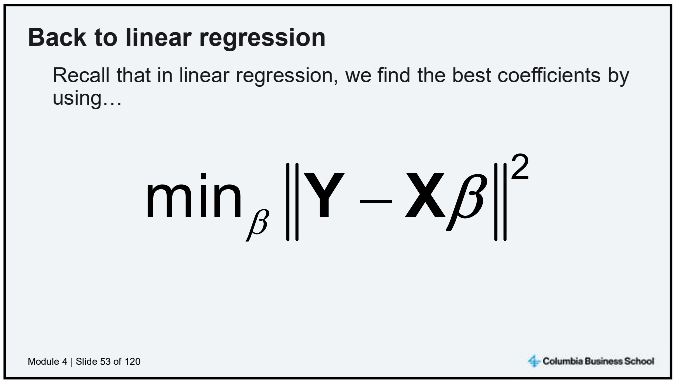
This is just the formula for logistic regression!



Using a logistic distribution makes the model less sensitive to outliers than if we'd used a normal distribution… Why?



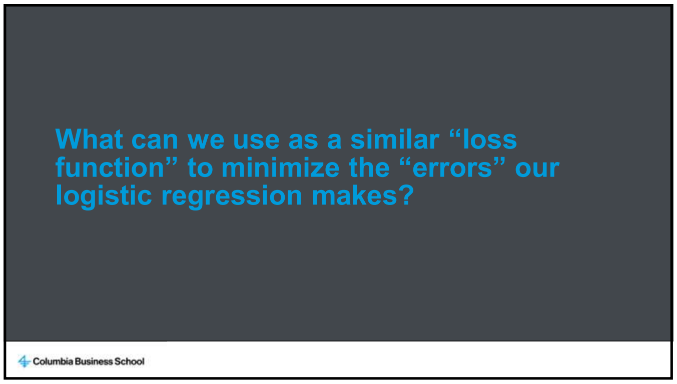
Finding the best coefficients in logistic regression



Back to linear regression

Recall that in linear regression, we find the best coefficients by using…

$$\min_{\beta} \left\| \mathbf{Y} - \mathbf{X}\beta \right\|^2$$



What can we use as a similar "loss function" to minimize the "errors" our logistic regression makes?

An example

APR$_j$	Outcome$_j$
2.1	1
2.2	0
1.3	1

What if a = 2 and b = –3

$$\begin{aligned} P(\text{Data}) &= P(\text{APR 2.1 accepts}) \times P(\text{APR 2.2 doesn't accept}) \times P(\text{APR 1.3 accepts}) \\ &= \frac{1}{1+e^{-(2-3\times 2.1)}} \times \left(1-\frac{1}{1+e^{-(2-3\times 2.2)}}\right) \times \frac{1}{1+e^{-(2-3\times 1.3)}} \\ &= 0.0134 \times 0.9900 \times 0.1301 \\ &= 0.0017 \end{aligned}$$

Now suppose a = 1 and b = –2

$$\begin{aligned} P(\text{Data}) &= P(\text{APR 2.1 accepts}) \times P(\text{APR 2.2 doesn't accept}) \times P(\text{APR 1.3 accepts}) \\ &= \frac{1}{1+e^{-(1-2\times 2.1)}} \times \left(1-\frac{1}{1+e^{-(1-2\times 2.2)}}\right) \times \frac{1}{1+e^{-(1-2\times 1.3)}} \\ &= 0.0392 \times 0.0.9677 \times 0.1680 \\ &= 0.0064 \end{aligned}$$

The likelihood in logistic regression

Recall that logistic regression assumes

$$P(j \text{ Accepting given APR}) = \frac{1}{1+e^{-(a+b\cdot\text{APR}_j)}}$$

And therefore

$$\begin{aligned} P(j \text{ NOT Accepting given APR}) &= 1-\frac{1}{1+e^{-(a+b\cdot\text{APR}_j)}} \\ &= \frac{e^{-(a+b\cdot\text{APR}_j)}}{1+e^{-(a+b\cdot\text{APR}_j)}} \end{aligned}$$

Using these formulas, we can calculate the **likelihood** of the data we're observing given any value of a and b.

More generally

Suppose we have N datapoints, with rates APR$_j$ and outcomes y_j (equal to 1 if the loan is accepted, and 0 otherwise)

$$\begin{aligned} P(\text{Data}) &= \prod_{j=1}^{N} \left[P(\text{APR}_j \text{ accepts})\right]^{y_j} \left[P(\text{APR}_j \text{ rejects})\right]^{1-y_j} \\ &= \prod_{j=1}^{N} \left(\frac{1}{1+e^{-(a+b\cdot\text{APR}_j)}}\right)^{y_j} \left(\frac{e^{-(a+b\cdot\text{APR}_j)}}{1+e^{-(a+b\cdot\text{APR}_j)}}\right)^{1-y_j} \end{aligned}$$

Logistic regression finds the best a and b by **maximizing** this likelihood

Can you think of any issues trying to maximize this expression?

The log-likelihood

The likelihood can become very small. Instead, therefore, we usually use the log-likelihood:

$$\begin{aligned} \log P(\text{Data}) &= \log \prod_{j=1}^{N} \left(\frac{1}{1+e^{-(a+b\cdot\text{APR}_j)}}\right)^{y_j} \left(\frac{e^{-(a+b\cdot\text{APR}_j)}}{1+e^{-(a+b\cdot\text{APR}_j)}}\right)^{1-y_j} \\ &= \sum_{j=1}^{N} y_j \log\left(\frac{1}{1+e^{-(a+b\cdot\text{APR}_j)}}\right) + (1-y_j)\log\left(\frac{e^{-(a+b\cdot\text{APR}_j)}}{1+e^{-(a+b\cdot\text{APR}_j)}}\right) \\ &= \sum_{j=1}^{N} y_j \log\left(\frac{e^{(a+b\cdot\text{APR}_j)}}{1+e^{(a+b\cdot\text{APR}_j)}}\right) + (1-y_j)\log\left(\frac{1}{1+e^{(a+b\cdot\text{APR}_j)}}\right) \\ &= \sum_{j=1}^{N} y_j \left(a+b\cdot\text{APR}_j\right) - \log\left(1+e^{a+b\cdot\text{APR}_j}\right) \end{aligned}$$

Multiply the top and bottom of each fraction in the previous line by $e^{a+b\text{APR}}$

Work through the logarithms

CE 4

Finding the best coefficients

- The best a and b can be found by **maximizing** this log likelihood, or, equivalently, **minimizing** the **negative log likelihood**.
- This negative log-likelihood is also called the **loss**.

$$\min_{a,b}\left[-\log P(\text{Data})\right] = \min_{a,b}\left[\sum_{j=1}^{N} \log\left(1+e^{a+b\cdot\text{APR}_j}\right) - y_j\left(a+b\cdot\text{APR}_j\right)\right]$$

The concept of minimizing a loss function is ubiquitous in all of AI and machine learning, from linear regression to logistic regression. The log-likelihood is often the basis for this loss function

Columbia Business School

Finding the best coefficients

$$\min_{a,b}\left[-\log P(\text{Data})\right]=\min_{a,b}\left[\sum_{j=1}^{N}\log\left(1+e^{a+b\cdot \text{APR}_j}\right)-y_j\left(a+b\cdot \text{APR}_j\right)\right]$$

To find the minimum, let’s find the derivative of this expression

$$\frac{\partial}{\partial a}\left[-\log P(\text{Data})\right]=\sum_{j=1}^{N}\frac{e^{a+b\cdot \text{APR}_j}}{1+e^{a+b\cdot \text{APR}_j}}-y_j$$

$$\frac{\partial}{\partial b}\left[-\log P(\text{Data})\right]=\sum_{j=1}^{N}\frac{\text{APR}_j\cdot e^{a+b\cdot \text{APR}_j}}{1+e^{a+b\cdot \text{APR}_j}}-y_j\text{APR}_j$$

$$=\sum_{j=1}^{N}\left[\frac{e^{a+b\cdot \text{APR}_j}}{1+e^{a+b\cdot \text{APR}_j}}-y_j\right]\text{APR}_j$$

Columbia Business School

Unlike in linear regression, it is impossible to solve these equations exactly

Columbia Business School

Columbia Business School
AT THE VERY CENTER OF BUSINESS

Gradient descent

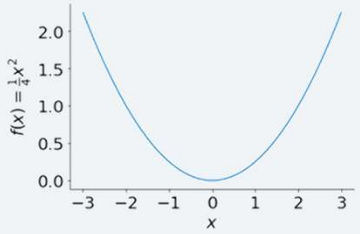
Gradient descent

- Gradient descent is a very general algorithm that can be used to solve these kinds of optimization problems
- The idea is to start with some random values for the parameters…
- …and then move in the direction of the gradient
- Let’s look at an example with an easy function

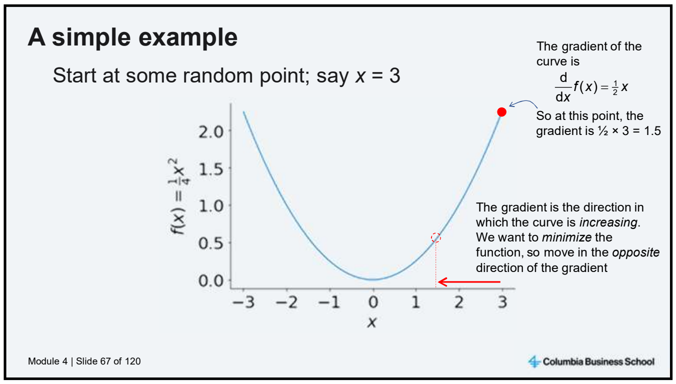
Columbia Business School

A simple example

Suppose we are trying to find the minimum of $f(x)$ = $0.25x^2$



Columbia Business School



A simple example

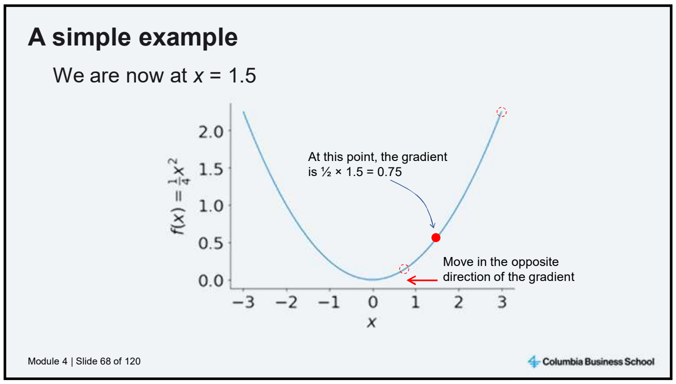
Start at some random point; say $x = 3$

The gradient of the curve is

$$\frac{\mathrm{d}}{\mathrm{d}x} f(x) = \tfrac{1}{2}x$$

So at this point, the gradient is ½ × 3 = 1.5

The gradient is the direction in which the curve is *increasing*. We want to *minimize* the function, so move in the *opposite* direction of the gradient

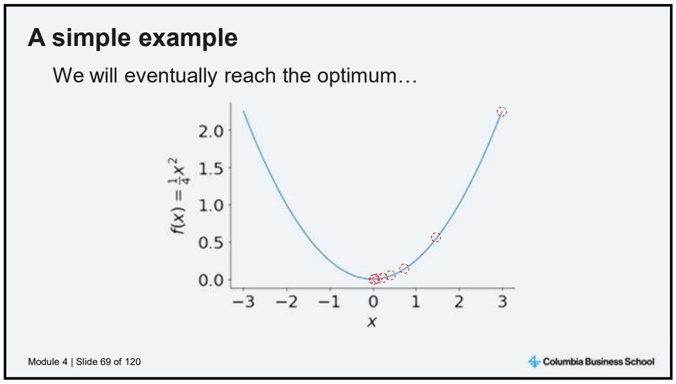


A simple example

We are now at $x = 1.5$

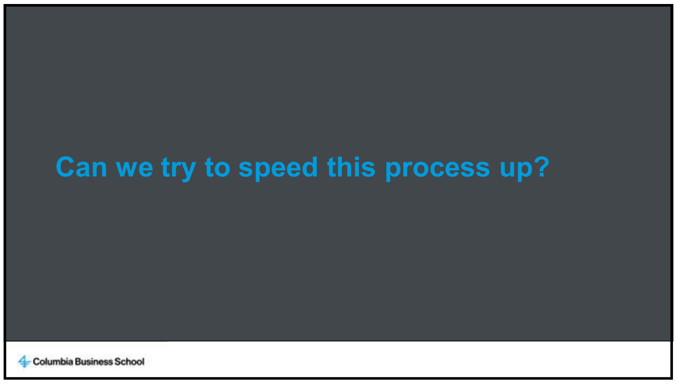
At this point, the gradient is ½ × 1.5 = 0.75

Move in the opposite direction of the gradient

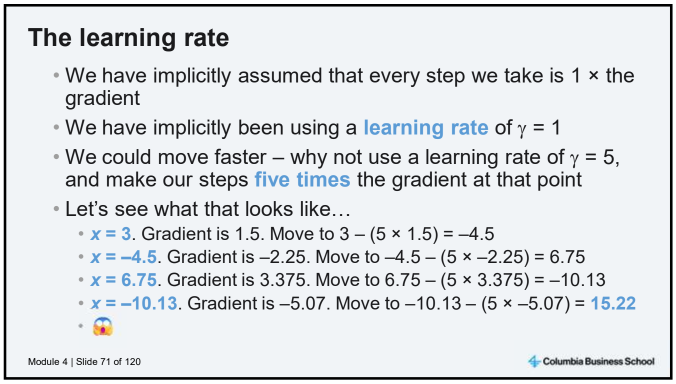


A simple example

We will eventually reach the optimum…

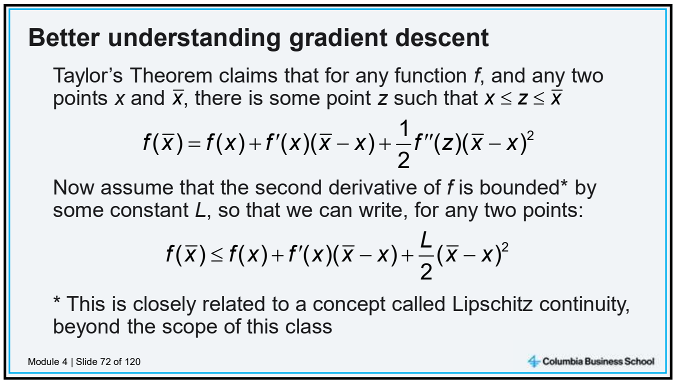


Can we try to speed this process up?



The learning rate

- We have implicitly assumed that every step we take is 1 × the gradient
- We have implicitly been using a **learning rate** of $\gamma = 1$
- We could move faster – why not use a learning rate of $\gamma = 5$, and make our steps **five times** the gradient at that point
- Let's see what that looks like…
  - ***x* = 3**. Gradient is 1.5. Move to 3 – (5 × 1.5) = –4.5
  - ***x* = –4.5**. Gradient is –2.25. Move to –4.5 – (5 × –2.25) = 6.75
  - ***x* = 6.75**. Gradient is 3.375. Move to 6.75 – (5 × 3.375) = –10.13
  - ***x* = –10.13**. Gradient is –5.07. Move to –10.13 – (5 × –5.07) = **15.22**
  - 😱



Better understanding gradient descent

Taylor's Theorem claims that for any function f, and any two points x and $\bar{x}$, there is some point z such that $x \le z \le \bar{x}$

$$f(\bar{x}) = f(x) + f'(x)(\bar{x} - x) + \frac{1}{2} f''(z)(\bar{x} - x)^2$$

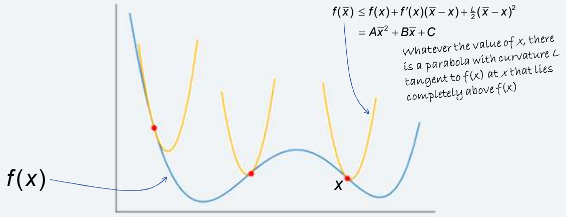
Now assume that the second derivative of f is bounded* by some constant L, so that we can write, for any two points:

$$f(\bar{x}) \le f(x) + f'(x)(\bar{x} - x) + \frac{L}{2}(\bar{x} - x)^2$$

* This is closely related to a concept called Lipschitz continuity, beyond the scope of this class

Better understanding gradient descent

There is a simple graphical interpretation of this inequality



$$f(\bar{x}) \le f(x) + f'(x)(\bar{x} - x) + \tfrac{L}{2}(\bar{x} - x)^2$$
$$= A\bar{x}^2 + B\bar{x} + C$$

Whatever the value of x, there is a parabola with curvature L tangent to $f(x)$ at x that lies completely above $f(x)$

Key insight to analyze gradient descent: we take a gradient descent step *on the parabola* instead of taking a step on the function itself

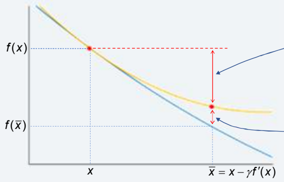
Gradient descent step

- Let x be the current step in the algorithm
- Let $\bar{x} = x - \gamma f'(x)$ be the *next* step in the algorithm
- What is the value of the parabola at that new point?

$$f(\bar{x}) \le f(x) + f'(x)(\bar{x} - x) + \frac{L}{2}(\bar{x} - x)^2$$
$$= f(x) - \gamma[f'(x)]^2 + \frac{L}{2}\gamma^2[f'(x)]^2$$
$$= f(x) - \left(1 - \frac{L\gamma}{2}\right)\gamma[f'(x)]^2$$

Gradient descent step

$$f(\bar{x}) \le f(x) - \left(1 - \frac{L\gamma}{2}\right)\gamma[f'(x)]^2$$



Provided that $\gamma < 2/L$, this term will be negative, and so the new point on the parabola will be *less than* $f(x)$, the value of the function at the previous point

But we know the parabola lies above the curve, so this means the new point on the function must be below the previous one. This is true of every step, and so we eventually converge to a minimum…

Increasing γ can make the algorithm go faster, but if it's too large, the algorithm isn't guaranteed to converge. We need to make sure $\gamma < 2/L$, but we don't necessarily know *L*

Gradient descent – going beyond the basics

- Gradient descent is ubiquitous in all of machine learning – from logistic regression to deep neural nets
- Gradient descent works best for **convex optimization problems** – but it can still help with nonconvex problems
- The choice of learning rate is important – choosing the wrong learning rate can mean the algorithm doesn't converge
- In practice it is often helpful to use an **adaptive learning rate**, which change as the algorithm progresses
- In some cases, the gradient can't be calculated analytically – gradient descent can use an **empirical gradient** based on data in those cases
- We will later see a version of the algorithm called **stochastic gradient descent** that can work with small chunks of data at a time
- Gradient descent can get **very slow**, especially in high dimensions – there are many, more advanced techniques that perform much better

Gradient descent for logistic regression

The gradient of the loss in logistic regression

Recall that for logistic regression, the loss is given by

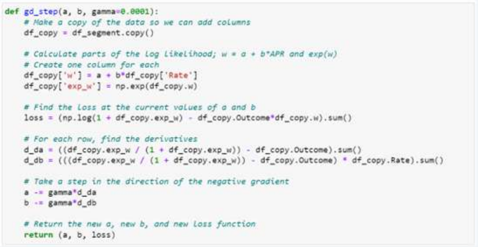
$$-\log P(\text{Data}) = \sum_{j=1}^{N} \log\left(1+e^{a+b\cdot\text{APR}_j}\right) - y_j\left(a+b\cdot\text{APR}_j\right)$$

And that

$$\frac{\partial}{\partial a}\left[-\log P(\text{Data})\right] = \sum_{j=1}^{N} \frac{e^{a+b\cdot\text{APR}_j}}{1+e^{a+b\cdot\text{APR}_j}} - y_j$$

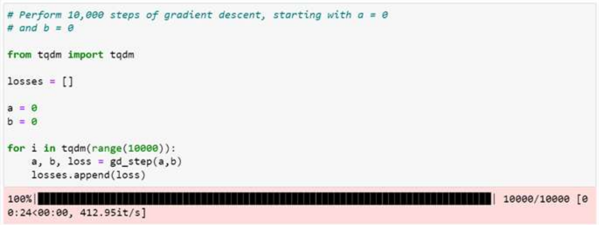
$$\frac{\partial}{\partial b}\left[-\log P(\text{Data})\right] = \sum_{j=1}^{N} \left[\frac{e^{a+b\cdot\text{APR}_j}}{1+e^{a+b\cdot\text{APR}_j}} - y_j\right]\text{APR}_j$$

Gradient descent step



```
def gd_step(a, b, gamma=0.0001):
    # Make a copy of the data so we can add columns
    df_copy = df_segment.copy()

    # Calculate parts of the log likelihood; w = a + b*APR and exp(w)
    # Create one column for each
    df_copy['w'] = a + b*df_copy['Rate']
    df_copy['exp_w'] = np.exp(df_copy.w)

    # Find the loss at the current values of a and b
    loss = (np.log(1 + df_copy.exp_w) - df_copy.Outcome*df_copy.w).sum()

    # For each row, find the derivatives
    d_da = ((df_copy.exp_w / (1 + df_copy.exp_w)) - df_copy.Outcome).sum()
    d_db = (((df_copy.exp_w / (1 + df_copy.exp_w)) - df_copy.Outcome) * df_copy.Rate).sum()

    # Take a step in the direction of the negative gradient
    a -= gamma*d_da
    b -= gamma*d_db

    # Return the new a, new b, and new loss function
    return (a, b, loss)
```

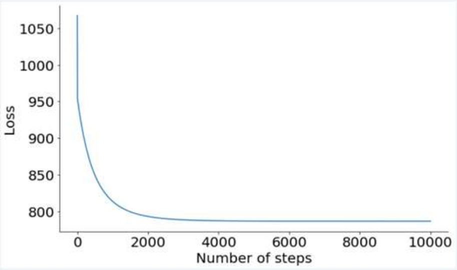
Carrying out gradient descent



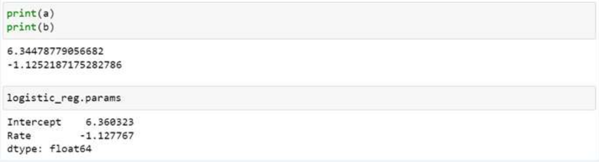
```
# Perform 10,000 steps of gradient descent, starting with a = 0
# and b = 0

from tqdm import tqdm

losses = []

a = 0
b = 0

for i in tqdm(range(10000)):
    a, b, loss = gd_step(a,b)
    losses.append(loss)
```

```
100%|████████████████████████████████████████████████████████████| 10000/10000 [0
0:24<00:00, 412.95it/s]
```

Carrying out gradient descent



Carrying out gradient descent



```
print(a)
print(b)
```

```
6.34478779056682
-1.1252187175282786
```

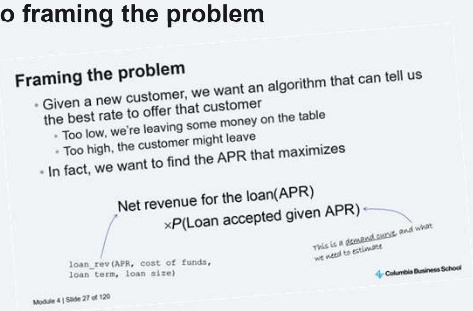
```
logistic_reg.params
```

```
Intercept    6.360323
Rate        -1.127767
dtype: float64
```

Columbia Business School
AT THE VERY CENTER OF BUSINESS

Back to Nomis

Back to framing the problem



Framing the problem

- Given a new customer, we want an algorithm that can tell us the best rate to offer that customer
  - Too low, we're leaving some money on the table
  - Too high, the customer might leave
- In fact, we want to find the APR that maximizes

Net revenue for the loan(APR)
×*P*(Loan accepted given APR)

loan_rev(APR, cost of funds, loan term, loan size)

This is a demand curve, and what we need to estimate

Module 4 | Slide 27 of 120

Columbia Business School

Columbia Business School

We now have a way to estimate the probability a loan will be accepted! How can we use this to get to the best APR?

Columbia Business School

Getting to the best APR

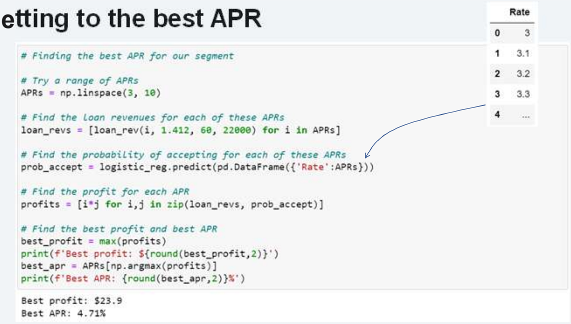
- Suppose a customer in our reduced segment (the one we've been working with) arrives
- The size of the loan is $22K, and the cost of funds is 1.412%
- What APR should we offer this person?
  - On the one hand, we want to maximize the price we can get…
  - …on the other, we want to maximize the number of customers who accept our offer
- In fact, we want

$$\max_{APR}[\text{loan\_rev}(APR,1.412,60,22000)\times P(\text{Accept given APR})]$$

$$\max_{APR}\left[\text{loan\_rev}(APR,1.412,60,22000)\times\frac{1}{1+e^{-(a+b\cdot APR)}}\right]$$

Columbia Business School

Getting to the best APR



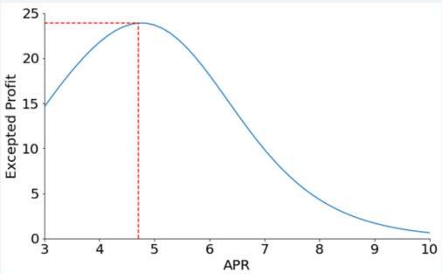
	Rate
0	3
1	3.1
2	3.2
3	3.3
4	...

```
# Finding the best APR for our segment

# Try a range of APRs
APRs = np.linspace(3, 10)

# Find the loan revenues for each of these APRs
loan_revs = [loan_rev(i, 1.412, 60, 22000) for i in APRs]

# Find the probability of accepting for each of these APRs
prob_accept = logistic_reg.predict(pd.DataFrame({'Rate':APRs}))

# Find the profit for each APR
profits = [i*j for i,j in zip(loan_revs, prob_accept)]

# Find the best profit and best APR
best_profit = max(profits)
print(f'Best profit: ${round(best_profit,2)}')
best_apr = APRs[np.argmax(profits)]
print(f'Best APR: {round(best_apr,2)}%')
```

```
Best profit: $23.9
Best APR: 4.71%
```

Columbia Business School

Getting to the best APR

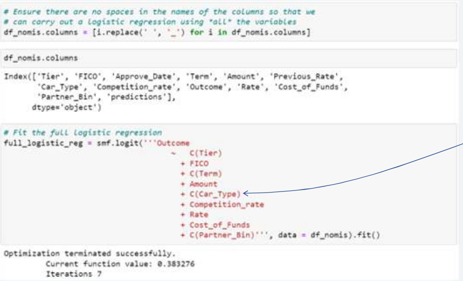


Columbia Business School

Multivariate logistic regression

Is there a way to fit a logistic regression with *many* variables, just like a multivariate linear regression?

Multivariate logistic regression



```
# Ensure there are no spaces in the names of the columns so that we
# can carry out a logistic regression using *all* the variables
df_nomis.columns = [i.replace(' ', '_') for i in df_nomis.columns]
```

```
df_nomis.columns
```

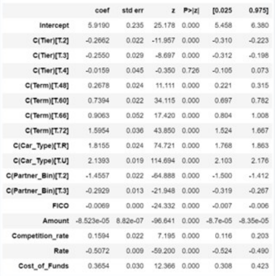
```
Index(['Tier', 'FICO', 'Approve_Date', 'Term', 'Amount', 'Previous_Rate',
       'Car_Type', 'Competition_rate', 'Outcome', 'Rate', 'Cost_of_Funds',
       'Partner_Bin', 'predictions'],
      dtype='object')
```

```
# Fit the full logistic regression
full_logistic_reg = smf.logit('''Outcome
                                    ~   C(Tier)
                                      + FICO
                                      + C(Term)
                                      + Amount
                                      + C(Car_Type)
                                      + Competition_rate
                                      + Rate
                                      + Cost_of_Funds
                                      + C(Partner_Bin)''', data = df_nomis).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.383276
         Iterations 7
```

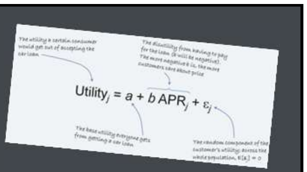
Create categorical variables

Multivariate logistic regression



	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	5.9190	0.235	25.178	0.000	5.458	6.380
C(Tier)[T.2]	-0.2662	0.022	-11.957	0.000	-0.310	-0.223
C(Tier)[T.3]	-0.2550	0.029	-8.697	0.000	-0.312	-0.198
C(Tier)[T.4]	-0.0159	0.045	-0.350	0.726	-0.105	0.073
C(Term)[T.48]	0.2678	0.024	11.111	0.000	0.221	0.315
C(Term)[T.60]	0.7394	0.022	34.115	0.000	0.697	0.782
C(Term)[T.66]	0.9063	0.052	17.420	0.000	0.804	1.008
C(Term)[T.72]	1.5954	0.036	43.850	0.000	1.524	1.667
C(Car_Type)[T.R]	1.8155	0.024	74.721	0.000	1.768	1.863
C(Car_Type)[T.U]	2.1393	0.019	114.694	0.000	2.103	2.176
C(Partner_Bin)[T.2]	-1.4557	0.022	-64.888	0.000	-1.500	-1.412
C(Partner_Bin)[T.3]	-0.2929	0.013	-21.948	0.000	-0.319	-0.267
FICO	-0.0069	0.000	-24.332	0.000	-0.007	-0.006
Amount	-8.523e-05	8.82e-07	-96.641	0.000	-8.7e-05	-8.35e-05
Competition_rate	0.1594	0.022	7.195	0.000	0.116	0.203
Rate	-0.5072	0.009	-59.200	0.000	-0.524	-0.490
Cost_of_Funds	0.3654	0.030	12.366	0.000	0.308	0.423

Notice we get access to all the same "goodies" as we did in linear regression – p-values, confidence intervals, etc…



$$\text{Utility}_j = a + \overbrace{b\,\text{APR}_j} + \varepsilon_j$$

The utility a certain consumer would get out of accepting the car loan
The disutility from having to pay for the loan (b will be negative). The more negative b is, the more customers care about price
The base utility everyone gets from getting a car loan
The random component of the customer's utility; across the whole population, E[ε_j] = 0

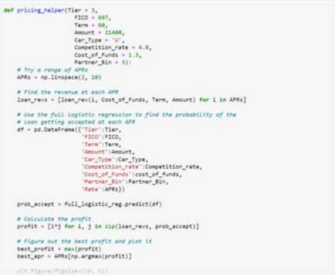
Think back to our motivation for logistic regression… How does multivariate logistic regression fit into this framework?

Finding the optimal rate

Hi there! I'd like a 60 month $22K loan to buy a used car

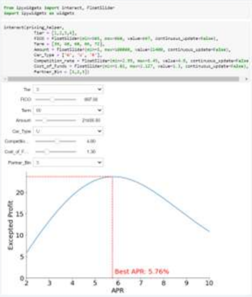
Sure, let me check what rate we can give you

Finding the optimal rate



```
def pricing_helper(Tier = 3,
                   FICO = 697,
                   Term = 60,
                   Amount = 21400,
                   Car_Type = 'U',
                   Competition_rate = 4.8,
                   Cost_of_Funds = 1.3,
                   Partner_Bin = 3):
    # Try a range of APRs
    APRs = np.linspace(2, 10)

    # Find the revenue at each APR
    loan_revs = [loan_rev(i, Cost_of_Funds, Term, Amount) for i in APRs]

    # Use the full logistic regression to find the probability of the
    # loan getting accepted at each APR
    df = pd.DataFrame({'Tier':Tier,
                       'FICO':FICO,
                       'Term':Term,
                       'Amount':Amount,
                       'Car_Type':Car_Type,
                       'Competition_rate':Competition_rate,
                       'Cost_of_Funds':cost_of_funds,
                       'Partner_Bin':Partner_Bin,
                       'Rate':APRs})

    prob_accept = full_logistic_reg.predict(df)

    # Calculate the profit
    profit = [i*j for i, j in zip(loan_revs, prob_accept)]

    # Figure out the best profit and plot it
    best_profit = max(profit)
    best_apr = APRs[np.argmax(profit)]

    plt.figure(figsize=(10, 6))
```

A decision support system



```
from ipywidgets import interact, FloatSlider
import ipywidgets as widgets
```

```
interact(pricing_helper,
         Tier = [1,2,3,4],
         FICO = FloatSlider(min=[illegible], max=850, value=697, continuous_update=False),
         Term = [36, 48, 60, 66, 72],
         Amount = FloatSlider(min=1, max=100000, value=21400, continuous_update=False),
         Car_Type = ['N', 'U', 'R'],
         Competition_rate = FloatSlider(min=2.99, max=6.45, value=4.8, continuous_update=False
         Cost_of_Funds = FloatSlider(min=1.01, max=2.127, value=1.3, continuous_update=False),
         Partner_Bin = [1,2,3])
```

Tier: 3
FICO: 697.00
Term: 60
Amount: 21400.00
Car_Type: U
Competitio...: 4.80
Cost_of_F...: 1.30
Partner_Bin: 3

Columbia Business School
AT THE VERY CENTER OF BUSINESS

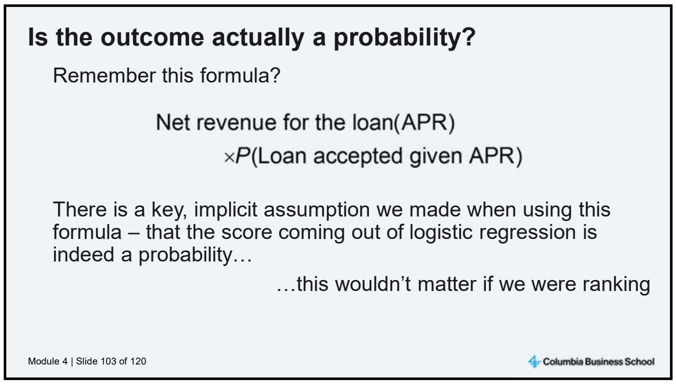
Model quality and calibration

Is there an equivalent of the R^2 for logistic regression? Something to tell us how “good” our logistic regression is?

Evaluating a logistic regression

- Binary models such as logistic regressions aren’t as simple to evaluate as continuous regression models
- There are several reasons for this – among them
  - The predicted outcome (a probability) is not of the same “type” as the true outcome (a 0/1 binary outcome)
  - There are many ways the outcome might be used; each will have different definition of a “good” model
    - **As a probability**; this is how we’re using it here
    - **To rank outcomes**; “we have a 100 loans sitting in our inbox, but only time to follow up on 30 of them; rank them by score and follow up on the top ones”
    - **To make a yes/no decision**: “a loan comes in and we think it might be fraud; use a model to predict the probability it’s fraud, and reject it if it’s above a certain threshold”

We will look at the second two applications in later lectures; let’s focus on the first



Is the outcome actually a probability?

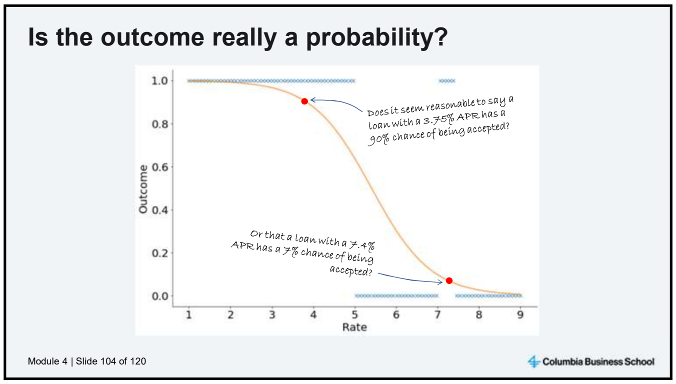
Remember this formula?

Net revenue for the loan(APR)
$\times P$(Loan accepted given APR)

There is a key, implicit assumption we made when using this formula – that the score coming out of logistic regression is indeed a probability…

…this wouldn't matter if we were ranking

Module 4 | Slide 103 of 120

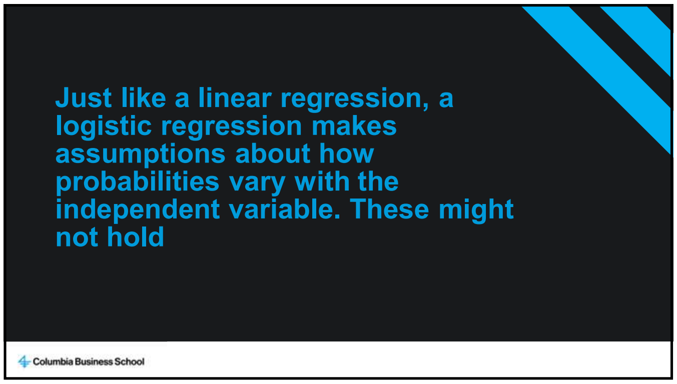


Is the outcome really a probability?

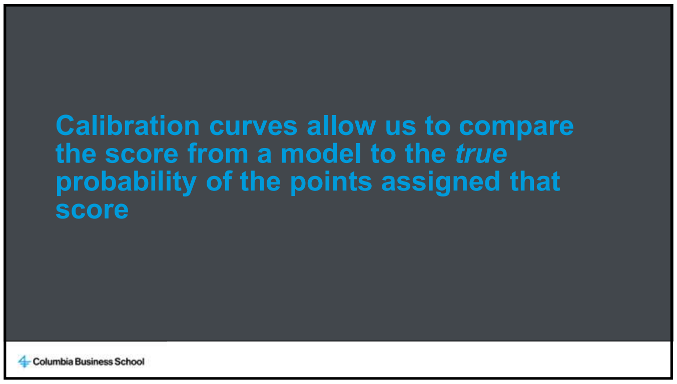
Does it seem reasonable to say a loan with a 3.75% APR has a 90% chance of being accepted?

Or that a loan with a 7.4% APR has a 7% chance of being accepted?

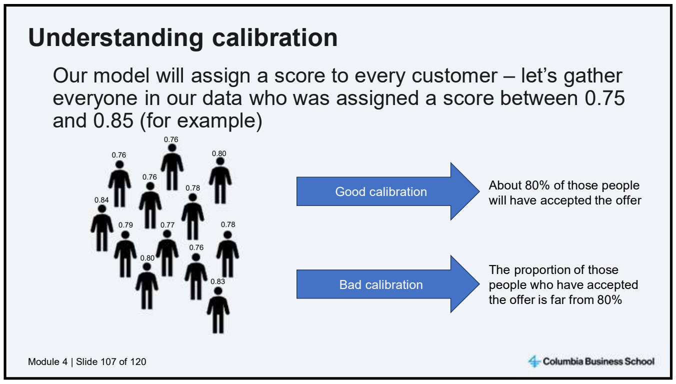
Module 4 | Slide 104 of 120



Just like a linear regression, a logistic regression makes assumptions about how probabilities vary with the independent variable. These might not hold



Calibration curves allow us to compare the score from a model to the *true* probability of the points assigned that score



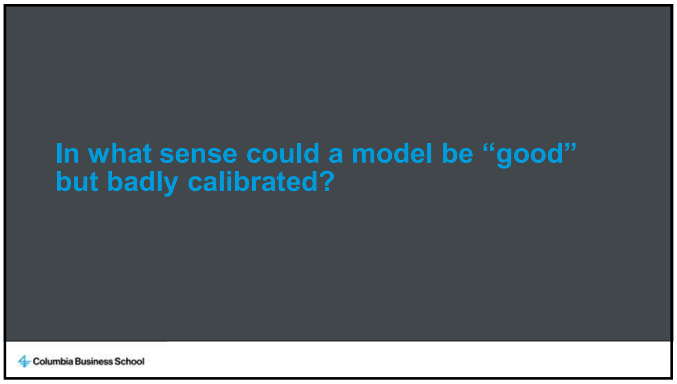
Understanding calibration

Our model will assign a score to every customer – let's gather everyone in our data who was assigned a score between 0.75 and 0.85 (for example)

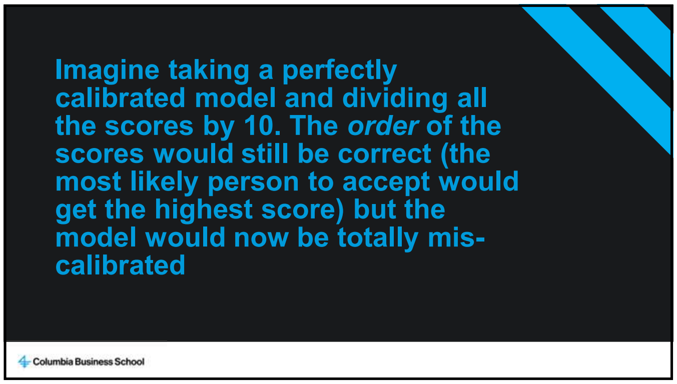
Good calibration → About 80% of those people will have accepted the offer

Bad calibration → The proportion of those people who have accepted the offer is far from 80%

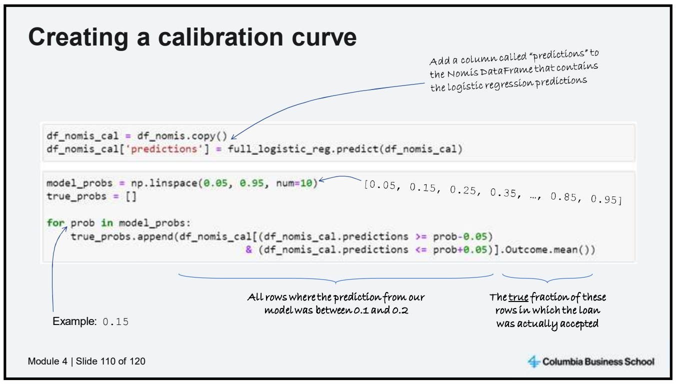
Module 4 | Slide 107 of 120



In what sense could a model be "good" but badly calibrated?



Imagine taking a perfectly calibrated model and dividing all the scores by 10. The *order* of the scores would still be correct (the most likely person to accept would get the highest score) but the model would now be totally mis-calibrated



Creating a calibration curve

Add a column called "predictions" to the Nomis DataFrame that contains the logistic regression predictions

```
df_nomis_cal = df_nomis.copy()
df_nomis_cal['predictions'] = full_logistic_reg.predict(df_nomis_cal)
```

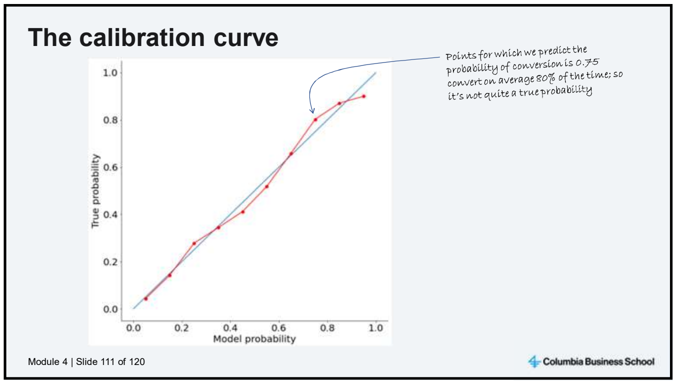
```
model_probs = np.linspace(0.05, 0.95, num=10)
true_probs = []

for prob in model_probs:
    true_probs.append(df_nomis_cal[(df_nomis_cal.predictions >= prob-0.05)
                                   & (df_nomis_cal.predictions <= prob+0.05)].Outcome.mean())
```

`[0.05, 0.15, 0.25, 0.35, …, 0.85, 0.95]`

All rows where the prediction from our model was between 0.1 and 0.2

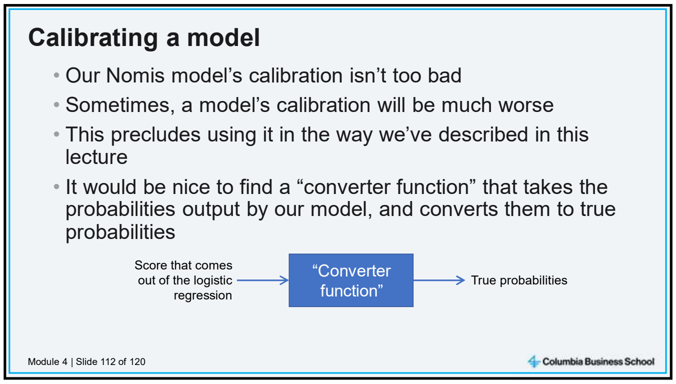
The true fraction of these rows in which the loan was actually accepted

Example: `0.15`



The calibration curve

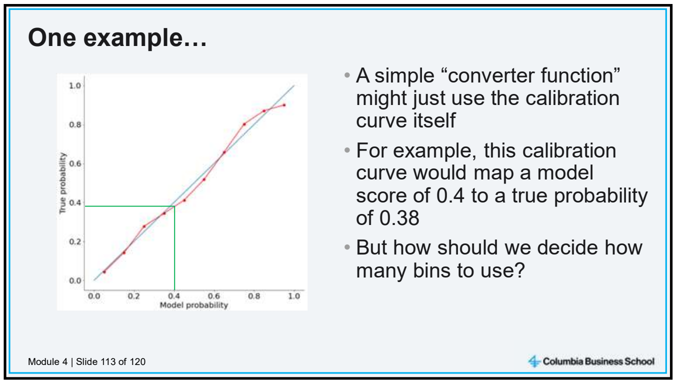
Points for which we predict the probability of conversion is 0.75 convert on average 80% of the time; so it's not quite a true probability



Calibrating a model

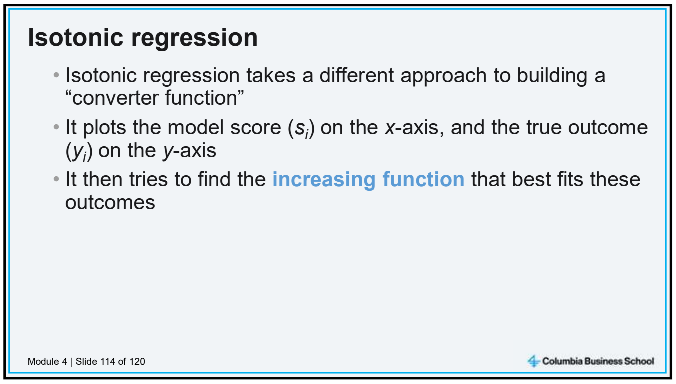
- Our Nomis model's calibration isn't too bad
- Sometimes, a model's calibration will be much worse
- This precludes using it in the way we've described in this lecture
- It would be nice to find a "converter function" that takes the probabilities output by our model, and converts them to true probabilities

Score that comes out of the logistic regression → "Converter function" → True probabilities



One example…

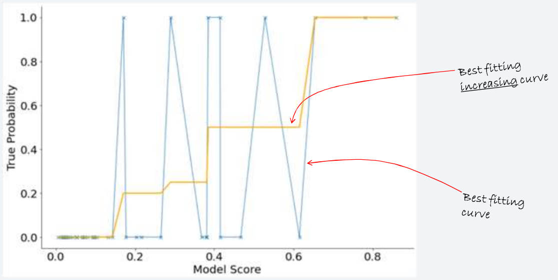
- A simple "converter function" might just use the calibration curve itself
- For example, this calibration curve would map a model score of 0.4 to a true probability of 0.38
- But how should we decide how many bins to use?



Isotonic regression

- Isotonic regression takes a different approach to building a "converter function"
- It plots the model score (s_i) on the x-axis, and the true outcome (y_i) on the y-axis
- It then tries to find the **increasing function** that best fits these outcomes

Isotonic regression (small sample of Nomis data)



Best fitting *increasing* curve

Best fitting curve

Isotonic regression

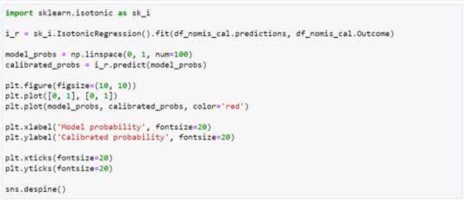
- Suppose we have N points with scores s_i and true outcomes y_i
- For each score s_i, Isotonic regression finds the best fitting "true probability" $z(s_i)$ that solves

$$\min \sum_{i=1}^{N} [y_i - z(s_i)]^2 \text{ such that } z(s_i) \le z(s_{i+1})$$

- z is our "converter function"
- This problem can be solved using the **pair-adjacent violators** algorithm, which I'll demo in class

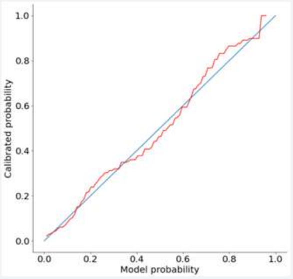
Isotonic regression in Python

The `sklearn` package can carry out Isotonic regression in Python



```
import sklearn.isotonic as sk_i

i_r = sk_i.IsotonicRegression().fit(df_nomis_cal.predictions, df_nomis_cal.Outcome)

model_probs = np.linspace(0, 1, num=100)
calibrated_probs = i_r.predict(model_probs)

plt.figure(figsize=(10, 10))
plt.plot([0, 1], [0, 1])
plt.plot(model_probs, calibrated_probs, color='red')

plt.xlabel('Model probability', fontsize=20)
plt.ylabel('Calibrated probability', fontsize=20)

plt.xticks(fontsize=20)
plt.yticks(fontsize=20)

sns.despine()
```

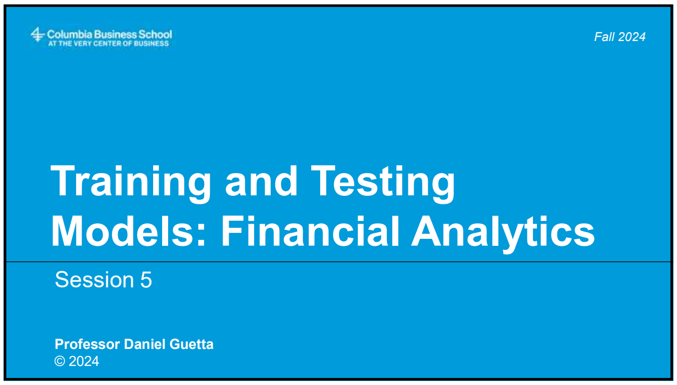
Isotonic regression on the Nomis data



Nomis Solutions today

Nomis solutions today

[redacted]



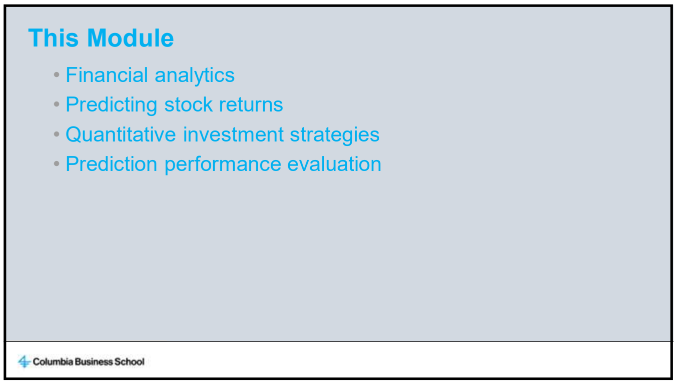
Columbia Business School
AT THE VERY CENTER OF BUSINESS

Fall 2024

Training and Testing Models: Financial Analytics

Session 5

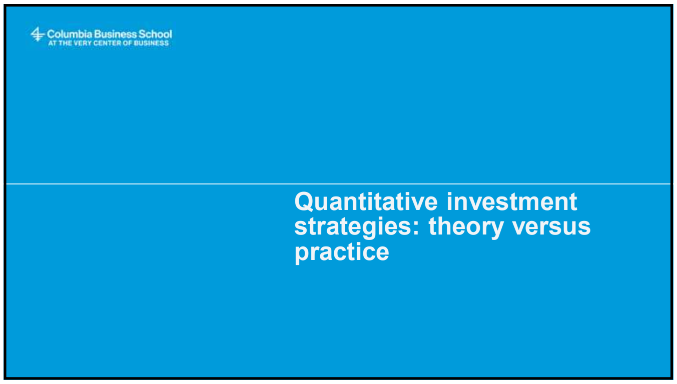
Professor Daniel Guetta



This Module

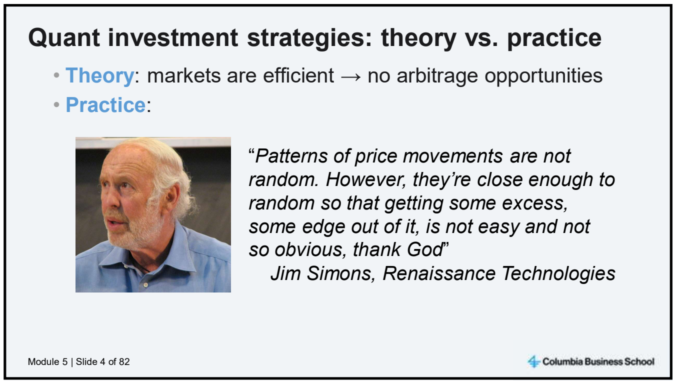
- Financial analytics
- Predicting stock returns
- Quantitative investment strategies
- Prediction performance evaluation

Columbia Business School



Columbia Business School
AT THE VERY CENTER OF BUSINESS

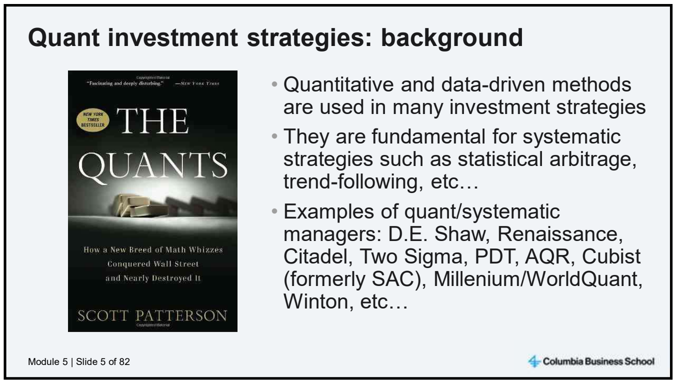
Quantitative investment strategies: theory versus practice



Quant investment strategies: theory vs. practice

- **Theory**: markets are efficient → no arbitrage opportunities
- **Practice**:

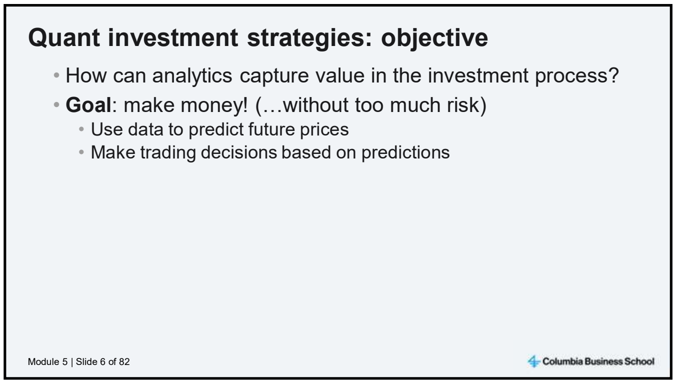
> *"Patterns of price movements are not random. However, they're close enough to random so that getting some excess, some edge out of it, is not easy and not so obvious, thank God"*
> *Jim Simons, Renaissance Technologies*



Quant investment strategies: background

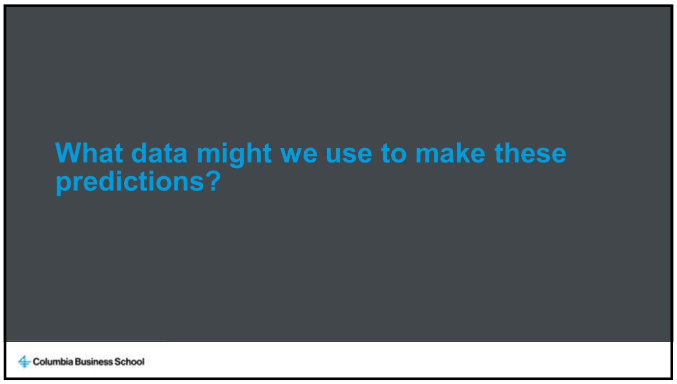
"Fascinating and deeply disturbing." —New York Times
NEW YORK TIMES BESTSELLER
THE QUANTS
How a New Breed of Math Whizzes Conquered Wall Street and Nearly Destroyed It
SCOTT PATTERSON

- Quantitative and data-driven methods are used in many investment strategies
- They are fundamental for systematic strategies such as statistical arbitrage, trend-following, etc…
- Examples of quant/systematic managers: D.E. Shaw, Renaissance, Citadel, Two Sigma, PDT, AQR, Cubist (formerly SAC), Millenium/WorldQuant, Winton, etc…

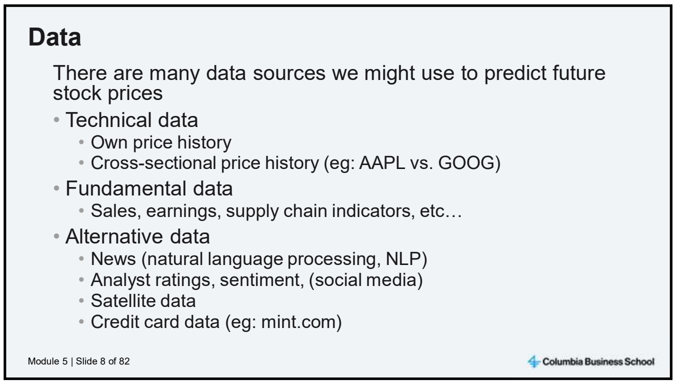


Quant investment strategies: objective

- How can analytics capture value in the investment process?
- **Goal**: make money! (…without too much risk)
  - Use data to predict future prices
  - Make trading decisions based on predictions



What data might we use to make these predictions?

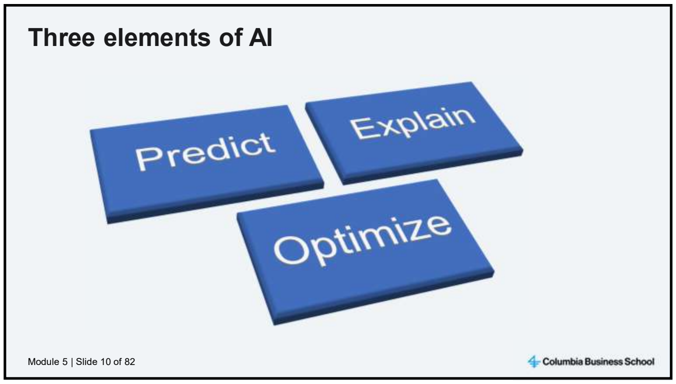


Data

There are many data sources we might use to predict future stock prices

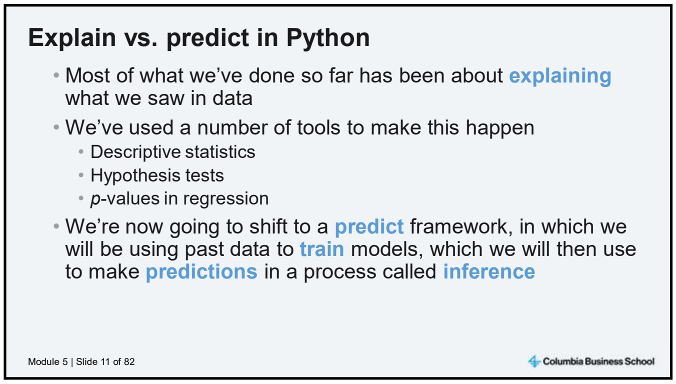
- Technical data
  - Own price history
  - Cross-sectional price history (eg: AAPL vs. GOOG)
- Fundamental data
  - Sales, earnings, supply chain indicators, etc…
- Alternative data
  - News (natural language processing, NLP)
  - Analyst ratings, sentiment, (social media)
  - Satellite data
  - Credit card data (eg: mint.com)

`statsmodels` vs. `sklearn`



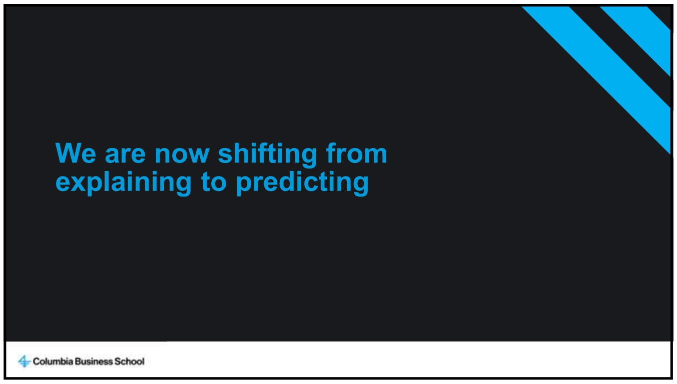
Three elements of AI

Predict
Explain
Optimize



Explain vs. predict in Python

- Most of what we've done so far has been about **explaining** what we saw in data
- We've used a number of tools to make this happen
  - Descriptive statistics
  - Hypothesis tests
  - *p*-values in regression
- We're now going to shift to a **predict** framework, in which we will be using past data to **train** models, which we will then use to make **predictions** in a process called **inference**



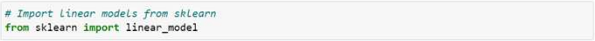
We are now shifting from explaining to predicting

statsmodels vs. scikitlearn

- We have thus far relied on statsmodels for our modelling efforts
- The package is useful for “explain” use cases (what we might call “traditional” statistics)
- We *could* also use it for predict use cases, but there is another Python package, scikit-learn (or sklearn) that is far better suited for these use cases
- It comprises an enormous number of features – we’ll only scratch the surface in this class

Importing sklearn

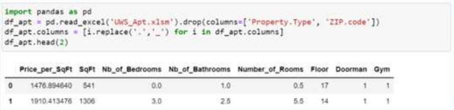
- sklearn is vast and contains many sub-packages; it is good practice to import only those you need
- The documentation is a great place to start if you want to learn more
- Let’s begin by importing the package that does linear regression



```
# Import linear models from sklearn
from sklearn import linear_model
```

Re-running the UWS apartment regression

- Let’s re-run the UWS apartment regression
- Start by loading the data

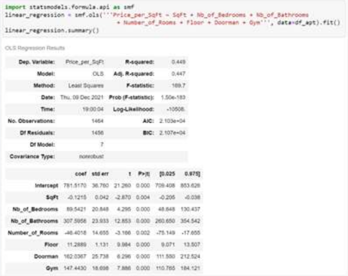


```
import pandas as pd
df_apt = pd.read_excel('UWS_Apt.xlsm').drop(columns=['Property.Type', 'ZIP.code'])
df_apt.columns = [i.replace('.','_') for i in df_apt.columns]
df_apt.head(2)
```

	Price_per_SqFt	SqFt	Nb_of_Bedrooms	Nb_of_Bathrooms	Number_of_Rooms	Floor	Doorman	Gym
0	1476.894640	541	0.0	1.0	0.5	17	1	1
1	1910.413476	1306	3.0	2.5	5.5	14	1	1

- Notice that we are dropping the categorical variables
- sklearn can handle categoricals, but it’s a little more difficult than with statsmodels. If we have time, we’ll cover this at the end of class

A reminder: linear regression in statsmodels



```
import statsmodels.formula.api as smf
linear_regression = smf.ols('''Price_per_SqFt ~ SqFt + Nb_of_Bedrooms + Nb_of_Bathrooms
                        + Number_of_Rooms + Floor + Doorman + Gym''', data=df_apt).fit()
linear_regression.summary()
```

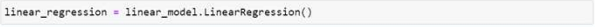
OLS Regression Results

Dep. Variable:	Price_per_SqFt	R-squared:	0.449
Model:	OLS	Adj. R-squared:	0.447
Method:	Least Squares	F-statistic:	169.7
Date:	Thu, 09 Dec 2021	Prob (F-statistic):	1.50e-183
Time:	19:00:04	Log-Likelihood:	-10508.
No. Observations:	1464	AIC:	2.103e+04
Df Residuals:	1456	BIC:	2.107e+04
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	781.5170	36.760	21.260	0.000	709.408	853.626
SqFt	-0.1215	0.042	-2.870	0.004	-0.205	-0.038
Nb_of_Bedrooms	89.5421	20.848	4.295	0.000	48.648	130.437
Nb_of_Bathrooms	307.5958	23.933	12.853	0.000	260.650	354.542
Number_of_Rooms	-46.4018	14.655	-3.166	0.002	-75.149	-17.655
Floor	11.2889	1.131	9.984	0.000	9.071	13.507
Doorman	162.0367	25.738	6.296	0.000	111.550	212.524
Gym	147.4430	18.698	7.886	0.000	110.765	184.121

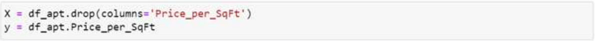
Linear regression in sklearn

All predictive models in sklearn begin with a **model object**. Let’s create a linear regression model object



```
linear_regression = linear_model.LinearRegression()
```

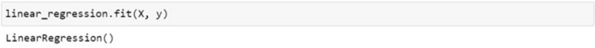
sklearn models don’t use formulas – instead, we have to create a DataFrame containing the *x* columns only, and a Series containing the *y* column only:



```
X = df_apt.drop(columns='Price_per_SqFt')
y = df_apt.Price_per_SqFt
```

Linear regression in sklearn

We are now ready to fit the model using the X and y DataFrames

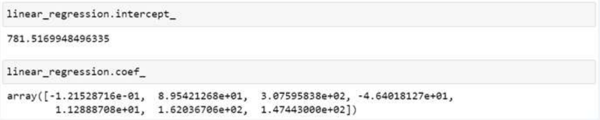


```
linear_regression.fit(X, y)
LinearRegression()
```

Notice how – unlike in statsmodels, .fit() modifies the model object itself; there’s no need to save anything it returns.

Linear regression in sklearn

Once we've fit our model, we can view the intercept and the coefficients



```
linear_regression.intercept_
```

```
781.5169948496335
```

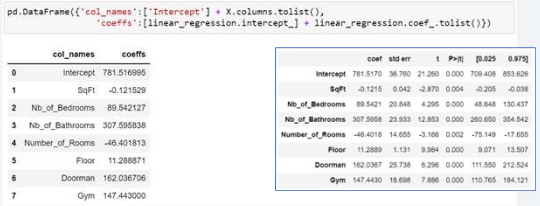
```
linear_regression.coef_
```

```
array([-1.21528716e-01,  8.95421268e+01,  3.07595838e+02, -4.64018127e+01,
        1.12888708e+01,  1.62036706e+02,  1.47443000e+02])
```

Notice how much less convenient this is for explain use cases – there is no `.summary()` in the style of `statsmodels`… In predict use cases, the coefficients aren't really the point.

Linear regression in sklearn

If we want to, we can view the coefficients next to the variable names; they match `statsmodels` exactly



```
pd.DataFrame({'col_names':['Intercept'] + X.columns.tolist(),
              'coeffs':[linear_regression.intercept_] + linear_regression.coef_.tolist()})
```

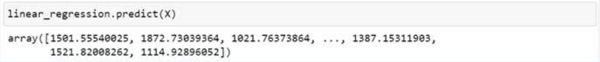
	col_names	coeffs
0	Intercept	781.516995
1	SqFt	-0.121529
2	Nb_of_Bedrooms	89.542127
3	Nb_of_Bathrooms	307.595838
4	Number_of_Rooms	-46.401813
5	Floor	11.288871
6	Doorman	162.036706
7	Gym	147.443000

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	781.5170	36.760	21.260	0.000	709.408	853.626
SqFt	-0.1215	0.042	-2.870	0.004	-0.205	-0.038
Nb_of_Bedrooms	89.5421	20.848	4.295	0.000	48.648	130.437
Nb_of_Bathrooms	307.5958	23.933	12.853	0.000	260.650	354.542
Number_of_Rooms	-46.4018	14.655	-3.166	0.002	-75.149	-17.655
Floor	11.2889	1.131	9.984	0.000	9.071	13.507
Doorman	162.0367	25.738	6.296	0.000	111.550	212.524
Gym	147.4430	18.698	7.886	0.000	110.765	184.121

`sklearn` doesn't give us *p*-values, nor does it allow us to see coefficients particularly easily. But it's perfect for predict use cases, and supports many more models than `statsmodels`

Making predictions in sklearn

- Making predictions in `sklearn` is easy
- We simply create a new matrix containing *exactly* the same *x* columns in the same order…
- …and then use `.predict()` on them

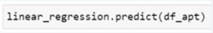


```
linear_regression.predict(X)
```

```
array([1501.55540025, 1872.73039364, 1021.76373864, ..., 1387.15311903,
       1521.82008262, 1114.92896052])
```

Predictions: statsmodels vs sklearn

`statsmodels`

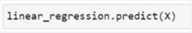


```
linear_regression.predict(df_apt)
```

A statsmodels linear regression object can make a predicition on a DataFrame even if

- It contains extra columns over and above those in the training data
- And even if they are not in the same order

`sklearn`



```
linear_regression.predict(X)
```

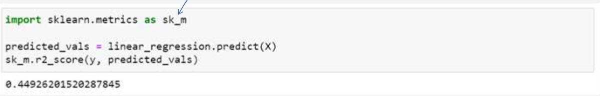
A `sklearn` linear regression object can only make predictions on a DataFrame that has

- *exactly* the same columns as the training data
- in the same order

Calculating R^2 in sklearn

- `sklearn` contains utility functions that allow us to measure all kinds of metrics – for example, the *R*-squared
- You first have to make predictions using the model, and you can then use `sklearn` to compare them to the true values and find the R^2

In this class, we will use abbreviations of this kind to refer to sklearn packages we import

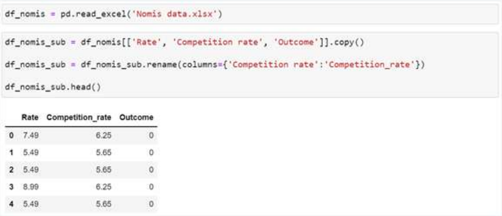


```
import sklearn.metrics as sk_m

predicted_vals = linear_regression.predict(X)
sk_m.r2_score(y, predicted_vals)
```

```
0.44926201520287845
```

Logistic regression

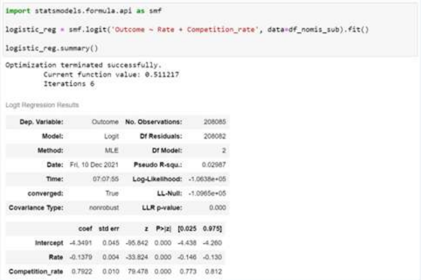
sklearn can also handle logistic regression; let's import the Nomis dataset, and keep only two variables for simplicity



```
df_nomis = pd.read_excel('Nomis data.xlsx')

df_nomis_sub = df_nomis[['Rate', 'Competition rate', 'Outcome']].copy()

df_nomis_sub = df_nomis_sub.rename(columns={'Competition rate':'Competition_rate'})

df_nomis_sub.head()
```

	Rate	Competition_rate	Outcome
0	7.49	6.25	0
1	5.49	5.65	0
2	5.49	5.65	0
3	8.99	6.25	0
4	5.49	5.65	0

Logistic regression in statsmodels



```
import statsmodels.formula.api as smf

logistic_reg = smf.logit('Outcome ~ Rate + Competition_rate', data=df_nomis_sub).fit()

logistic_reg.summary()
```

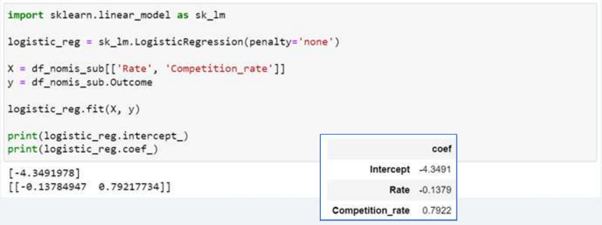
```
Optimization terminated successfully.
         Current function value: 0.511217
         Iterations 6
```

Logit Regression Results

Dep. Variable:	Outcome	No. Observations:	208085
Model:	Logit	Df Residuals:	208082
Method:	MLE	Df Model:	2
Date:	Fri, 10 Dec 2021	Pseudo R-squ.:	0.02987
Time:	07:07:55	Log-Likelihood:	-1.0638e+05
converged:	True	LL-Null:	-1.0965e+05
Covariance Type:	nonrobust	LLR p-value:	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-4.3491	0.045	-95.842	0.000	-4.438	-4.260
Rate	-0.1379	0.004	-33.824	0.000	-0.146	-0.130
Competition_rate	0.7922	0.010	79.478	0.000	0.773	0.812

Logistic regression in sklearn

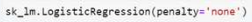


```
import sklearn.linear_model as sk_lm

logistic_reg = sk_lm.LogisticRegression(penalty='none')

X = df_nomis_sub[['Rate', 'Competition_rate']]
y = df_nomis_sub.Outcome

logistic_reg.fit(X, y)

print(logistic_reg.intercept_)
print(logistic_reg.coef_)
```

```
[-4.3491978]
[[-0.13784947  0.79217734]]
```

	coef
Intercept	-4.3491
Rate	-0.1379
Competition_rate	0.7922

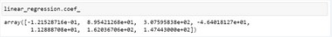
Logistic regression in sklearn: warning 1



```
sk_lm.LogisticRegression(penalty='none')
```

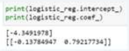
- By default, sklearn adds a **penalty** to logistic regression
- This topic goes beyond this class, and is covered in BA2
- For now, suffice it to say that if you want to fit a simple logistic regression of the kind we have described (that matches the statsmodels regression) you must pass penalty='none' when you create the LogisticRegression model

Logistic regression in sklearn: warning 2



```
linear_regression.coef_
```

```
array([-1.21528716e-01,  8.95421268e+01,  3.07595838e+02, -4.64018127e+01,
        1.12888708e+01,  1.62036706e+02,  1.47443000e+02])
```

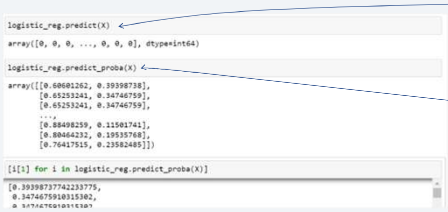


```
print(logistic_reg.intercept_)
print(logistic_reg.coef_)
```

```
[-4.3491978]
[[-0.13784947  0.79217734]]
```

- LinearRegression.coef_ looks reasonable; an array/list with one entry per coefficient
- For reasons we won't get into, LogisticRegresion.coef_ returns a list with *one* element (another list); that list contains one entry per coefficient

Making logistic regression predictions in sklearn



```
logistic_reg.predict(X)
```

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

```
logistic_reg.predict_proba(X)
```

```
array([[0.60601262, 0.39398738],
       [0.65253241, 0.34746759],
       [0.65253241, 0.34746759],
       ...,
       [0.88498259, 0.11501741],
       [0.80464232, 0.19535768],
       [0.76417515, 0.23582485]])
```

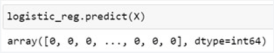
```
[i[1] for i in logistic_reg.predict_proba(X)]
```

```
[0.39398737742233775,
 0.3474675910315302,
 0.3474675910315302
```

This function will use 0.5 as a threshold; any points with a score above 0.5 get classified as "1", others as "0"

This function returns one length-2 list per datapoint. The first element gives the predicted probability is outcome will be 0. The second gives the predicted probability the outcome will be 1.

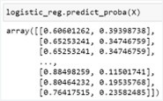
Logistic regression in `sklearn`: warning 3



```
logistic_reg.predict(X)

array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

- **Never, ever, ever, ever**, use `.predict()` for a classification model, unless you know exactly what you're doing
- The choice of 0.5 as a threshold is completely arbitrary (as we'll see in a later lecture)
- Remove this function from your minds completely

Logistic regression in `sklearn`: warning 4



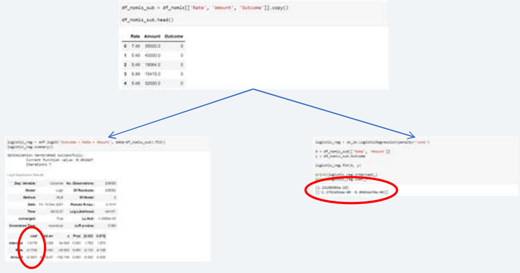
```
logistic_reg.predict_proba(X)

array([[0.60601262, 0.39398738],
       [0.65253241, 0.34746759],
       [0.65253241, 0.34746759],
       ...,
       [0.88498259, 0.11501741],
       [0.80464232, 0.19535768],
       [0.76417515, 0.23582485]])
```

- Remember that `.predict_proba()` returns a list of lists
- You generally want to extract the *second* element to get the probability the outcome is 1.

Logistic regression in `sklearn`: warning 5

- Packages like `sklearn` and `statsmodels` can make all these models seem like simple commodities
- It's easy to forget there are complex, iterative algorithms working in the background (like gradient descent but more complicated) that fit these models
- Like all algorithms, these can sometimes struggle
- Let's look at an example in which two columns are of very different magnitudes

Logistic regression in `sklearn`: warning 5



```
df_nomis_sub = df_nomis[['Rate', 'Amount', 'Outcome']].copy()

df_nomis_sub.head()

   Rate  Amount   Outcome
0  7.49  30000.0  0
1  5.49  40000.0  0
2  5.49  18064.0  0
3  8.99  15415.0  0
4  5.49  32000.0  0
```

```
logistic_reg = smf.logit('Outcome ~ Rate + Amount', data=df_nomis_sub).fit()
logistic_reg.summary()

Optimization terminated successfully.
         Current function value: 0.[illegible]
         Iterations 7

Logit Regression Results
   Dep. Variable:  Outcome    No. Observations:  [illegible]
           Model:  Logit          Df Residuals:  [illegible]
          Method:  MLE                Df Model:  2
            Date:  [illegible]   Pseudo R-squ.:  [illegible]
            Time:  [illegible]  Log-Likelihood:  [illegible]
       converged:  True                LL-Null:  [illegible]
 Covariance Type:  nonrobust       LLR p-value:  [illegible]

             coef         std err  z            P>|z|        [0.025       0.975]
Intercept    [illegible]  [illegible]  [illegible]  [illegible]  [illegible]  [illegible]
Rate         [illegible]  [illegible]  [illegible]  [illegible]  [illegible]  [illegible]
Amount       [illegible]  [illegible]  [illegible]  [illegible]  [illegible]  [illegible]
```

```
logistic_reg = sk_lm.LogisticRegression(penalty='none')

X = df_nomis_sub[['Rate', 'Amount']]
y = df_nomis_sub.Outcome

logistic_reg.fit(X, y)

print(logistic_reg.intercept_)
print(logistic_reg.coef_)

[1.[illegible]e-10]
[[-1.[illegible]e-09 -4.[illegible]e-05]]
```

When coefficients have very different magnitudes, the algorithms we've discussed can sometimes struggle…

Columbia Business School
AT THE VERY CENTER OF BUSINESS

Back to financial analytics

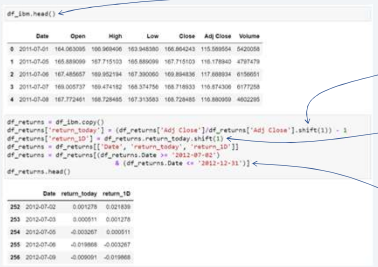
Back to financial analytics

- Let's kick off with a simple, down to earth model
- Consider the IBM stock as an example
- On each day, we can calculate the stock's return as follows

$$\frac{\text{Today's adjusted close} - \text{Last trading day's adjusted close}}{\text{Last trading day's adjusted close}}$$

$$= \frac{\text{Today's adjusted close}}{\text{Last trading day's adjusted close}} - 1$$

Calculating IBM returns



```
df_ibm.head()
```

```
         Date        Open        High         Low       Close   Adj Close   Volume
0  2011-07-01  164.063095  166.969406  163.948380  166.864243  115.589554  5420058
1  2011-07-05  165.889099  167.715103  165.889099  167.715103  116.178940  4797479
2  2011-07-06  167.485657  169.952194  167.390060  169.894836  117.688934  6156651
3  2011-07-07  169.005737  169.474182  168.374756  168.718933  116.874306  6177258
4  2011-07-08  167.772461  168.728485  167.313583  168.728485  116.880959  4602295
```

```
df_returns = df_ibm.copy()
df_returns['return_today'] = (df_returns['Adj Close']/df_returns['Adj Close'].shift(1)) - 1
df_returns['return_1D'] = df_returns.return_today.shift(1)
df_returns = df_returns[['Date', 'return_today', 'return_1D']]
df_returns = df_returns[(df_returns.Date >= '2012-07-02')
                        & (df_returns.Date <= '2012-12-31')]
df_returns.head()
```

```
           Date  return_today  return_1D
252  2012-07-02      0.001278   0.021839
253  2012-07-03      0.000511   0.001278
254  2012-07-05     -0.003267   0.000511
255  2012-07-06     -0.019868  -0.003267
256  2012-07-09     -0.009091  -0.019868
```

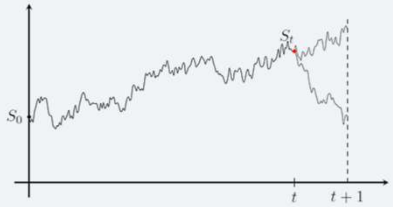
Downloaded from a finance API – see notebook for optional code

Shift the column one row down, so we're looking at the previous day's value

Keep track of the returns the day before; we'll need this later

Keep July-December 2012

Examples of patterns: momentum and reversals



Suppose the stock has been running up… Will it continue going up (**momentum**) or start going down (**reversal**)?

If we could predict whether momentum or reversal is more likely for a stock, we could trade on this information!

How might we use analytics to predict this?

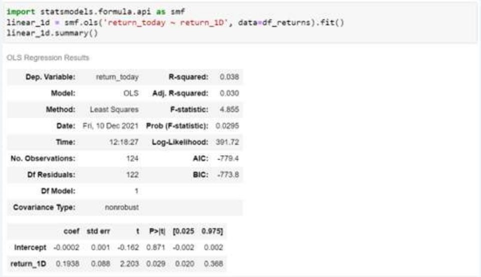
Linear regression

We could start with a very simple model that predicts returns on each day based on returns the day before

$$\text{return\_today} = \beta_0 + \beta_1 \cdot \text{return\_1D} + \text{error}$$

- What would you expect the value of β_0 to be?
- How could we look at the results of this model and determine whether we have momentum, reversal, neither or both?

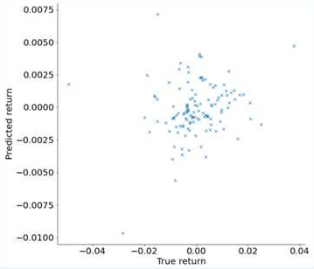
Linear regression



```
import statsmodels.formula.api as smf
linear_1d = smf.ols('return_today ~ return_1D', data=df_returns).fit()
linear_1d.summary()
```

```
OLS Regression Results
   Dep. Variable:    return_today       R-squared:   0.038
           Model:             OLS   Adj. R-squared:   0.030
          Method:   Least Squares     F-statistic:   4.855
            Date: Fri, 10 Dec 2021 Prob (F-statistic): 0.0295
            Time:        12:18:27  Log-Likelihood:  391.72
No. Observations:             124             AIC:  -779.4
    Df Residuals:             122             BIC:  -773.8
        Df Model:               1
 Covariance Type:       nonrobust

              coef  std err       t  P>|t|  [0.025  0.975]
Intercept  -0.0002    0.001  -0.162  0.871  -0.002   0.002
return_1D   0.1938    0.088   2.203  0.029   0.020   0.368
```

Linear regression



From predictions to trades

How can we design a trading strategy based on these predictions?

A simple trading strategy

- Every night, close out your position
- Then, observe the previous day's return
- Predict the next day's return
  - If we predict a positive return, buy the stock (go long)
  - If we predict a negative return, sell the stock (go short)
- (This, of course, ignores any tax/transaction fee implications, but it'll serve as a first model)

A simple strategy

`= -0.0002 + 0.1938 × Yesterday's return`

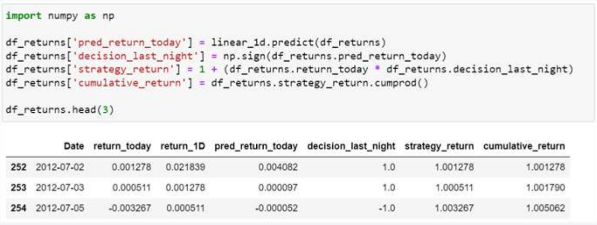
Buy if the predicted return is positive, sell otherwise

Equal to actual return if we went long, and <u>minus</u> the actual return if we shorted

Previous day's "return thus far" × (1 + this day's strategy return)

Date	Yesterday's return	Predicted return today	Decision made last night	Actual return today	Strategy return	Return thus far
7/2/2012	2.18%	0.41%	Long	0.13%	0.13%	1.0013
7/3/2012	0.13%	0.01%	Long	0.05%	0.05%	1.0018
7/5/2012	0.05%	–0.01%	Short	–0.32%	0.32%	1.0051
7/6/2012	–0.32%	–0.08%	Short	–1.99%	1.99%	1.0250
7/9/2012	–1.99%	–0.40%	Short	–0.91%	0.91%	1.0343

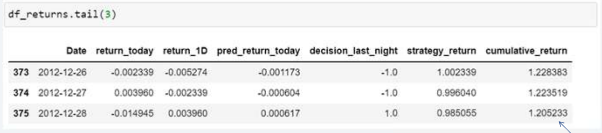
A simple strategy in Python



```python
import numpy as np

df_returns['pred_return_today'] = linear_1d.predict(df_returns)
df_returns['decision_last_night'] = np.sign(df_returns.pred_return_today)
df_returns['strategy_return'] = 1 + (df_returns.return_today * df_returns.decision_last_night)
df_returns['cumulative_return'] = df_returns.strategy_return.cumprod()

df_returns.head(3)
```

```
           Date  return_today  return_1D  pred_return_today  decision_last_night  strategy_return  cumulative_return
252  2012-07-02      0.001278   0.021839           0.004082                  1.0         1.001278           1.001278
253  2012-07-03      0.000511   0.001278           0.000097                  1.0         1.000511           1.001790
254  2012-07-05     -0.003267   0.000511          -0.000052                 -1.0         1.003267           1.005062
```

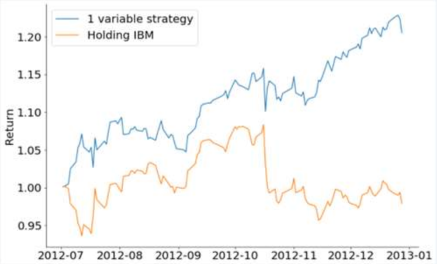
Total return after 6 months…



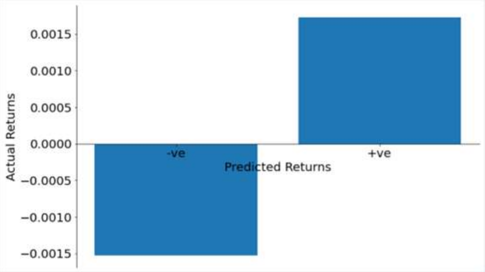
```
df_returns.tail(3)
```

	Date	return_today	return_1D	pred_return_today	decision_last_night	strategy_return	cumulative_return
373	2012-12-26	-0.002339	-0.005274	-0.001173	-1.0	1.002339	1.228383
374	2012-12-27	0.003960	-0.002339	-0.000604	-1.0	0.996040	1.223519
375	2012-12-28	-0.014945	0.003960	0.000617	1.0	0.985055	1.205233

Total return after 6 months

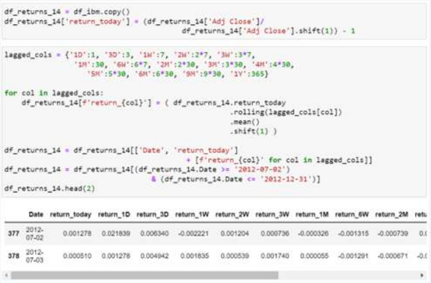


How does a bad model do so well?



How can we do even better?

Using 14 variables

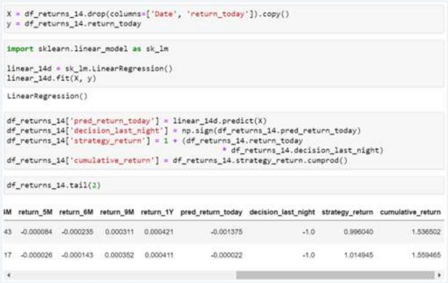


```
df_returns_14 = df_ibm.copy()
df_returns_14['return_today'] = (df_returns_14['Adj Close']/
                                  df_returns_14['Adj Close'].shift(1)) - 1
```

```
lagged_cols = {'1D':1, '3D':3, '1W':7, '2W':2*7, '3W':3*7,
               '1M':30, '6W':6*7, '2M':2*30, '3M':3*30, '4M':4*30,
                 '5M':5*30, '6M':6*30, '9M':9*30, '1Y':365}

for col in lagged_cols:
    df_returns_14[f'return_{col}'] = ( df_returns_14.return_today
                                              .rolling(lagged_cols[col])
                                              .mean()
                                              .shift(1) )

df_returns_14 = df_returns_14[['Date', 'return_today']
                              + [f'return_{col}' for col in lagged_cols]]
df_returns_14 = df_returns_14[(df_returns_14.Date >= '2012-07-02')
                          & (df_returns_14.Date <= '2012-12-31')]
df_returns_14.head(2)
```

	Date	return_today	return_1D	return_3D	return_1W	return_2W	return_3W	return_1M	return_6W	return_2M	retu
377	2012-07-02	0.001278	0.021839	0.006340	-0.002221	0.001204	0.000736	-0.000326	-0.001315	-0.000739	0.(
378	2012-07-03	0.000510	0.001278	0.004942	0.001835	0.000539	0.001740	0.000055	-0.001291	-0.000671	-0.(

Using 14 variables



```
X = df_returns_14.drop(columns=['Date', 'return_today']).copy()
y = df_returns_14.return_today
```

```
import sklearn.linear_model as sk_lm

linear_14d = sk_lm.LinearRegression()
linear_14d.fit(X, y)
```

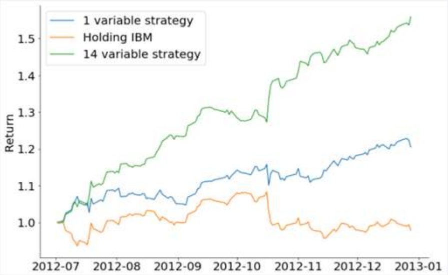
```
LinearRegression()
```

```
df_returns_14['pred_return_today'] = linear_14d.predict(X)
df_returns_14['decision_last_night'] = np.sign(df_returns_14.pred_return_today)
df_returns_14['strategy_return'] = 1 + (df_returns_14.return_today
                                        * df_returns_14.decision_last_night)
df_returns_14['cumulative_return'] = df_returns_14.strategy_return.cumprod()
```

```
df_returns_14.tail(2)
```

4M	return_5M	return_6M	return_9M	return_1Y	pred_return_today	decision_last_night	strategy_return	cumulative_return
43	-0.000084	-0.000235	0.000311	0.000421	-0.001375	-1.0	0.996040	1.536502
17	-0.000026	-0.000143	0.000352	0.000411	-0.000022	-1.0	1.014945	1.559465

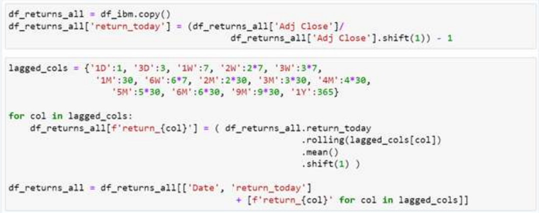
Using 14 variables



How does our strategy perform over the next 6 months?

The next 6 months…

First, prepare the data without removing any dates

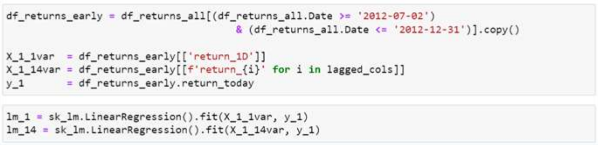


```
df_returns_all = df_ibm.copy()
df_returns_all['return_today'] = (df_returns_all['Adj Close']/
                                   df_returns_all['Adj Close'].shift(1)) - 1
```

```
lagged_cols = {'1D':1, '3D':3, '1W':7, '2W':2*7, '3W':3*7,
               '1M':30, '6W':6*7, '2M':2*30, '3M':3*30, '4M':4*30,
                 '5M':5*30, '6M':6*30, '9M':9*30, '1Y':365}

for col in lagged_cols:
    df_returns_all[f'return_{col}'] = ( df_returns_all.return_today
                                               .rolling(lagged_cols[col])
                                               .mean()
                                               .shift(1) )

df_returns_all = df_returns_all[['Date', 'return_today']
                                 + [f'return_{col}' for col in lagged_cols]]
```

The next 6 months…

Create the model as we did on July 2012 – December 2012

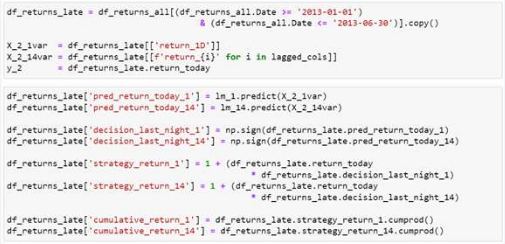


```
df_returns_early = df_returns_all[(df_returns_all.Date >= '2012-07-02')
                                  & (df_returns_all.Date <= '2012-12-31')].copy()

X_1_1var  = df_returns_early[['return_1D']]
X_1_14var = df_returns_early[[f'return_{i}' for i in lagged_cols]]
y_1       = df_returns_early.return_today
```

```
lm_1 = sk_lm.LinearRegression().fit(X_1_1var, y_1)
lm_14 = sk_lm.LinearRegression().fit(X_1_14var, y_1)
```

The next 6 months…

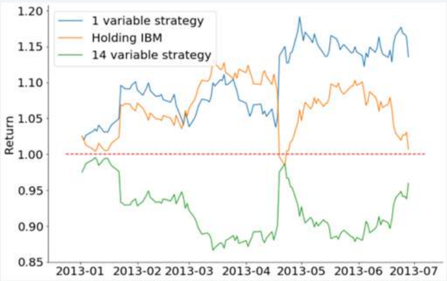
Run the strategy on the next 6 months



```
df_returns_late = df_returns_all[(df_returns_all.Date >= '2013-01-01')
                                 & (df_returns_all.Date <= '2013-06-30')].copy()

X_2_1var  = df_returns_late[['return_1D']]
X_2_14var = df_returns_late[[f'return_{i}' for i in lagged_cols]]
y_2       = df_returns_late.return_today
```

```
df_returns_late['pred_return_today_1'] = lm_1.predict(X_2_1var)
df_returns_late['pred_return_today_14'] = lm_14.predict(X_2_14var)

df_returns_late['decision_last_night_1'] = np.sign(df_returns_late.pred_return_today_1)
df_returns_late['decision_last_night_14'] = np.sign(df_returns_late.pred_return_today_14)

df_returns_late['strategy_return_1'] = 1 + (df_returns_late.return_today
                                            * df_returns_late.decision_last_night_1)
df_returns_late['strategy_return_14'] = 1 + (df_returns_late.return_today
                                             * df_returns_late.decision_last_night_14)

df_returns_late['cumulative_return_1'] = df_returns_late.strategy_return_1.cumprod()
df_returns_late['cumulative_return_14'] = df_returns_late.strategy_return_14.cumprod()
```

The next 6 months…

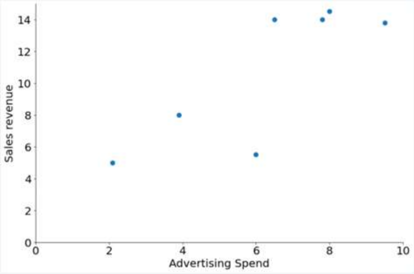


What happened?!

(a) Why didn’t either strategy do as well as expected? (b) Why did the 14 variable strategy (expected to do better) end up doing worse?!

Performance evaluation: train versus test sets

Example



Two potential models

Model 1

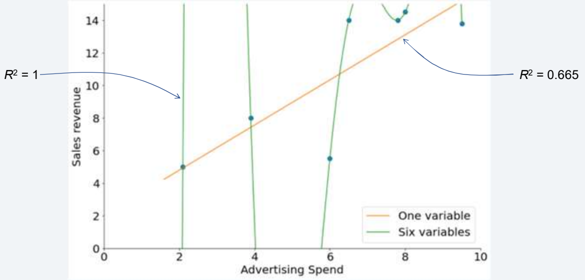
$$y = 2.00 + 1.39x$$

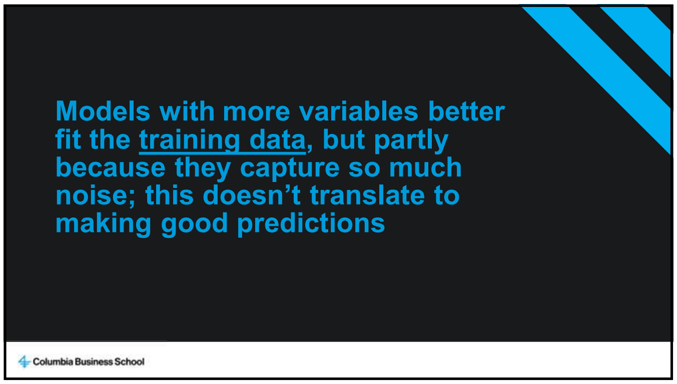
Model 2

$$y = -5506.49 + 6892.44x - 3315.93x^2 + 799.61x^3 - 103.25x^4 + 6.83x^5 - 0.18x^6$$

Which of these models would you use?

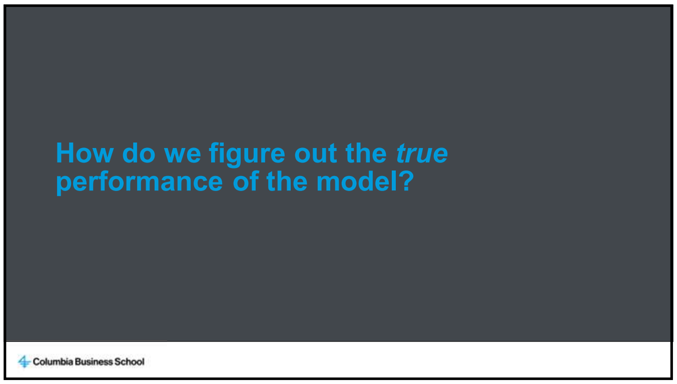
More variables = better fit = better predictions?





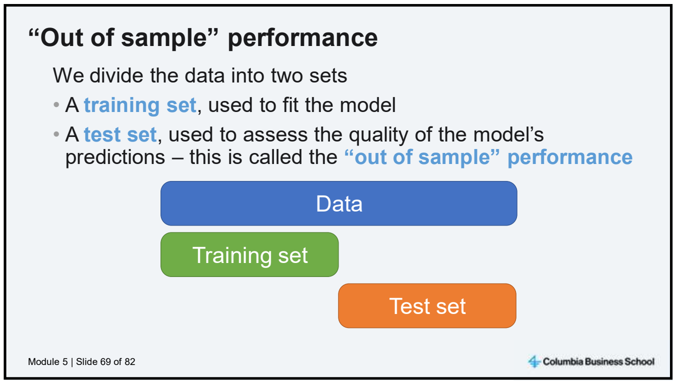
Models with more variables better fit the training data, but partly because they capture so much noise; this doesn't translate to making good predictions

Columbia Business School



How do we figure out the *true* performance of the model?

Columbia Business School

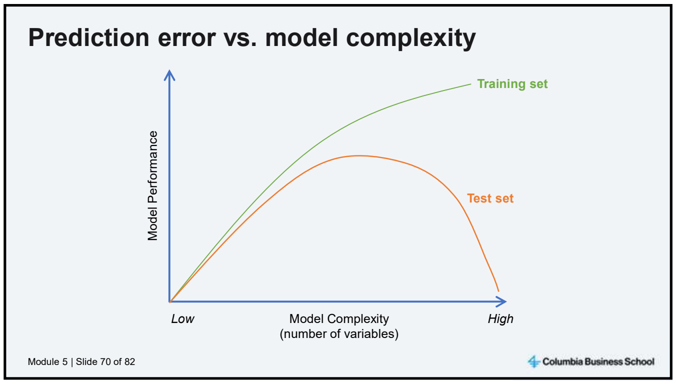


"Out of sample" performance

We divide the data into two sets

- A **training set**, used to fit the model
- A **test set**, used to assess the quality of the model's predictions – this is called the **"out of sample" performance**

Columbia Business School

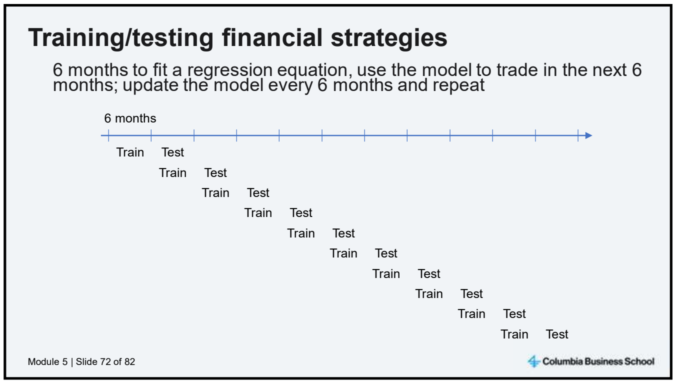


Prediction error vs. model complexity

Columbia Business School

Columbia Business School
AT THE VERY CENTER OF BUSINESS

Training/testing financial strategies

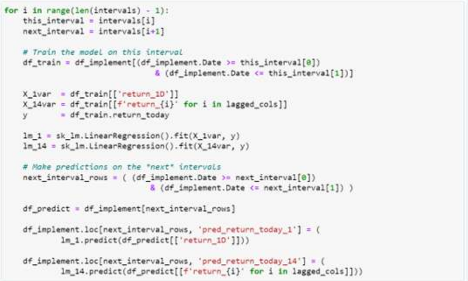


Training/testing financial strategies

6 months to fit a regression equation, use the model to trade in the next 6 months; update the model every 6 months and repeat

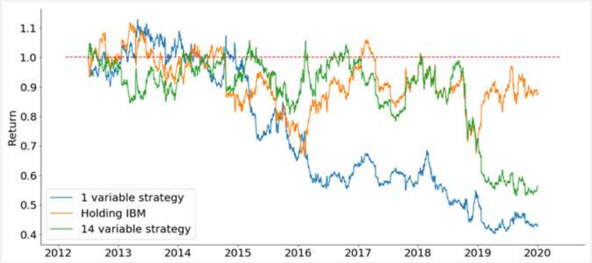
Columbia Business School

Sequentially training a model



```
for i in range(len(intervals) - 1):
    this_interval = intervals[i]
    next_interval = intervals[i+1]

    # Train the model on this interval
    df_train = df_implement[(df_implement.Date >= this_interval[0])
                               & (df_implement.Date <= this_interval[1])]

    X_1var  = df_train[['return_1D']]
    X_14var = df_train[[f'return_{i}' for i in lagged_cols]]
    y       = df_train.return_today

    lm_1 = sk_lm.LinearRegression().fit(X_1var, y)
    lm_14 = sk_lm.LinearRegression().fit(X_14var, y)

    # Make predictions on the "next" intervals
    next_interval_rows = ( (df_implement.Date >= next_interval[0])
                              & (df_implement.Date <= next_interval[1]) )

    df_predict = df_implement[next_interval_rows]

    df_implement.loc[next_interval_rows, 'pred_return_today_1'] = (
            lm_1.predict(df_predict[['return_1D']]))

    df_implement.loc[next_interval_rows, 'pred_return_today_14'] = (
            lm_14.predict(df_predict[[f'return_{i}' for i in lagged_cols]]))
```

The results



Where to go from here

- We can get more power by using 50 stocks instead of just 1
  - Every day, predict the returns for the 50 stocks
  - Buy those with the top 5 predicted returns, short those with the bottom 5 predicted returns (this is a “neutral” portfolio)
- However complex the strategy, we need a principled test/train approach to make sure we’re not overfitting

Alternative data has also become a major part of the way quantitative trading is done today

Miscellaneous thoughts

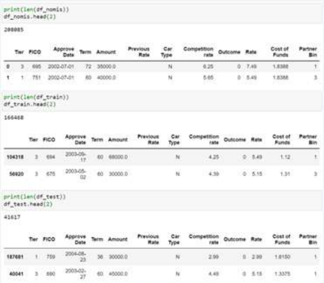
Methods for creating train/test sets

- This lecture has dealt with a very specific kind of time series data, in which we can create train/test sets chronologically
- In other non-time series cases, it makes more sense to split training and test sets randomly
- `sklearn` has functions to make this happen – let’s look at an example on the Nomis data

```
import sklearn.model_selection as sk_ms
df_train, df_test = sk_ms.train_test_split(df_nomis, train_size=0.8, random_state=123)
```

Train and test sets are split randomly, but if you provide a random state, the split will be the same every time you provide the same random state; we will discuss this in far greater detail in our simulation lecture

Methods for creating train/test sets



```
print(len(df_nomis))
df_nomis.head(2)
```

```
208085
```

	Tier	FICO	Approve Date	Term	Amount	Previous Rate	Car Type	Competition rate	Outcome	Rate	Cost of Funds	Partner Bin
0	3	695	2002-07-01	72	35000.0		N	6.25	0	7.49	1.8388	1
1	1	751	2002-07-01	60	40000.0		N	5.65	0	5.49	1.8388	3

```
print(len(df_train))
df_train.head(2)
```

```
166468
```

	Tier	FICO	Approve Date	Term	Amount	Previous Rate	Car Type	Competition rate	Outcome	Rate	Cost of Funds	Partner Bin
104318	3	694	2003-09-17	60	68000.0		N	4.25	0	5.49	1.12	1
56920	3	675	2003-05-02	60	30000.0		N	4.39	0	5.15	1.31	3

```
print(len(df_test))
df_test.head(2)
```

```
41617
```

	Tier	FICO	Approve Date	Term	Amount	Previous Rate	Car Type	Competition rate	Outcome	Rate	Cost of Funds	Partner Bin
187681	1	759	2004-08-23	36	30000.0		N	2.99	0	2.99	1.8150	1
40041	3	690	2003-02-27	60	45000.0		N	4.49	0	5.15	1.3375	1

Methods for creating train/test sets

- In practice, a technique called *K*-fold cross validation is used to achieve the train/test effect without "wasting" data
- This technique goes beyond what we'll discuss in this class, but is covered in BA2

In theory, we should have done all this with Nomis… Why is it likely it wouldn't have made a massive difference?

Picking models using the train/test strategy

- We have been using train/test sets to test our 1 variable strategy against a 14 variable strategy
- What happens if we do this with thousands of different models…
- …we end up **overfitting** to the test set
- The solution is to create a separate **validation set**
- We discuss this in more detail in BA2

Training set
Test set
Evaluation set

Columbia Business School
AT THE VERY CENTER OF BUSINESS

Fall 2024

Quality of Predictions and Classification; Healthcare Analytics

Module 6

Professor Daniel Guetta

This Module

- The challenge of readmissions reduction at Tahoe Healthcare
- Classification: predicting an outcome
- Performance of a classifier (“R^2 for 0/1 outcomes”)
- Economic tradeoffs in classification

Columbia Business School
AT THE VERY CENTER OF BUSINESS

Readmissions at the Tahoe healthcare system

What is a readmission?

Hospital Readmissions

New England Journal of Medicine Study (2009)

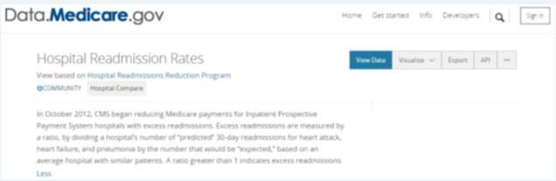
- Approximately 20% of hospitalized Medicare patients are readmitted within 30 days; 34% are readmitted within 90 days
- Estimated cost to the US healthcare system: $17 billion

2010 Affordable Care Act established a Hospital Readmissions Reduction Program (HRRP)

- Medicare payments to hospitals are reduced for excess readmissions
- Three conditions: acute myocardial infraction (AMI), heart failure (HF), and pneumonia
- Based on 30-day, risk-adjusted readmissions rate
- 3-year rolling horizon measure

https://www.cms.gov/Medicare/Medicare-Fee-for-Service-Payment/AcuteInpatientPPS/Readmissions-Reduction-Program

Medicare readmissions stats



Hospital Readmission Rates

View based on Hospital Readmissions Reduction Program

COMMUNITY Hospital Compare

In October 2012, CMS began reducing Medicare payments for Inpatient Prospective Payment System hospitals with excess readmissions. Excess readmissions are measured by a ratio, by dividing a hospital's number of "predicted" 30-day readmissions for heart attack, heart failure, and pneumonia by the number that would be "expected," based on an average hospital with similar patients. A ratio greater than 1 indicates excess readmissions.

Less

Tell me about the Tahoe Healthcare System. How is it affected by readmissions?

Tahoe Healthcare System

- Case study uses real, but anonymized data
- Operates 14 hospitals in the Pacific Northwest
- 18% of total revenues are from Medicare reimbursement for the three HRRP conditions
- Management is concerned about the impact of the new HRRP rules on reimbursement revenues

Interventions to reduce readmissions

- During hospitalization
  - Tailored patient care
  - Communication with PCP, family and home care
  - Patient education
- At discharge
  - Discharge planning
  - Patient/caregiver education
  - Transition coaching
  - Schedule and prepare follow-up appointments
- Post-discharge
  - Home nursing visits
  - Phone follow-up checks
  - Tele-health monitoring

CareTracker

- Tahoe has been working with a variety of interventions to try and reduce readmissions
- CareTracker, a new program the clinical staff has piloted with AMI patients has proved effective at reducing readmissions through a combination of patient education and post-discharge monitoring
  - Cost/patient: **$1,200**
  - Reduces readmission risk by **40%**
  - Reimbursement penalty per reamidission: **$8,000**

Data on past patients

Tahoe has provided data on past patients that did **not** receive the CareTracker intervention. We can load it into Python

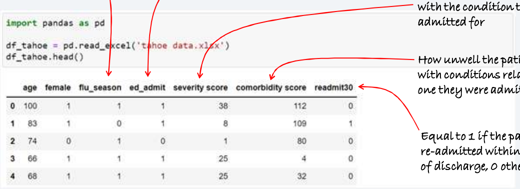
Was the patient admitted during flu season

Was the patient admitted through the emergency department

How unwell the patient is with the condition they were admitted for

How unwell the patient is with conditions related to the one they were admitted for

Equal to 1 if the patient was re-admitted within 30 days of discharge, 0 otherwise

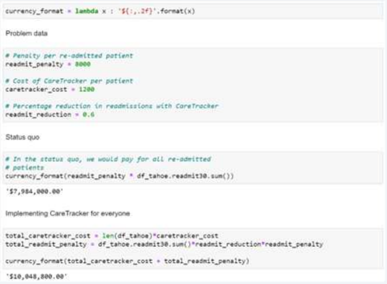


```
import pandas as pd

df_tahoe = pd.read_excel('tahoe data.xlsx')
df_tahoe.head()
```

```
   age  female  flu_season  ed_admit  severity score  comorbidity score  readmit30
0  100       1           1         1              38                112          0
1   83       1           0         1               8                109          1
2   74       0           1         0               1                 80          0
3   66       1           1         1              25                  4          0
4   68       1           1         1              25                 32          0
```

Based on these data, should CareTracker be deployed for *all* patients?

Initial analysis



```
currency_format = lambda x : '${:,.2f}'.format(x)
```

Problem data

```
# Penalty per re-admitted patient
readmit_penalty = 8000

# Cost of CareTracker per patient
caretracker_cost = 1200

# Percentage reduction in readmissions with CareTracker
readmit_reduction = 0.6
```

Status quo

```
# In the status quo, we would pay for all re-admitted
# patients
currency_format(readmit_penalty * df_tahoe.readmit30.sum())
```

```
'$7,984,000.00'
```

Implementing CareTracker for everyone

```
total_caretracker_cost = len(df_tahoe)*caretracker_cost
total_readmit_penalty = df_tahoe.readmit30.sum()*readmit_reduction*readmit_penalty

currency_format(total_caretracker_cost + total_readmit_penalty)
```

```
'$10,048,800.00'
```

The cost matrix approach

The cost matrix

We can calculate these numbers more systematically using a **cost matrix**

What we chose to do – did we give the patient CareTracker (1) or not (0)

		Treatment	
		0	1
Outcome	0	$0	$1,200
	1	$8,000	$6,000

What would have happened in the absence of treatment – would they have gotten readmitted (1) or not (0). This is what we observe in our data

Why $6,000?

If someone *would have been re-admitted* but we give them CareTracker, their probability of being re-admitted drops to 0.6, and so their expected penalty is

$$\$8,000 \times 0.6 = \$4,800$$

Of course, we also need to pay $1,200 to give them CareTracker, which results in a total cost of

$$\$4,800 + \$1,200 = \$6,000$$

Basic scenarios

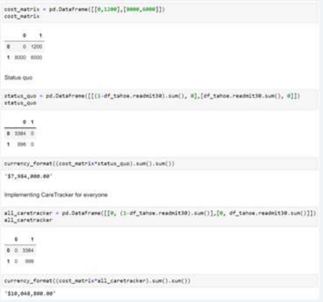
Status quo – no CareTracker for anyone

		Treatment	
		0	1
Outcome	0	3,384	0
	1	998	0

Use CareTracker for everyone

		Treatment	
		0	1
Outcome	0	0	3,384
	1	0	998

Basic scenarios



```
cost_matrix = pd.DataFrame([[0,1200],[8000,6000]])
cost_matrix
```

	0	1
0	0	1200
1	8000	6000

Status quo

```
status_quo = pd.DataFrame([[(1-df_tahoe.readmit30).sum(), 0],[df_tahoe.readmit30.sum(), 0]])
status_quo
```

	0	1
0	3384	0
1	998	0

```
currency_format((cost_matrix*status_quo).sum().sum())
```

```
'$7,984,000.00'
```

Implementing CareTracker for everyone

```
all_caretracker = pd.DataFrame([[0, (1-df_tahoe.readmit30).sum()],[0, df_tahoe.readmit30.sum()]])
all_caretracker
```

	0	1
0	0	3384
1	0	998

```
currency_format((cost_matrix*all_caretracker).sum().sum())
```

```
'$10,048,800.00'
```

Is the idea of CareTracker dead? What else could be done?

We could try and predict how likely patients are to *need* CareTracker, and only prescribe it to people who are very likely to need it

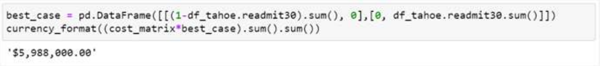
Before we even launch into this, how could we verify that there is some value to be captured here?

Perfect predictions

Suppose we had perfect foresight, and could apply CareTracker *only to patients we knew would be readmitted…*

		Treatment	
		0	1
Outcome	0	3,384	0
	1	0	998

Perfect predictions



```
best_case = pd.DataFrame([[(1-df_tahoe.readmit30).sum(), 0],[0, df_tahoe.readmit30.sum()]])
currency_format((cost_matrix*best_case).sum().sum())

'$5,988,000.00'
```

With perfect foresight, we would go from a status quo of **$7,984,000** to a perfect cost of **$5,988,000**, that is a potential saving of

$1,996,000

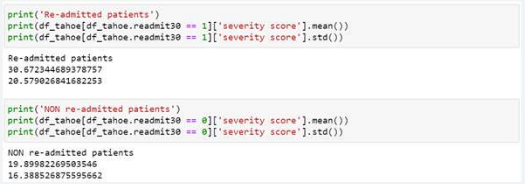
This provides a benchmark for evaluating future improvements.

How might we capture some of this potential value? What approach might we use to try and predict whether someone will need CareTracker?

A first classifier

A first classifier

The severity score seems like a likely candidate…



```
print('Re-admitted patients')
print(df_tahoe[df_tahoe.readmit30 == 1]['severity score'].mean())
print(df_tahoe[df_tahoe.readmit30 == 1]['severity score'].std())
```

```
Re-admitted patients
30.672344689378757
20.579026841682253
```

```
print('NON re-admitted patients')
print(df_tahoe[df_tahoe.readmit30 == 0]['severity score'].mean())
print(df_tahoe[df_tahoe.readmit30 == 0]['severity score'].std())
```

```
NON re-admitted patients
19.89982269503546
16.388526875595662
```

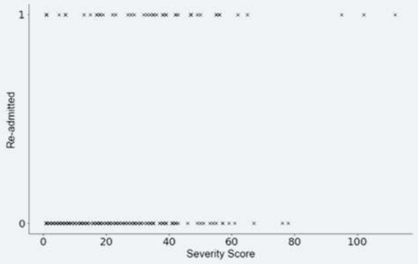
Maybe we should only give CareTracker to patients that have a high severity score when they are discharged?

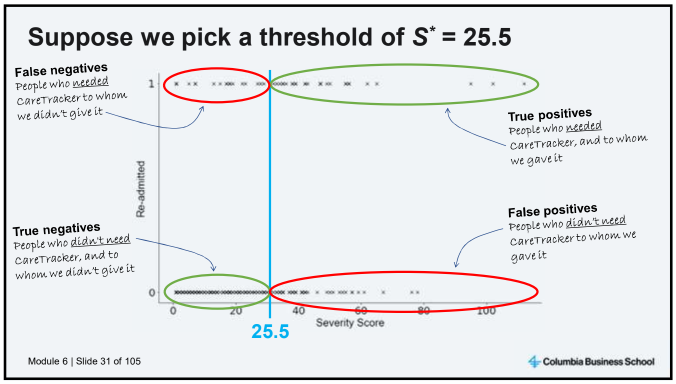
What score should we use as the threshold?

How can we rigorously determine what the best threshold might be?

Performance of a classifier

Evaluating the performance of a classifier





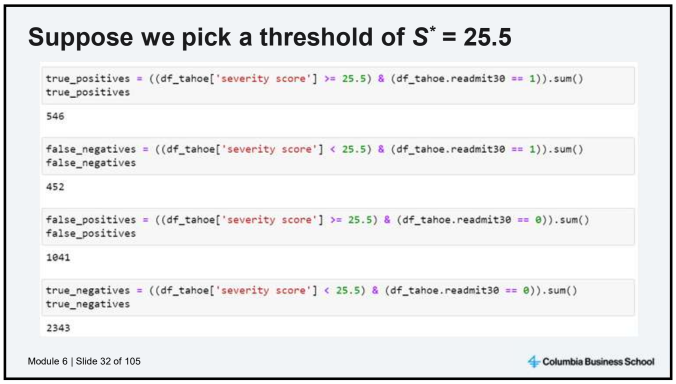
Suppose we pick a threshold of $S^* = 25.5$

False negatives
People who needed CareTracker to whom we didn't give it

True positives
People who needed CareTracker, and to whom we gave it

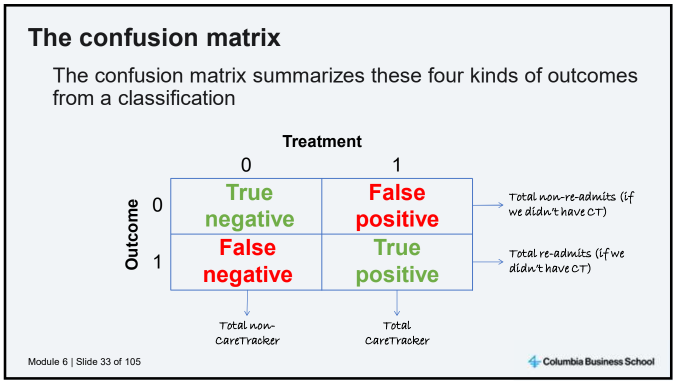
True negatives
People who didn't need CareTracker, and to whom we didn't give it

False positives
People who didn't need CareTracker to whom we gave it



Suppose we pick a threshold of $S^* = 25.5$

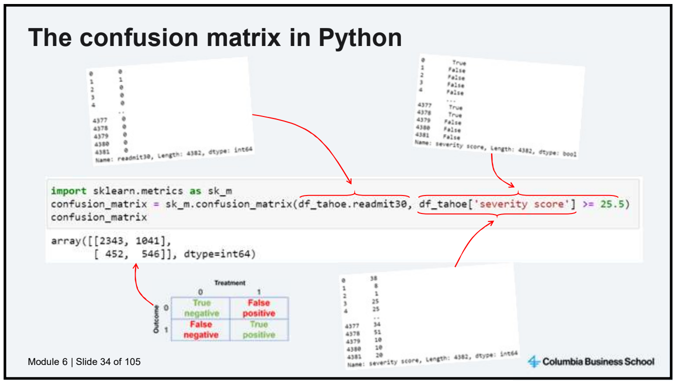
```
true_positives = ((df_tahoe['severity score'] >= 25.5) & (df_tahoe.readmit30 == 1)).sum()
true_positives

546

false_negatives = ((df_tahoe['severity score'] < 25.5) & (df_tahoe.readmit30 == 1)).sum()
false_negatives

452

false_positives = ((df_tahoe['severity score'] >= 25.5) & (df_tahoe.readmit30 == 0)).sum()
false_positives

1041

true_negatives = ((df_tahoe['severity score'] < 25.5) & (df_tahoe.readmit30 == 0)).sum()
true_negatives

2343
```



The confusion matrix

The confusion matrix summarizes these four kinds of outcomes from a classification

		Treatment		
		0	1	
Outcome	0	True negative	False positive	Total non-re-admits (if we didn't have CT)
	1	False negative	True positive	Total re-admits (if we didn't have CT)
		Total non-CareTracker	Total CareTracker	



The confusion matrix in Python

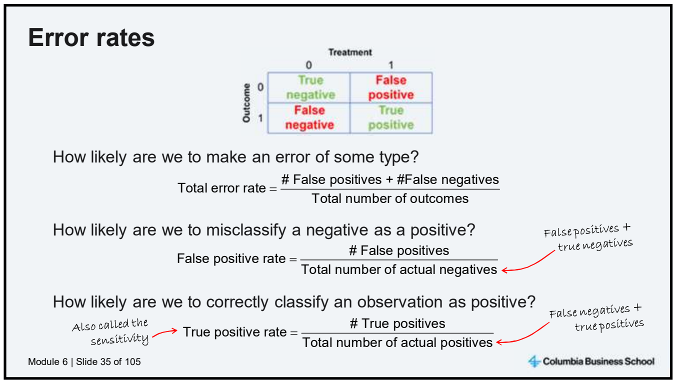
```
0       0
1       1
2       0
3       0
4       0
       ..
4377    0
4378    0
4379    0
4380    0
4381    0
Name: readmit30, Length: 4382, dtype: int64
```

```
0        True
1       False
2       False
3       False
4       False
        ...
4377     True
4378     True
4379    False
4380    False
4381    False
Name: severity score, Length: 4382, dtype: bool
```

```
import sklearn.metrics as sk_m
confusion_matrix = sk_m.confusion_matrix(df_tahoe.readmit30, df_tahoe['severity score'] >= 25.5)
confusion_matrix

array([[2343, 1041],
       [ 452,  546]], dtype=int64)
```

		Treatment	
		0	1
Outcome	0	True negative	False positive
	1	False negative	True positive

```
0       38
1        8
2        1
3       25
4       25
        ..
4377    34
4378    51
4379    10
4380    10
4381    20
Name: severity score, Length: 4382, dtype: int64
```



Error rates

		Treatment	
		0	1
Outcome	0	True negative	False positive
	1	False negative	True positive

How likely are we to make an error of some type?

$$\text{Total error rate} = \frac{\text{\# False positives + \#False negatives}}{\text{Total number of outcomes}}$$

How likely are we to misclassify a negative as a positive?

$$\text{False positive rate} = \frac{\text{\# False positives}}{\text{Total number of actual negatives}}$$

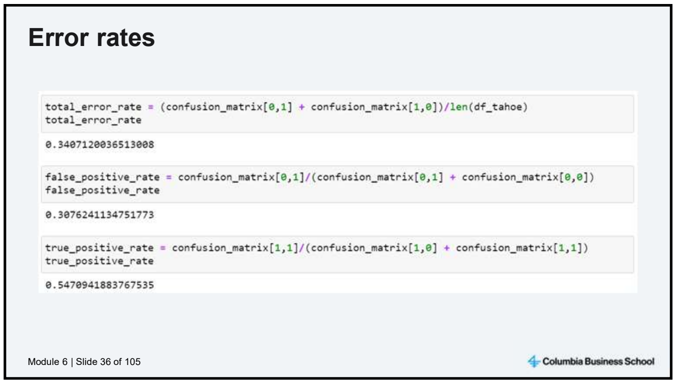
False positives + true negatives

How likely are we to correctly classify an observation as positive?

Also called the sensitivity

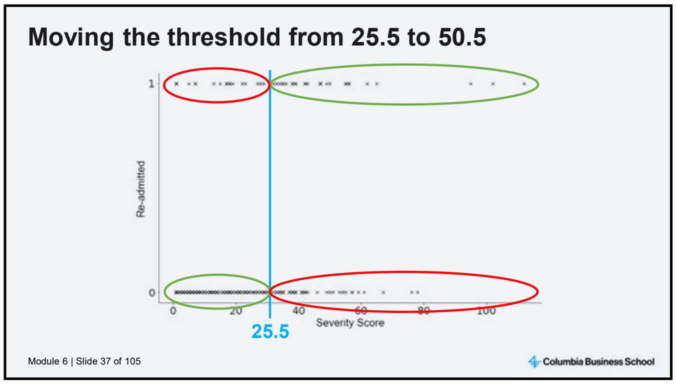
$$\text{True positive rate} = \frac{\text{\# True positives}}{\text{Total number of actual positives}}$$

False negatives + true positives

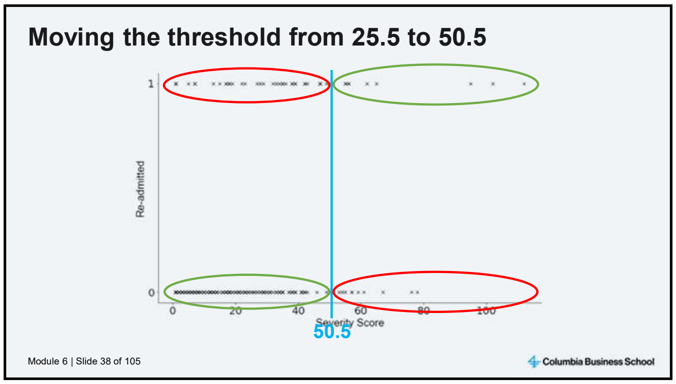


Error rates

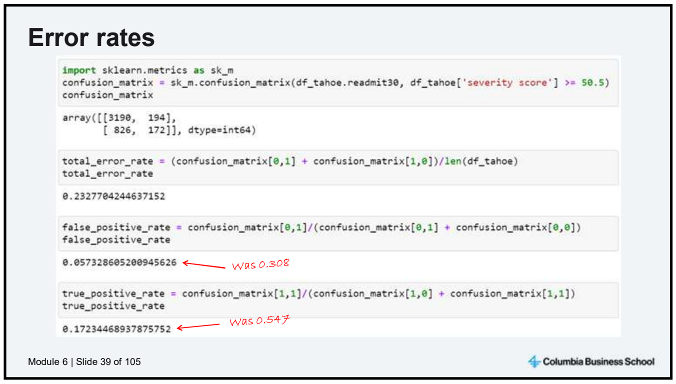
```
total_error_rate = (confusion_matrix[0,1] + confusion_matrix[1,0])/len(df_tahoe)
total_error_rate

0.3407120036513008

false_positive_rate = confusion_matrix[0,1]/(confusion_matrix[0,1] + confusion_matrix[0,0])
false_positive_rate

0.3076241134751773

true_positive_rate = confusion_matrix[1,1]/(confusion_matrix[1,0] + confusion_matrix[1,1])
true_positive_rate

0.5470941883767535
```



Moving the threshold from 25.5 to 50.5



Moving the threshold from 25.5 to 50.5

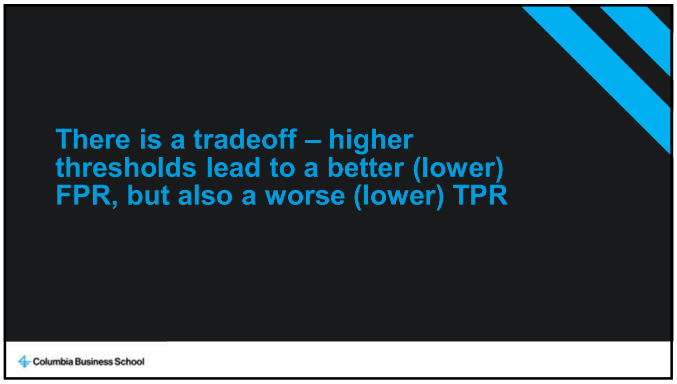


Error rates

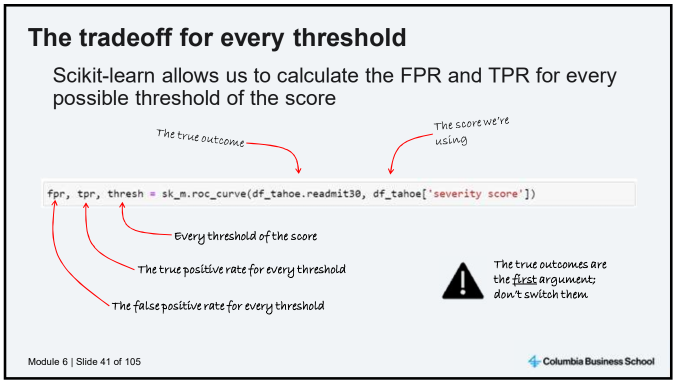
```
import sklearn.metrics as sk_m
confusion_matrix = sk_m.confusion_matrix(df_tahoe.readmit30, df_tahoe['severity score'] >= 50.5)
confusion_matrix

array([[3190,  194],
       [ 826,  172]], dtype=int64)

total_error_rate = (confusion_matrix[0,1] + confusion_matrix[1,0])/len(df_tahoe)
total_error_rate

0.2327704244637152

false_positive_rate = confusion_matrix[0,1]/(confusion_matrix[0,1] + confusion_matrix[0,0])
false_positive_rate

0.057328605200945626

true_positive_rate = confusion_matrix[1,1]/(confusion_matrix[1,0] + confusion_matrix[1,1])
true_positive_rate

0.17234468937875752
```

Was 0.308

Was 0.547



There is a tradeoff – higher thresholds lead to a better (lower) FPR, but also a worse (lower) TPR



The tradeoff for every threshold

Scikit-learn allows us to calculate the FPR and TPR for every possible threshold of the score

The true outcome

The score we're using

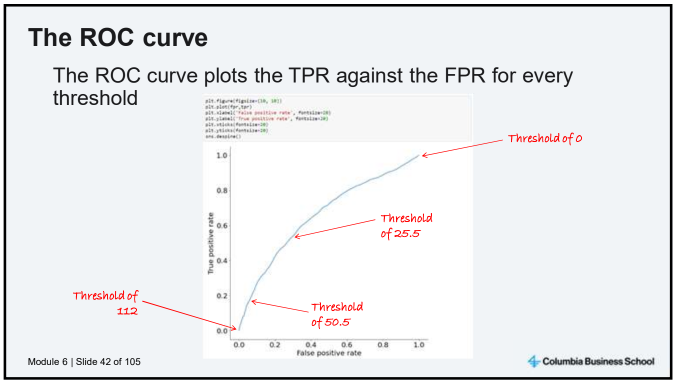
```
fpr, tpr, thresh = sk_m.roc_curve(df_tahoe.readmit30, df_tahoe['severity score'])
```

Every threshold of the score

The true positive rate for every threshold

The false positive rate for every threshold

The true outcomes are the first argument; don't switch them



The ROC curve

The ROC curve plots the TPR against the FPR for every threshold

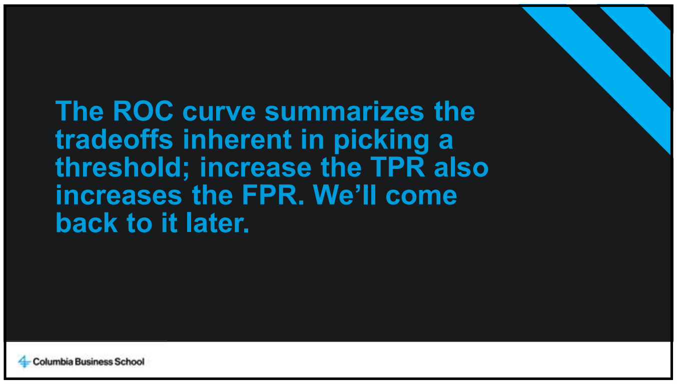
```
plt.figure(figsize=(10, 10))
plt.plot(fpr,tpr)
plt.xlabel('False positive rate', fontsize=20)
plt.ylabel('True positive rate', fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
sns.despine()
```

Threshold of 0

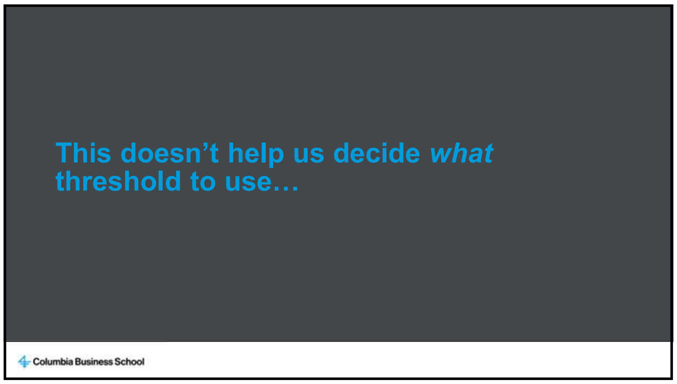
Threshold of 25.5

Threshold of 112

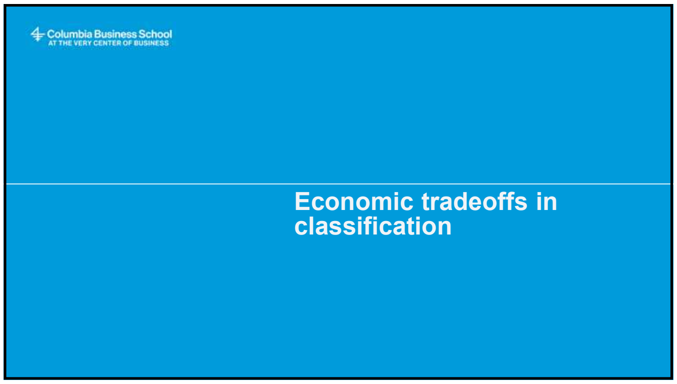
Threshold of 50.5



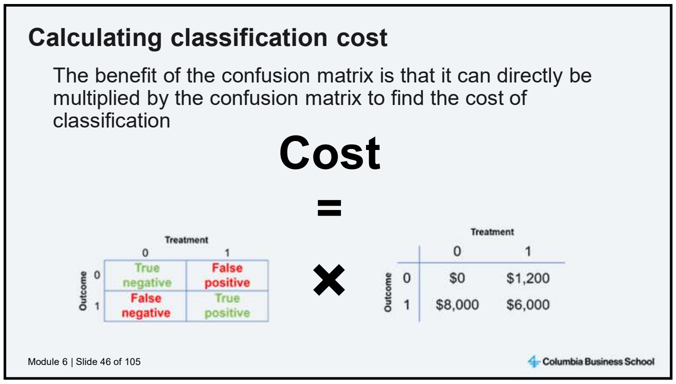
The ROC curve summarizes the tradeoffs inherent in picking a threshold; increase the TPR also increases the FPR. We'll come back to it later.



This doesn't help us decide *what* threshold to use…



Economic tradeoffs in classification



Calculating classification cost

The benefit of the confusion matrix is that it can directly be multiplied by the confusion matrix to find the cost of classification

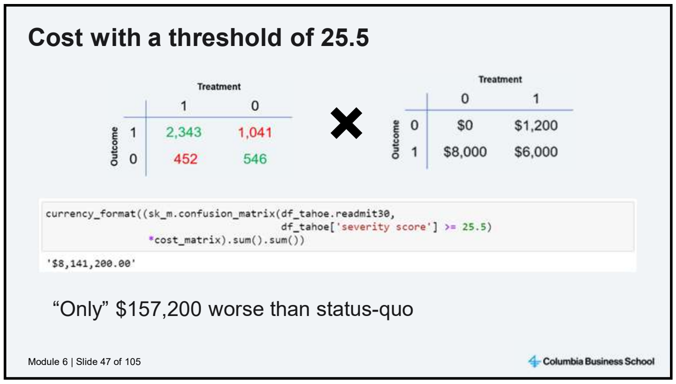
Cost

=

		Treatment	
		0	1
Outcome	0	True negative	False positive
	1	False negative	True positive

✖

		Treatment	
		0	1
Outcome	0	$0	$1,200
	1	$8,000	$6,000



Cost with a threshold of 25.5

		Treatment	
		1	0
Outcome	1	2,343	1,041
	0	452	546

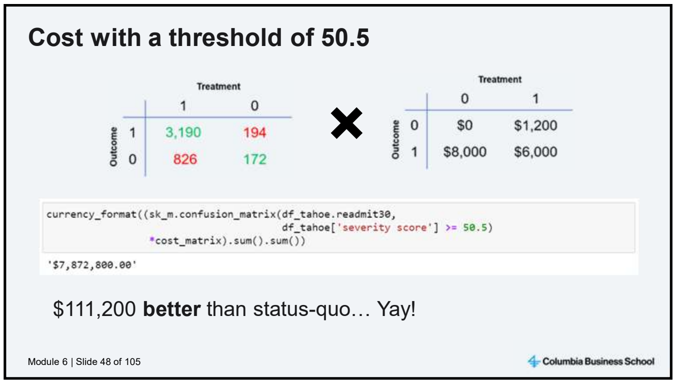
✖

		Treatment	
		0	1
Outcome	0	$0	$1,200
	1	$8,000	$6,000

```
currency_format((sk_m.confusion_matrix(df_tahoe.readmit30,
                                       df_tahoe['severity score'] >= 25.5)
                 *cost_matrix).sum().sum())
```

```
'$8,141,200.00'
```

"Only" $157,200 worse than status-quo



Cost with a threshold of 50.5

		Treatment	
		1	0
Outcome	1	3,190	194
	0	826	172

✖

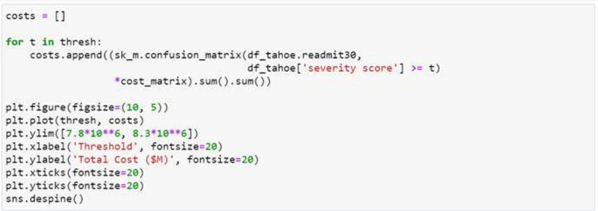
		Treatment	
		0	1
Outcome	0	$0	$1,200
	1	$8,000	$6,000

```
currency_format((sk_m.confusion_matrix(df_tahoe.readmit30,
                                       df_tahoe['severity score'] >= 50.5)
                 *cost_matrix).sum().sum())
```

```
'$7,872,800.00'
```

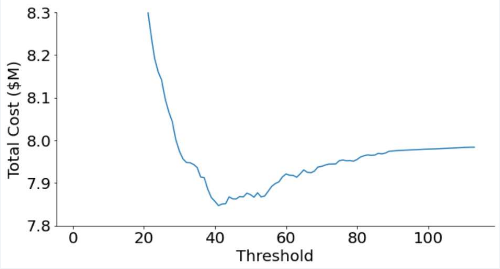
$111,200 **better** than status-quo… Yay!

Trying every threshold

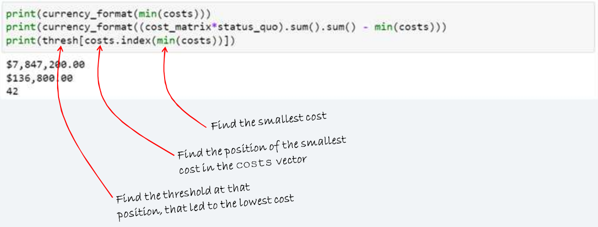


```
costs = []

for t in thresh:
    costs.append((sk_m.confusion_matrix(df_tahoe.readmit30,
                                        df_tahoe['severity score'] >= t)
                  *cost_matrix).sum().sum())

plt.figure(figsize=(10, 5))
plt.plot(thresh, costs)
plt.ylim([7.8*10**6, 8.3*10**6])
plt.xlabel('Threshold', fontsize=20)
plt.ylabel('Total Cost ($M)', fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
sns.despine()
```

Trying every threshold



Picking the best threshold



```
print(currency_format(min(costs)))
print(currency_format((cost_matrix*status_quo).sum().sum() - min(costs)))
print(thresh[costs.index(min(costs))])
```

```
$7,847,200.00
$136,800.00
42
```

Find the smallest cost

Find the position of the smallest cost in the costs vector

Find the threshold at that position, that led to the lowest cost

The optimal severity threshold is 42 with a net benefit over status quo of $136,800

Training/test sets

- It doesn't seem like we've "trained" a model, but in fact we have
- Picking the threshold of 42 is – in itself – a form of "training"
- It *could* be that this choice is "overfitting" to the data, and so in theory we should check the benefit of using this threshold on a test set
- That said, the model is so simple that it's really quite unlikely
- We will nevertheless shortly see what the test set performance looks like

Can we do better?

Using more complex classifiers

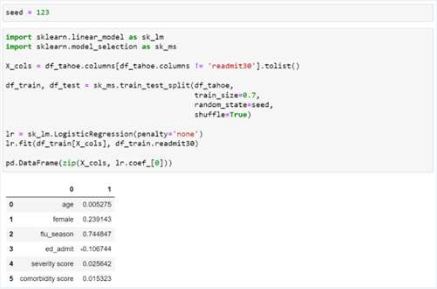
Logistic regression

- Severity is only one of the variables that could be used to carry out this classification
- But there are others – could we use all of them together?
- That is exactly what logistic regression allows us to do, by fitting the following model

$$P(\text{Re-admit}) = \frac{\exp(w)}{1+\exp(w)} = \frac{e^w}{1+e^w}$$

$$\text{with}\quad w = \beta_0 + \beta_1 \cdot \text{age} + \beta_2 \cdot \text{female} + \beta_3 \cdot \text{flu\_season} + \beta_4 \cdot \text{ed\_admit} + \beta_5 \cdot \text{severity} + \beta_6 \cdot \text{comorbidity}$$

Logistic regression (training set)

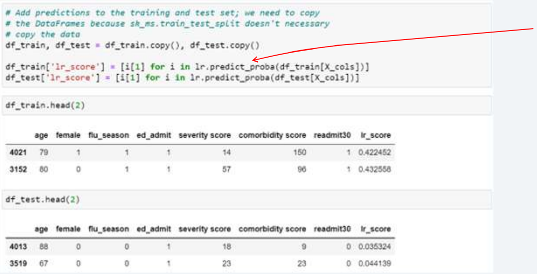


```
seed = 123
```

```
import sklearn.linear_model as sk_lm
import sklearn.model_selection as sk_ms

X_cols = df_tahoe.columns[df_tahoe.columns != 'readmit30'].tolist()

df_train, df_test = sk_ms.train_test_split(df_tahoe,
                                           train_size=0.7,
                                           random_state=seed,
                                           shuffle=True)

lr = sk_lm.LogisticRegression(penalty='none')
lr.fit(df_train[X_cols], df_train.readmit30)

pd.DataFrame(zip(X_cols, lr.coef_[0]))
```

	0	1
0	age	0.005275
1	female	0.239143
2	flu_season	0.744847
3	ed_admit	-0.106744
4	severity score	0.025642
5	comorbidity score	0.015323

Making predictions in the training and test set



```
# Add predictions to the training and test set; we need to copy
# the DataFrames because sk_ms.train_test_split doesn't necessary
# copy the data
df_train, df_test = df_train.copy(), df_test.copy()

df_train['lr_score'] = [i[1] for i in lr.predict_proba(df_train[X_cols])]
df_test['lr_score'] = [i[1] for i in lr.predict_proba(df_test[X_cols])]
```

```
df_train.head(2)
```

	age	female	flu_season	ed_admit	severity score	comorbidity score	readmit30	lr_score
4021	79	1	1	1	14	150	1	0.422452
3152	80	0	1	1	57	96	1	0.432558

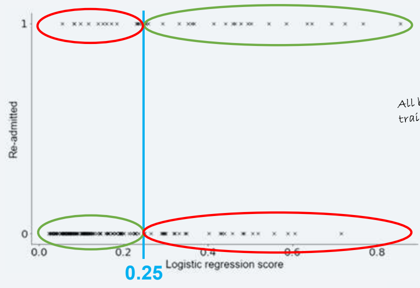
```
df_test.head(2)
```

	age	female	flu_season	ed_admit	severity score	comorbidity score	readmit30	lr_score
4013	88	0	0	1	18	9	0	0.035324
3519	67	0	0	1	23	23	0	0.044139

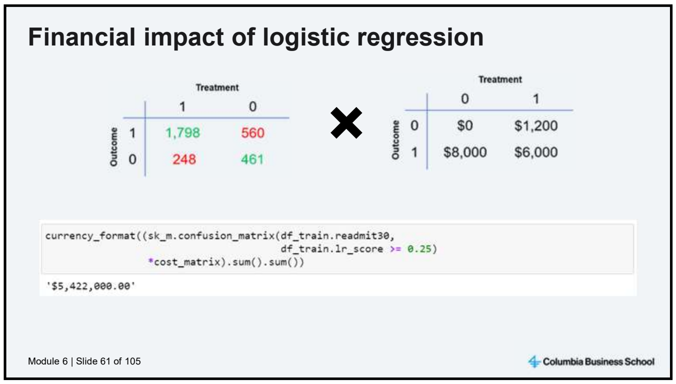
⚠ Do not use `.predict`

Any aspect of a model we train (whether the model or the threshold) needs to be chosen using the *training set*, and then evaluated on the *test set*

Picking a threshold for logistic regression



All based on the training set



Financial impact of logistic regression

		Treatment	
		1	0
Outcome	1	1,798	560
	0	248	461

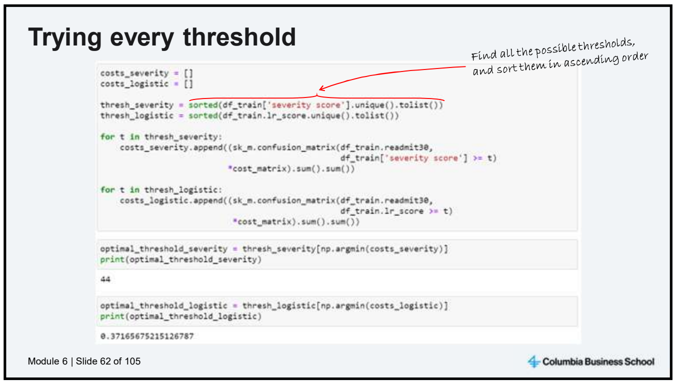
✖

		Treatment	
		0	1
Outcome	0	$0	$1,200
	1	$8,000	$6,000

```
currency_format((sk_m.confusion_matrix(df_train.readmit30,
                                       df_train.lr_score >= 0.25)
                 *cost_matrix).sum().sum())
```

```
'$5,422,000.00'
```

Columbia Business School



Trying every threshold

Find all the possible thresholds, and sort them in ascending order

```
costs_severity = []
costs_logistic = []

thresh_severity = sorted(df_train['severity score'].unique().tolist())
thresh_logistic = sorted(df_train.lr_score.unique().tolist())

for t in thresh_severity:
    costs_severity.append((sk_m.confusion_matrix(df_train.readmit30,
                                                  df_train['severity score'] >= t)
                           *cost_matrix).sum().sum())

for t in thresh_logistic:
    costs_logistic.append((sk_m.confusion_matrix(df_train.readmit30,
                                                  df_train.lr_score >= t)
                            *cost_matrix).sum().sum())
```

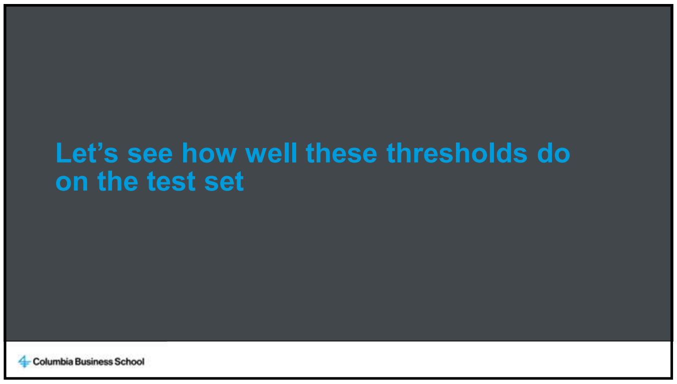
```
optimal_threshold_severity = thresh_severity[np.argmin(costs_severity)]
print(optimal_threshold_severity)
```

```
44
```

```
optimal_threshold_logistic = thresh_logistic[np.argmin(costs_logistic)]
print(optimal_threshold_logistic)
```

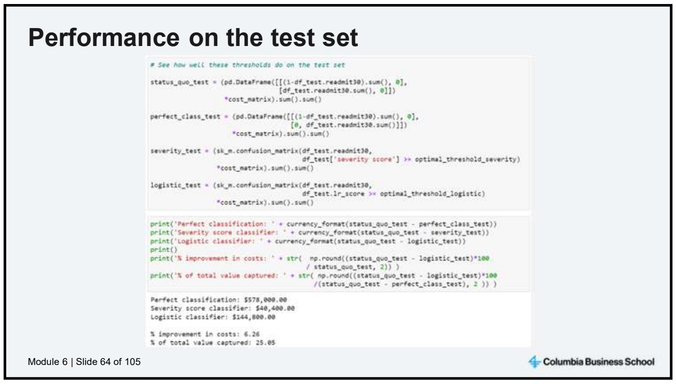
```
0.37165675215126787
```

Columbia Business School



Let’s see how well these thresholds do on the test set

Columbia Business School



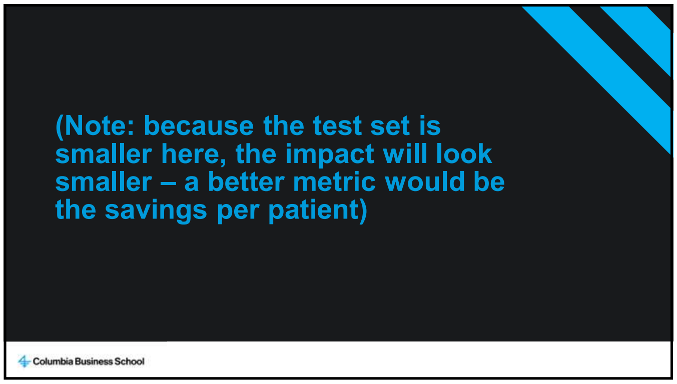
Performance on the test set

```
# See how well these thresholds do on the test set

status_quo_test = (pd.DataFrame([[(1-df_test.readmit30).sum(), 0],
                                 [df_test.readmit30.sum(), 0]])
                   *cost_matrix).sum().sum()

perfect_class_test = (pd.DataFrame([[(1-df_test.readmit30).sum(), 0],
                                    [0, df_test.readmit30.sum()]])
                      *cost_matrix).sum().sum()

severity_test = (sk_m.confusion_matrix(df_test.readmit30,
                                       df_test['severity score'] >= optimal_threshold_severity)
                 *cost_matrix).sum().sum()

logistic_test = (sk_m.confusion_matrix(df_test.readmit30,
                                       df_test.lr_score >= optimal_threshold_logistic)
                 *cost_matrix).sum().sum()
```

```
print('Perfect classification: ' + currency_format(status_quo_test - perfect_class_test))
print('Severity score classifier: ' + currency_format(status_quo_test - severity_test))
print('Logistic classifier: ' + currency_format(status_quo_test - logistic_test))
print()
print('% improvement in costs: ' + str(  np.round((status_quo_test - logistic_test)*100
                                         / status_quo_test, 2)) )
print('% of total value captured: ' + str( np.round((status_quo_test - logistic_test)*100
                                           /(status_quo_test - perfect_class_test), 2 )) )
```

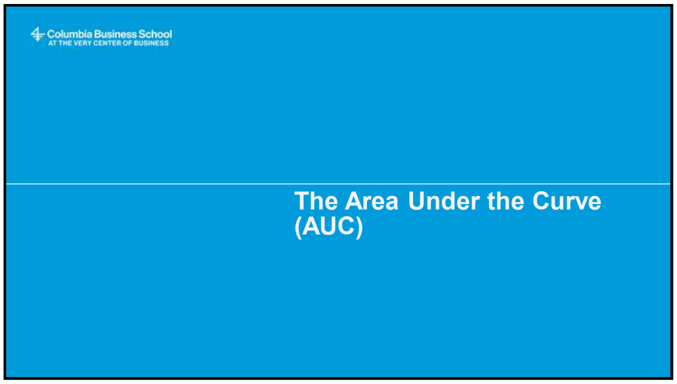
```
Perfect classification: $578,000.00
Severity score classifier: $40,400.00
Logistic classifier: $144,800.00

% improvement in costs: 6.26
% of total value captured: 25.05
```

Columbia Business School



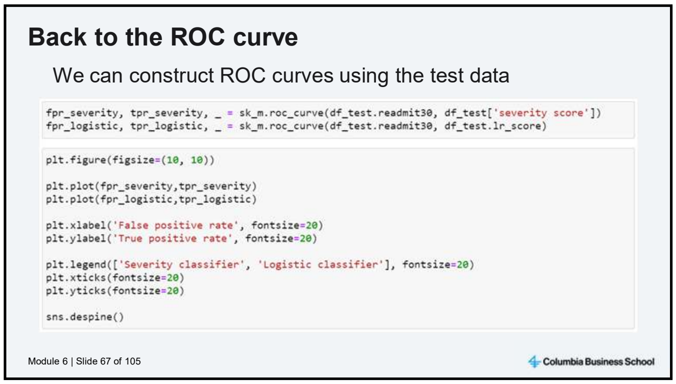
(Note: because the test set is smaller here, the impact will look smaller – a better metric would be the savings per patient)

Columbia Business School



Columbia Business School
AT THE VERY CENTER OF BUSINESS

The Area Under the Curve (AUC)

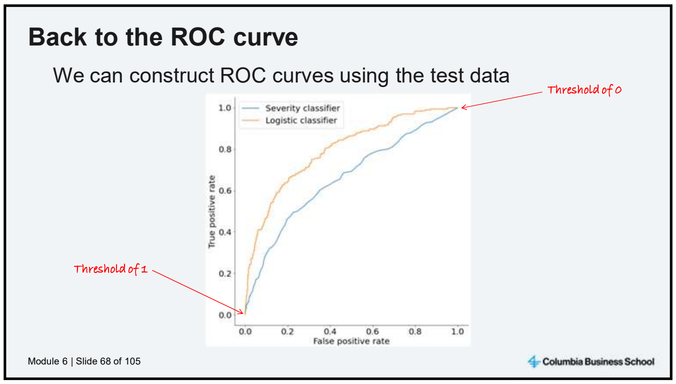


Back to the ROC curve

We can construct ROC curves using the test data

```
fpr_severity, tpr_severity, _ = sk_m.roc_curve(df_test.readmit30, df_test['severity score'])
fpr_logistic, tpr_logistic, _ = sk_m.roc_curve(df_test.readmit30, df_test.lr_score)
```

```
plt.figure(figsize=(10, 10))

plt.plot(fpr_severity,tpr_severity)
plt.plot(fpr_logistic,tpr_logistic)

plt.xlabel('False positive rate', fontsize=20)
plt.ylabel('True positive rate', fontsize=20)

plt.legend(['Severity classifier', 'Logistic classifier'], fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)

sns.despine()
```

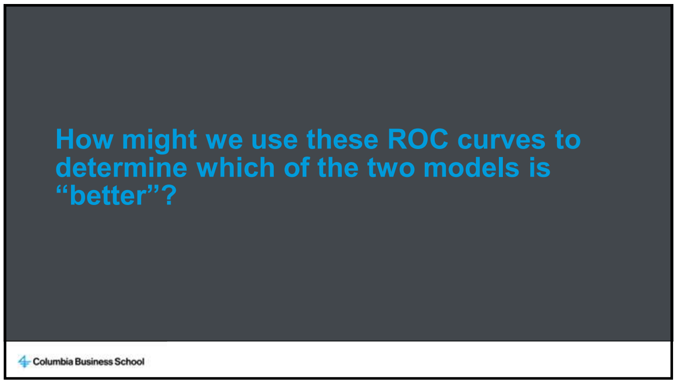


Back to the ROC curve

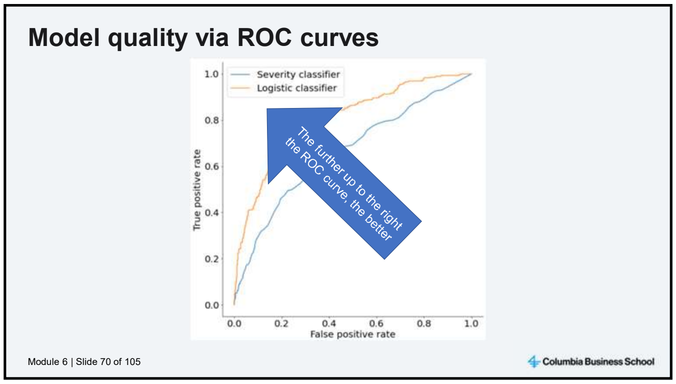
We can construct ROC curves using the test data

Threshold of 0

Threshold of 1

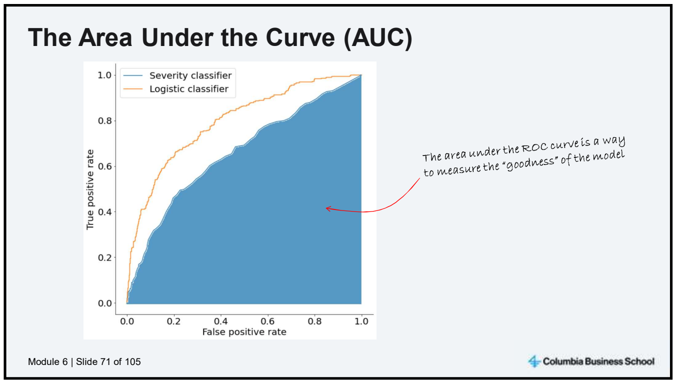


How might we use these ROC curves to determine which of the two models is "better"?



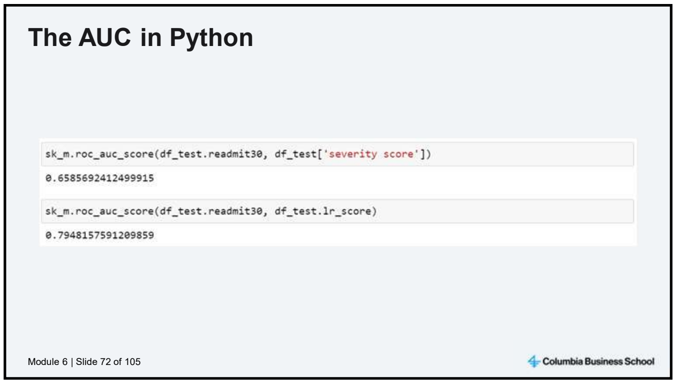
Model quality via ROC curves

The further up to the right the ROC curve, the better



The Area Under the Curve (AUC)

The area under the ROC curve is a way to measure the "goodness" of the model



The AUC in Python

```
sk_m.roc_auc_score(df_test.readmit30, df_test['severity score'])
0.6585692412499915
```

```
sk_m.roc_auc_score(df_test.readmit30, df_test.lr_score)
0.7948157591209859
```

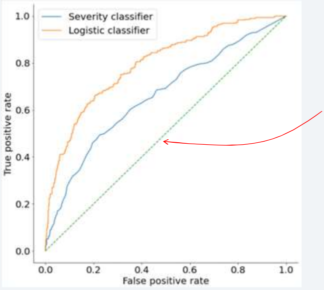
What is the smallest possible value the AUC could take?

In other words, what is the worst possible classification model, and what AUC would it achieve?

A “random score” model

- The worst imaginable model just assigns a random score between 0 and 1 to every data point
- What would the ROC curve look like for such a model?
- Suppose we set the threshold at 0.5
  - Half the true positives will be classified as positive, half the true negatives will be classified as negative
  - So FPR = TPR = 0.5
- Suppose we set the threshold at 0.7
  - FPR = TPR = 0.3
- etc…

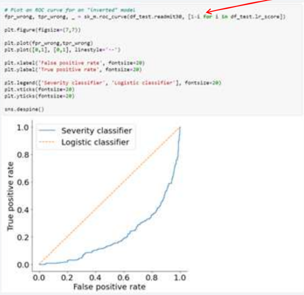
A “random score” model



Random model – AUC = 0.5. If AUC < 0.5, the positive and negative label have just been inverted

A model done wrong

Notice how the score is inverted

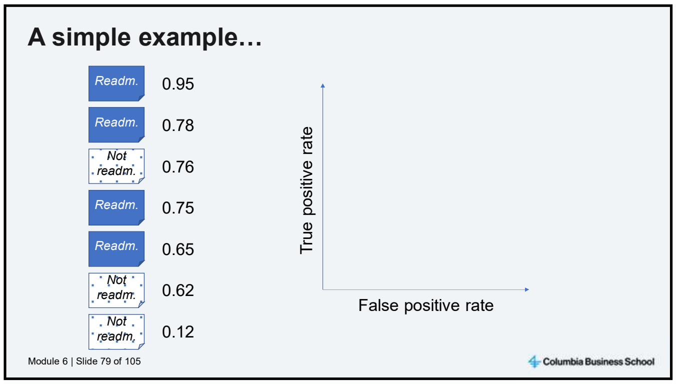


```
# Plot an ROC curve for an "inverted" model
fpr_wrong, tpr_wrong, _ = sk_m.roc_curve(df_test.readmit30, [1-i for i in df_test.lr_score])

plt.figure(figsize=(7,7))

plt.plot(fpr_wrong,tpr_wrong)
plt.plot([0,1], [0,1], linestyle='--')

plt.xlabel('False positive rate', fontsize=20)
plt.ylabel('True positive rate', fontsize=20)

plt.legend(['Severity classifier', 'Logistic classifier'], fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)

sns.despine()
```

Columbia Business School
AT THE VERY CENTER OF BUSINESS

Understanding the AUC

The “area under the curve” definition of the AUC makes sense, but what does it actually *mean* in practice?

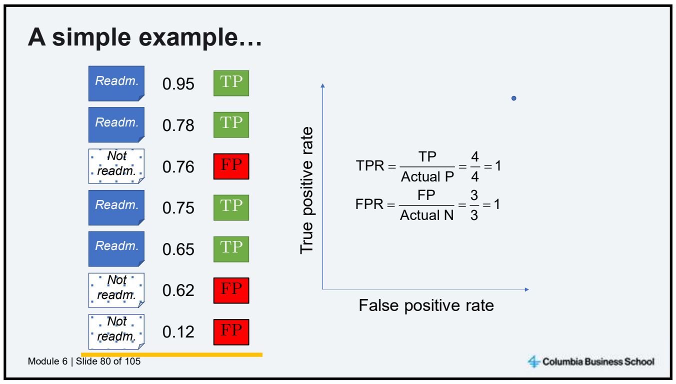


A simple example…

Readm.	0.95	
Readm.	0.78	
Not readm.	0.76	
Readm.	0.75	
Readm.	0.65	
Not readm.	0.62	
Not readm.	0.12	

Module 6 | Slide 79 of 105

Columbia Business School



A simple example…

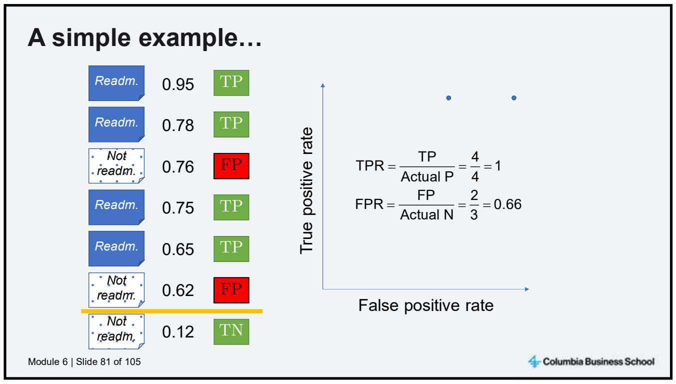
Readm.	0.95	TP
Readm.	0.78	TP
Not readm.	0.76	FP
Readm.	0.75	TP
Readm.	0.65	TP
Not readm.	0.62	FP
Not readm.	0.12	FP

$$TPR = \frac{TP}{Actual\ P} = \frac{4}{4} = 1$$

$$FPR = \frac{FP}{Actual\ N} = \frac{3}{3} = 1$$

Module 6 | Slide 80 of 105

Columbia Business School



A simple example…

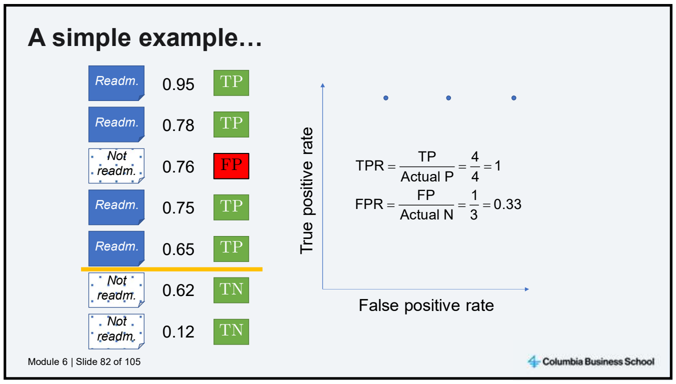
Readm.	0.95	TP
Readm.	0.78	TP
Not readm.	0.76	FP
Readm.	0.75	TP
Readm.	0.65	TP
Not readm.	0.62	FP
Not readm.	0.12	TN

$$TPR = \frac{TP}{Actual\ P} = \frac{4}{4} = 1$$

$$FPR = \frac{FP}{Actual\ N} = \frac{2}{3} = 0.66$$

Module 6 | Slide 81 of 105

Columbia Business School



A simple example…

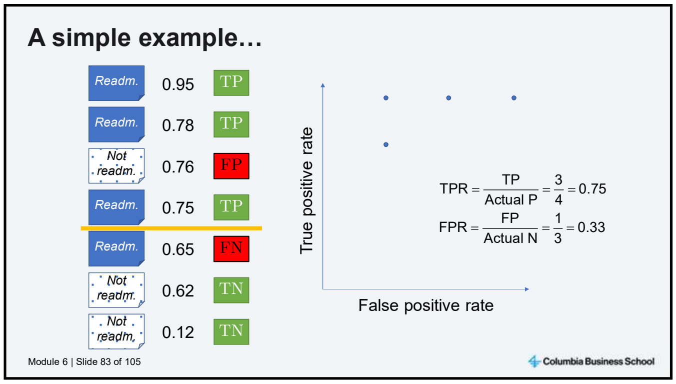
Readm.	0.95	TP
Readm.	0.78	TP
Not readm.	0.76	FP
Readm.	0.75	TP
Readm.	0.65	TP
Not readm.	0.62	TN
Not readm.	0.12	TN

$$TPR = \frac{TP}{Actual\ P} = \frac{4}{4} = 1$$

$$FPR = \frac{FP}{Actual\ N} = \frac{1}{3} = 0.33$$

Module 6 | Slide 82 of 105

Columbia Business School



A simple example…

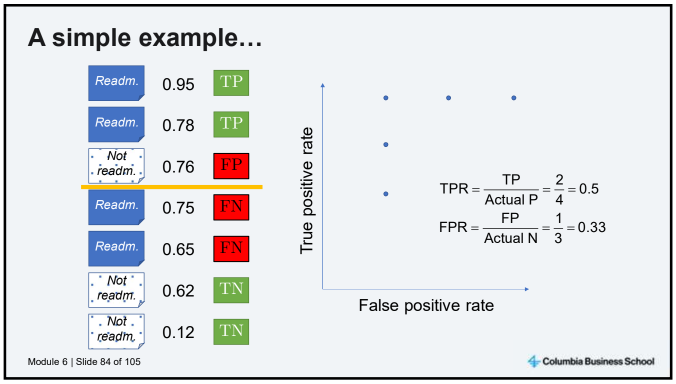
Readm.	0.95	TP
Readm.	0.78	TP
Not readm.	0.76	FP
Readm.	0.75	TP
Readm.	0.65	FN
Not readm.	0.62	TN
Not readm.	0.12	TN

$$TPR = \frac{TP}{Actual\ P} = \frac{3}{4} = 0.75$$

$$FPR = \frac{FP}{Actual\ N} = \frac{1}{3} = 0.33$$

Module 6 | Slide 83 of 105

Columbia Business School



A simple example…

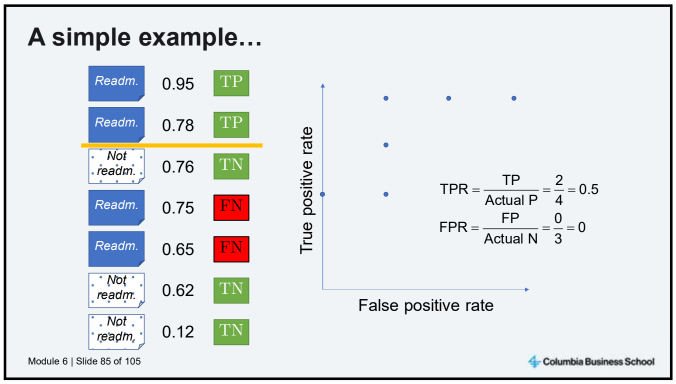
Readm.	0.95	TP
Readm.	0.78	TP
Not readm.	0.76	FP
Readm.	0.75	FN
Readm.	0.65	FN
Not readm.	0.62	TN
Not readm.	0.12	TN

$$TPR = \frac{TP}{Actual\ P} = \frac{2}{4} = 0.5$$

$$FPR = \frac{FP}{Actual\ N} = \frac{1}{3} = 0.33$$

Module 6 | Slide 84 of 105

Columbia Business School

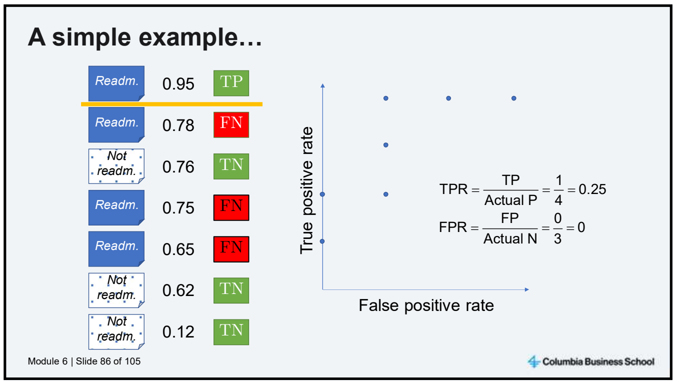


A simple example…

Readm.	0.95	TP
Readm.	0.78	TP
Not readm.	0.76	TN
Readm.	0.75	FN
Readm.	0.65	FN
Not readm.	0.62	TN
Not readm.	0.12	TN

$$TPR = \frac{TP}{\text{Actual P}} = \frac{2}{4} = 0.5$$

$$FPR = \frac{FP}{\text{Actual N}} = \frac{0}{3} = 0$$

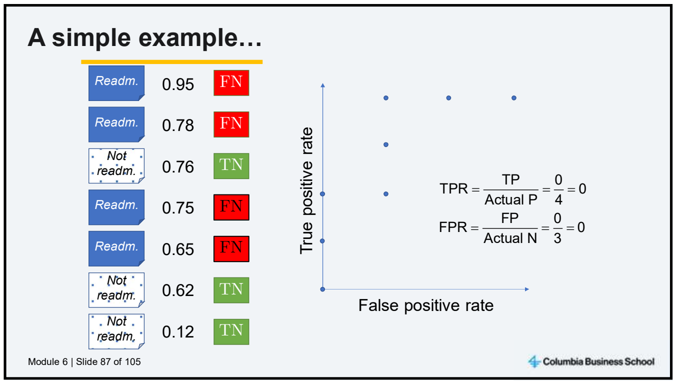


A simple example…

Readm.	0.95	TP
Readm.	0.78	FN
Not readm.	0.76	TN
Readm.	0.75	FN
Readm.	0.65	FN
Not readm.	0.62	TN
Not readm.	0.12	TN

$$TPR = \frac{TP}{\text{Actual P}} = \frac{1}{4} = 0.25$$

$$FPR = \frac{FP}{\text{Actual N}} = \frac{0}{3} = 0$$

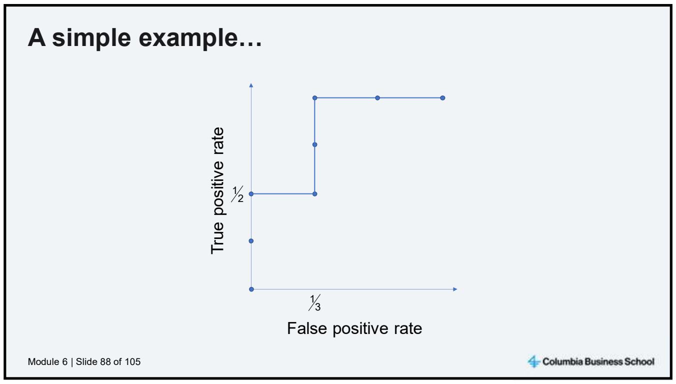


A simple example…

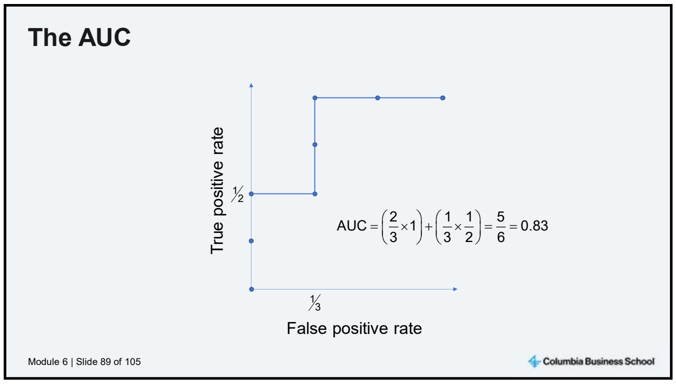
Readm.	0.95	FN
Readm.	0.78	FN
Not readm.	0.76	TN
Readm.	0.75	FN
Readm.	0.65	FN
Not readm.	0.62	TN
Not readm.	0.12	TN

$$TPR = \frac{TP}{\text{Actual P}} = \frac{0}{4} = 0$$

$$FPR = \frac{FP}{\text{Actual N}} = \frac{0}{3} = 0$$

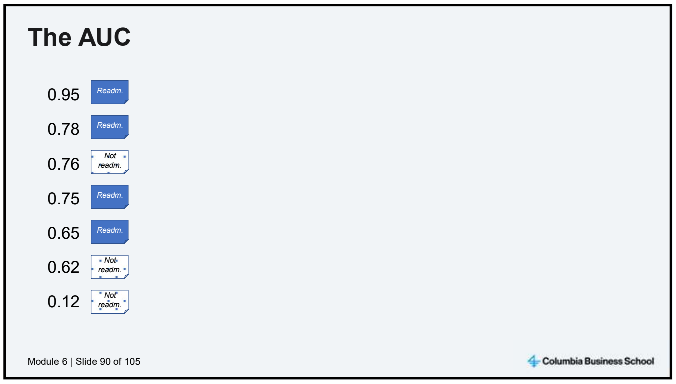


A simple example…



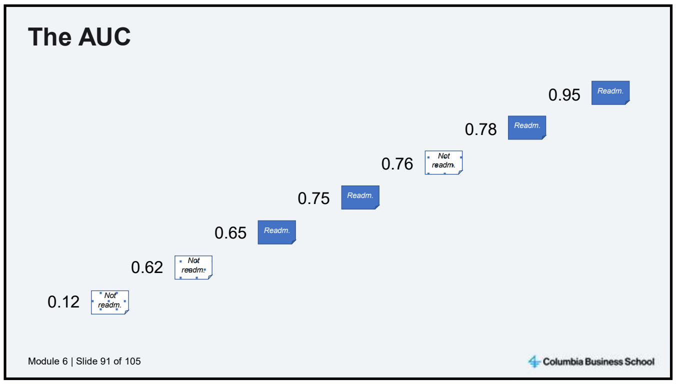
The AUC

$$AUC = \left(\frac{2}{3}\times 1\right) + \left(\frac{1}{3}\times\frac{1}{2}\right) = \frac{5}{6} = 0.83$$



The AUC

0.95	Readm.
0.78	Readm.
0.76	Not readm.
0.75	Readm.
0.65	Readm.
0.62	Not readm.
0.12	Not readm.

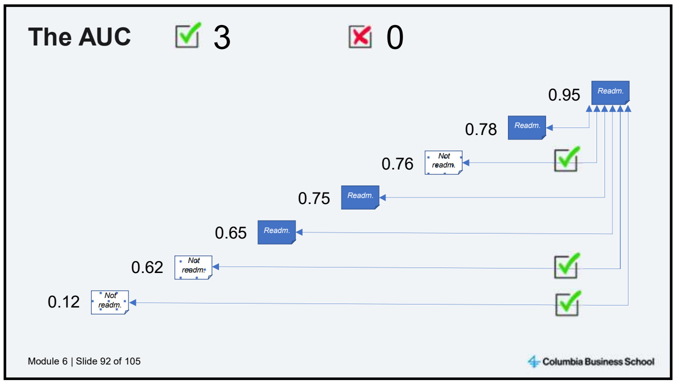


The AUC

0.95 Readm.
0.78 Readm.
0.76 Not readm.
0.75 Readm.
0.65 Readm.
0.62 Not readm.
0.12 Not readm.

Module 6 | Slide 91 of 105

Columbia Business School

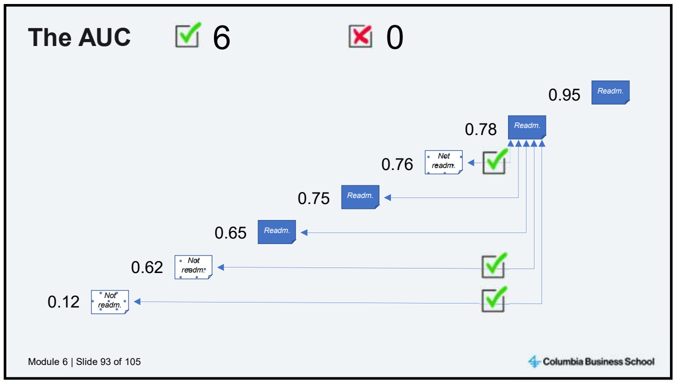


The AUC ☑ 3 ☒ 0

0.95 Readm.
0.78 Readm.
0.76 Not readm. ✓
0.75 Readm.
0.65 Readm.
0.62 Not readm. ✓
0.12 Not readm. ✓

Module 6 | Slide 92 of 105

Columbia Business School

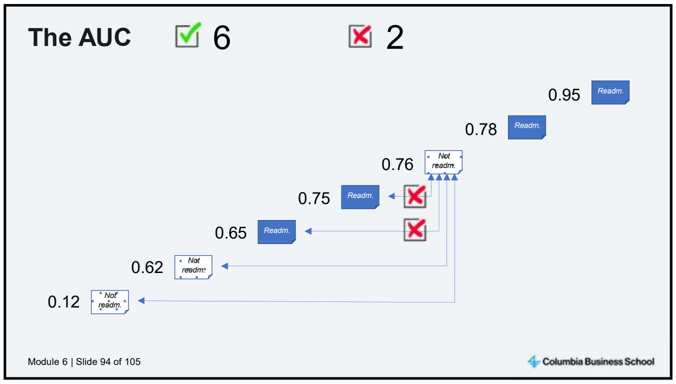


The AUC ☑ 6 ☒ 0

0.95 Readm.
0.78 Readm.
0.76 Not readm. ✓
0.75 Readm.
0.65 Readm.
0.62 Not readm. ✓
0.12 Not readm. ✓

Module 6 | Slide 93 of 105

Columbia Business School

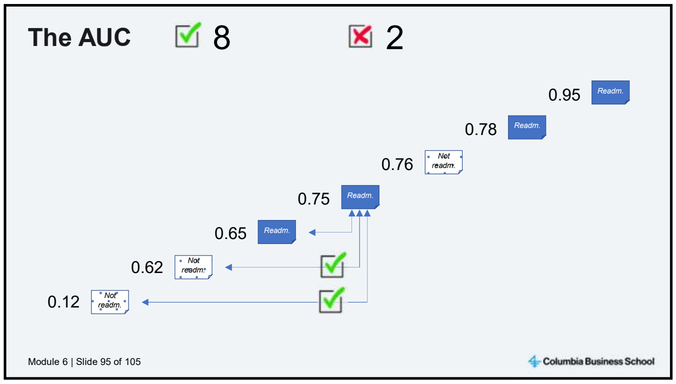


The AUC ☑ 6 ☒ 2

0.95 Readm.
0.78 Readm.
0.76 Not readm.
0.75 Readm. ✗
0.65 Readm. ✗
0.62 Not readm.
0.12 Not readm.

Module 6 | Slide 94 of 105

Columbia Business School

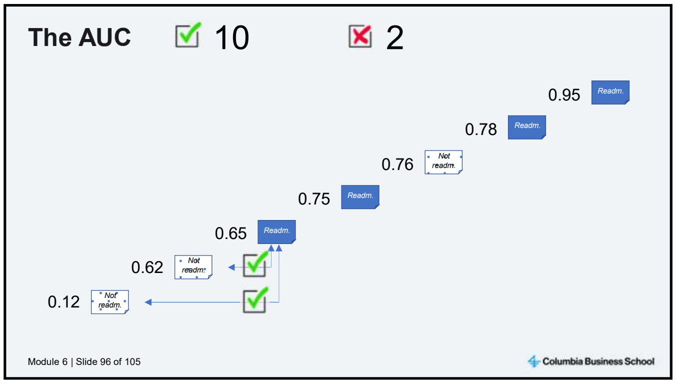


The AUC ☑ 8 ☒ 2

0.95 Readm.
0.78 Readm.
0.76 Not readm.
0.75 Readm.
0.65 Readm.
0.62 Not readm. ✓
0.12 Not readm. ✓

Module 6 | Slide 95 of 105

Columbia Business School

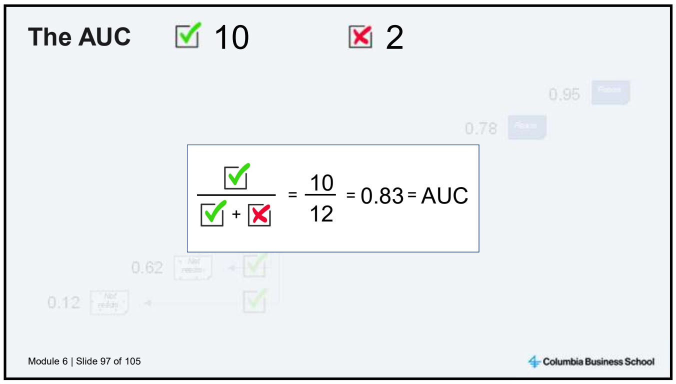


The AUC ☑ 10 ☒ 2

0.95 Readm.
0.78 Readm.
0.76 Not readm.
0.75 Readm.
0.65 Readm.
0.62 Not readm. ✓
0.12 Not readm. ✓

Module 6 | Slide 96 of 105

Columbia Business School



The AUC ☑ 10 ☒ 2

$$\frac{\boxed{\checkmark}}{\boxed{\checkmark}+\boxed{\times}} = \frac{10}{12} = 0.83 = \text{AUC}$$

A mathematical explanation

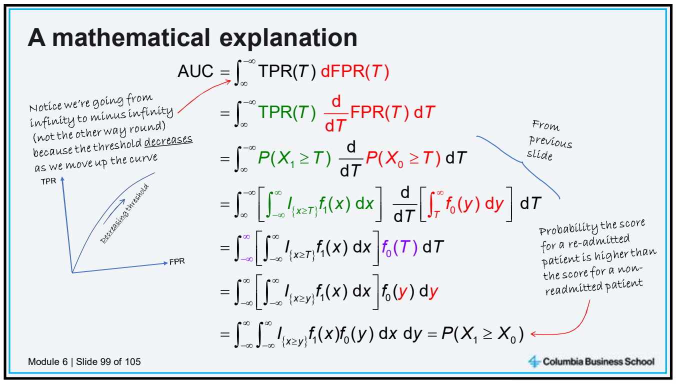
Define the following notation

- Let X_1 be a random variable denoting the model score of a re-admitted patient and X_0 be a random variable denoting the model score of a non-readmitted patient (p.d.f.s f_1 and f_0)
- Let TPR(T) and FPR(T) be the true positive rate and false positive rate when the threshold is T. Convince yourself that

$$\text{TPR}(T) = P(X_1 \geq T) = \int_{-\infty}^{\infty} I_{\{x \geq T\}} f_1(x)\,\mathrm{d}x$$

$$\text{FPR}(T) = P(X_0 \geq T) = \int_{T}^{\infty} f_0(y)\,\mathrm{d}y$$

Even though these are similar expressions, we're writing them slightly differently for reasons that will become obvious



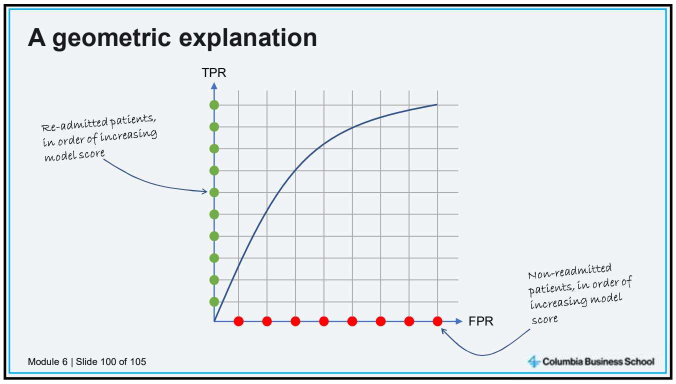
A mathematical explanation

$$\begin{aligned}
\text{AUC} &= \int_{\infty}^{-\infty} \text{TPR}(T)\,\mathrm{dFPR}(T) \\
&= \int_{\infty}^{-\infty} \text{TPR}(T)\,\frac{\mathrm{d}}{\mathrm{d}T}\text{FPR}(T)\,\mathrm{d}T \\
&= \int_{\infty}^{-\infty} P(X_1 \geq T)\,\frac{\mathrm{d}}{\mathrm{d}T}P(X_0 \geq T)\,\mathrm{d}T \\
&= \int_{\infty}^{-\infty} \left[\int_{-\infty}^{\infty} I_{\{x \geq T\}} f_1(x)\,\mathrm{d}x\right] \frac{\mathrm{d}}{\mathrm{d}T}\left[\int_{T}^{\infty} f_0(y)\,\mathrm{d}y\right] \mathrm{d}T \\
&= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} I_{\{x \geq T\}} f_1(x)\,\mathrm{d}x\right] f_0(T)\,\mathrm{d}T \\
&= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} I_{\{x \geq y\}} f_1(x)\,\mathrm{d}x\right] f_0(y)\,\mathrm{d}y \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I_{\{x \geq y\}} f_1(x) f_0(y)\,\mathrm{d}x\,\mathrm{d}y = P(X_1 \geq X_0)
\end{aligned}$$

Notice we're going from infinity to minus infinity (not the other way round) because the threshold decreases as we move up the curve

From previous slide

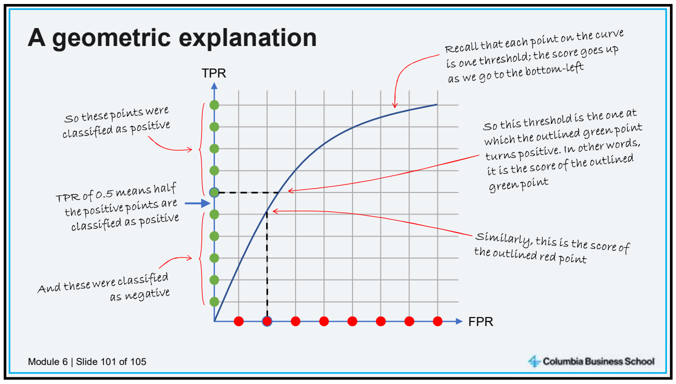
Probability the score for a re-admitted patient is higher than the score for a non-readmitted patient



A geometric explanation

Re-admitted patients, in order of increasing model score

Non-readmitted patients, in order of increasing model score



A geometric explanation

Recall that each point on the curve is one threshold; the score goes up as we go to the bottom-left

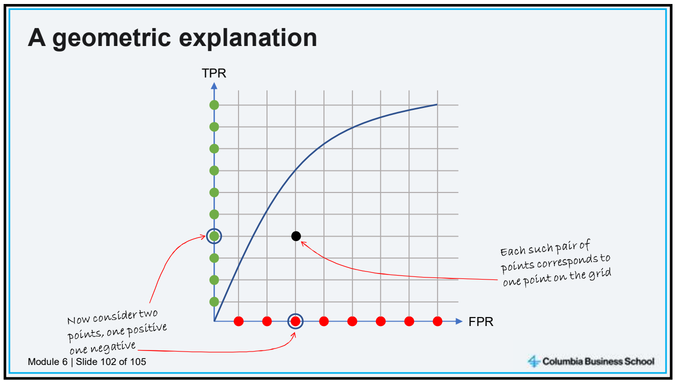
So these points were classified as positive

TPR of 0.5 means half the positive points are classified as positive

And these were classified as negative

So this threshold is the one at which the outlined green point turns positive. In other words, it is the score of the outlined green point

Similarly, this is the score of the outlined red point

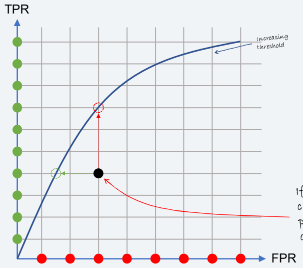


A geometric explanation

Each such pair of points corresponds to one point on the grid

Now consider two points, one positive one negative

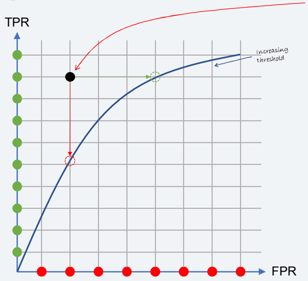
A geometric explanation



If the point is <u>below</u> the ROC curve, the score of the green point is <u>larger</u> than the score of the red point; this is good!

Columbia Business School

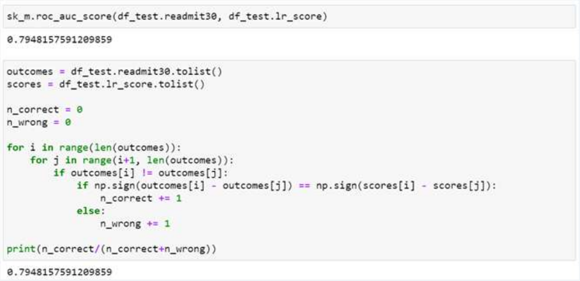
A geometric explanation



If the point is <u>above</u> the ROC curve, the score of the green point is <u>smaller</u> than the score of the red point; this is bad!

Columbia Business School

Verifying the probabilistic interpretation



```
sk_m.roc_auc_score(df_test.readmit30, df_test.lr_score)
```

```
0.7948157591209859
```

```
outcomes = df_test.readmit30.tolist()
scores = df_test.lr_score.tolist()

n_correct = 0
n_wrong = 0

for i in range(len(outcomes)):
    for j in range(i+1, len(outcomes)):
        if outcomes[i] != outcomes[j]:
            if np.sign(outcomes[i] - outcomes[j]) == np.sign(scores[i] - scores[j]):
                n_correct += 1
            else:
                n_wrong += 1

print(n_correct/(n_correct+n_wrong))
```

```
0.7948157591209859
```

Columbia Business School

Columbia Business School
AT THE VERY CENTER OF BUSINESS

Fall 2024

Skill vs. Luck: Sports Analytics

Module 7

Professor Daniel Guetta

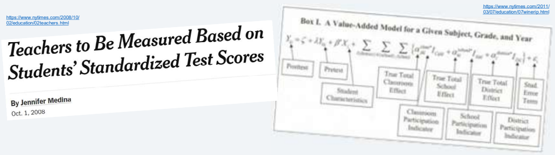
This Module

- Harnessing your competitive instincts
- Identifying skill versus luck
- Shrinkage estimators to predict future performance

Columbia Business School
AT THE VERY CENTER OF BUSINESS

Introduction – value-added teacher evaluations in New York City

Value-added teacher evaluations in New York City



https://www.nytimes.com/2008/10/02/education/02teachers.html

Teachers to Be Measured Based on Students' Standardized Test Scores

By Jennifer Medina
Oct. 1, 2008

https://www.nytimes.com/2011/03/07/education/07winerip.html

Box I. A Value-Added Model for a Given Subject, Grade, and Year

Posttest
Pretest
Student Characteristics
True Total Classroom Effect
True Total School Effect
True Total District Effect
Stud. Error Term
Classroom Participation Indicator
School Participation Indicator
District Participation Indicator

"To avoid a contentious fight with the teachers' union, the New York City Department of Education has agreed not to make public the reports… 'They won't be used in tenure determinations or the annual rating process' "

Despite the best intentions…

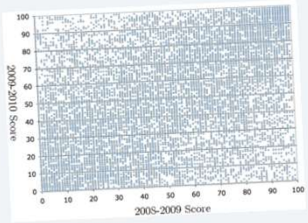
The Washington Post
Democracy Dies in Darkness

NYC releases teachers' value-added scores – unfortunately

By Valerie Strauss
February 24, 2012

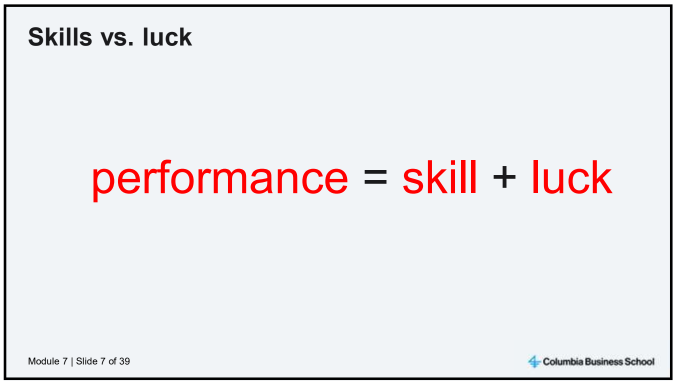
https://www.washingtonpost.com/blogs/answer-sheet/post/nyc-releases-teachers-value-added-scores--unfortunately/2012/02/24/gIQAtbVXYR_blog.html

One issue with these scores…



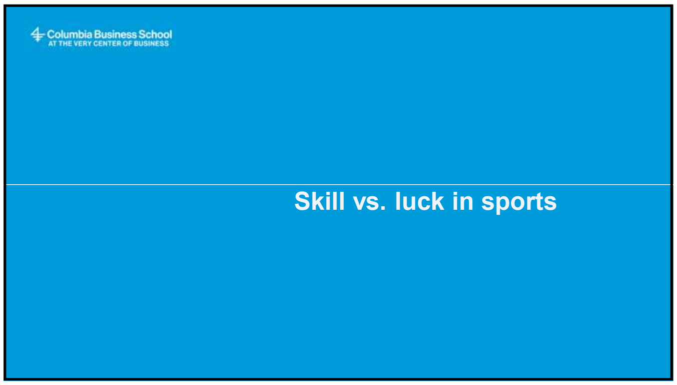
Each point is one teacher; the correlation is 0.35. These scores seem to be the result of *luck* as much as the teacher's intrinsic *skill*.

Source: Gary Rubinstein



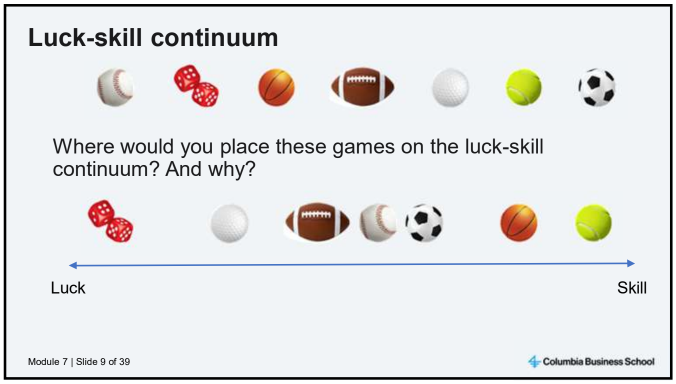
Skills vs. luck

performance = skill + luck



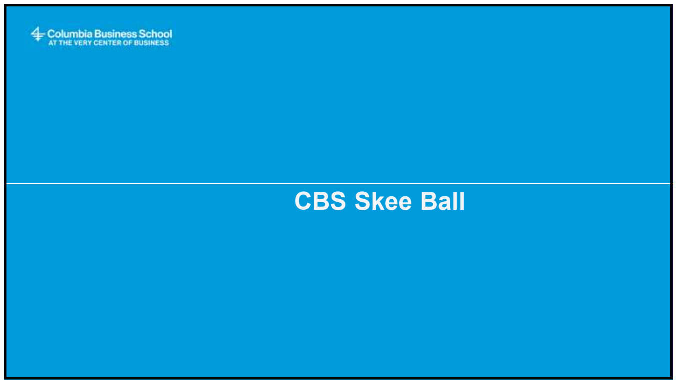
Columbia Business School
AT THE VERY CENTER OF BUSINESS

Skill vs. luck in sports



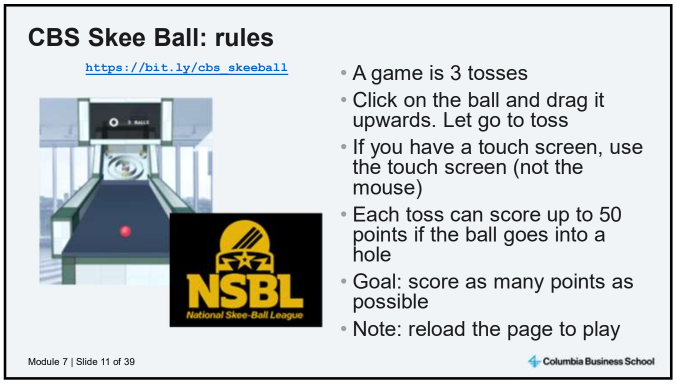
Luck-skill continuum

Where would you place these games on the luck-skill continuum? And why?



Columbia Business School
AT THE VERY CENTER OF BUSINESS

CBS Skee Ball

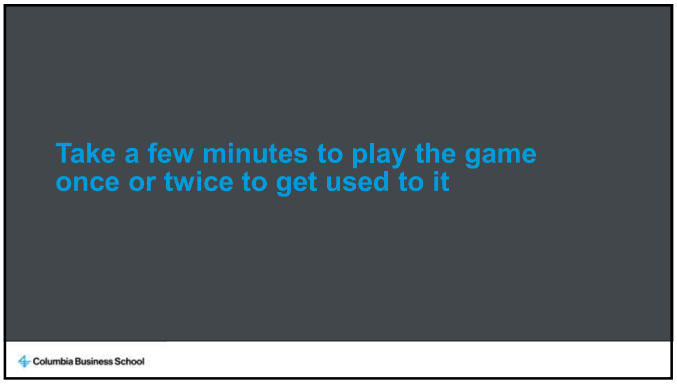


CBS Skee Ball: rules

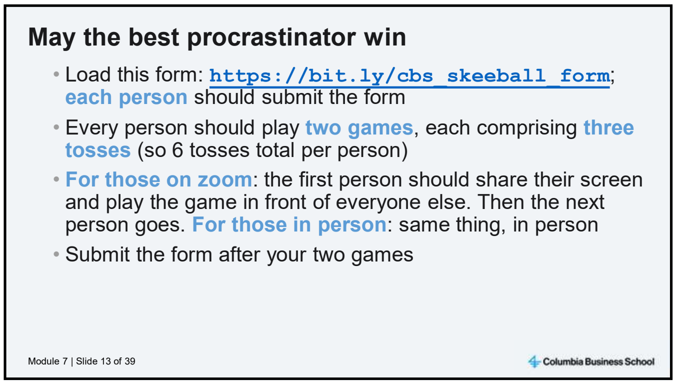
https://bit.ly/cbs_skeeball

NSBL
National Skee-Ball League

- A game is 3 tosses
- Click on the ball and drag it upwards. Let go to toss
- If you have a touch screen, use the touch screen (not the mouse)
- Each toss can score up to 50 points if the ball goes into a hole
- Goal: score as many points as possible
- Note: reload the page to play

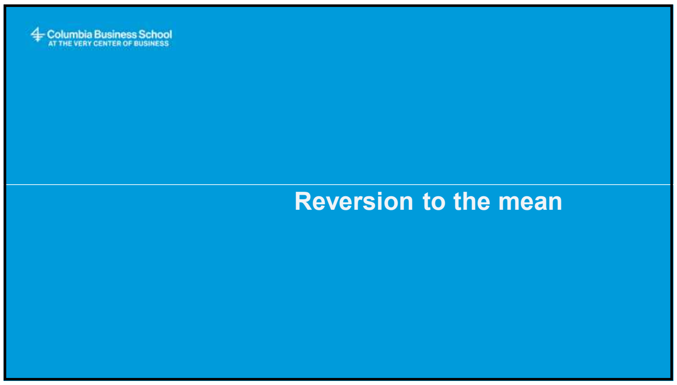


Take a few minutes to play the game once or twice to get used to it

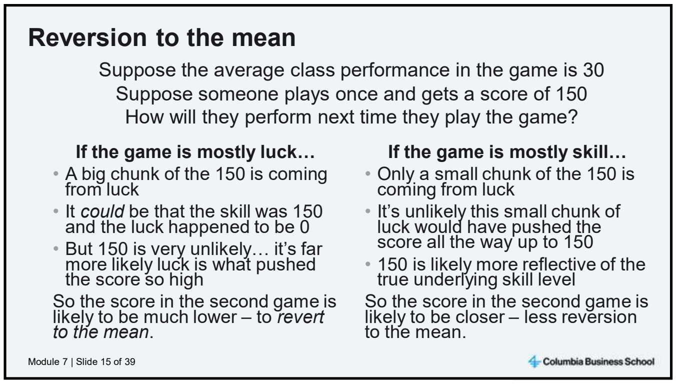


May the best procrastinator win

- Load this form: `https://bit.ly/cbs_skeeball_form`; **each person** should submit the form
- Every person should play **two games**, each comprising **three tosses** (so 6 tosses total per person)
- **For those on zoom**: the first person should share their screen and play the game in front of everyone else. Then the next person goes. **For those in person**: same thing, in person
- Submit the form after your two games



Reversion to the mean



Reversion to the mean

Suppose the average class performance in the game is 30
Suppose someone plays once and gets a score of 150
How will they perform next time they play the game?

If the game is mostly luck…

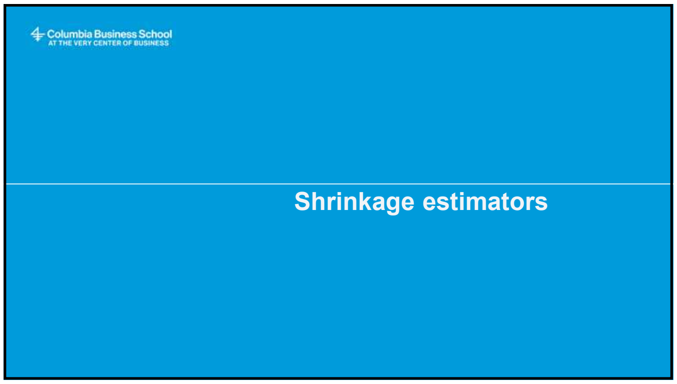
- A big chunk of the 150 is coming from luck
- It *could* be that the skill was 150 and the luck happened to be 0
- But 150 is very unlikely… it's far more likely luck is what pushed the score so high

So the score in the second game is likely to be much lower – to *revert to the mean*.

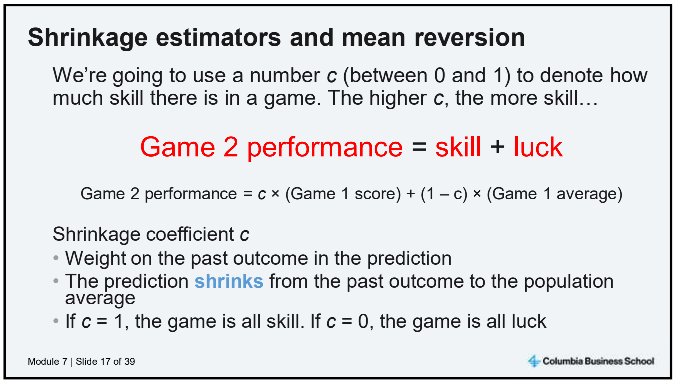
If the game is mostly skill…

- Only a small chunk of the 150 is coming from luck
- It's unlikely this small chunk of luck would have pushed the score all the way up to 150
- 150 is likely more reflective of the true underlying skill level

So the score in the second game is likely to be closer – less reversion to the mean.



Shrinkage estimators



Shrinkage estimators and mean reversion

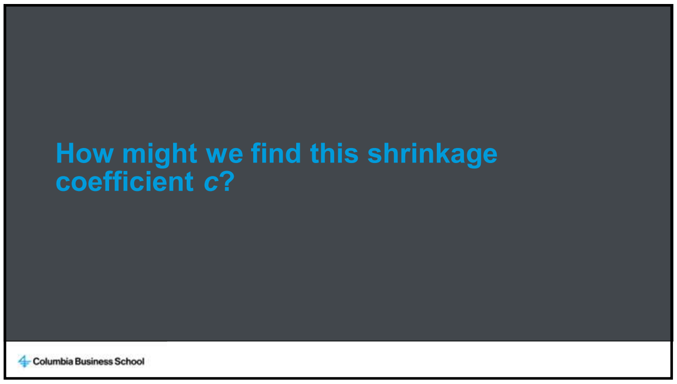
We're going to use a number *c* (between 0 and 1) to denote how much skill there is in a game. The higher *c*, the more skill…

Game 2 performance = skill + luck

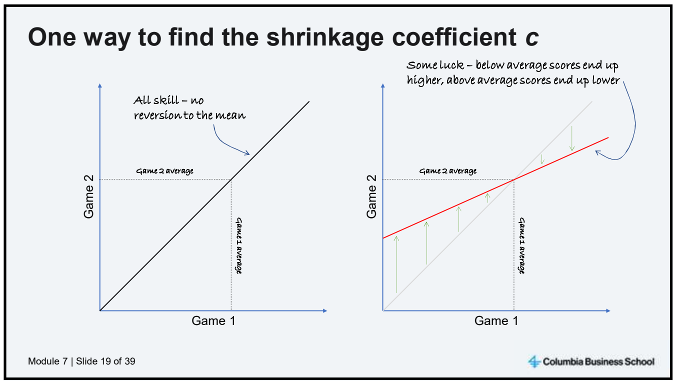
Game 2 performance = *c* × (Game 1 score) + (1 – c) × (Game 1 average)

Shrinkage coefficient *c*

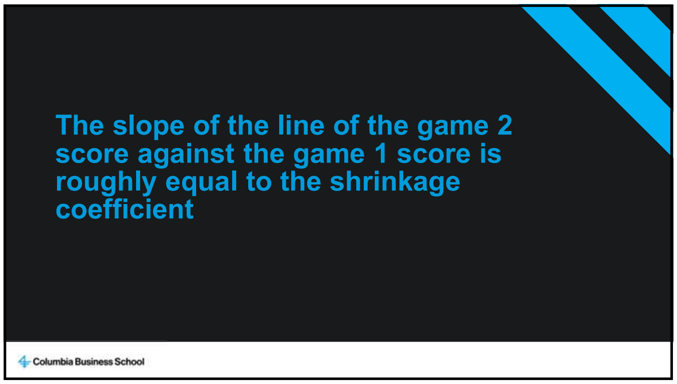
- Weight on the past outcome in the prediction
- The prediction **shrinks** from the past outcome to the population average
- If *c* = 1, the game is all skill. If *c* = 0, the game is all luck



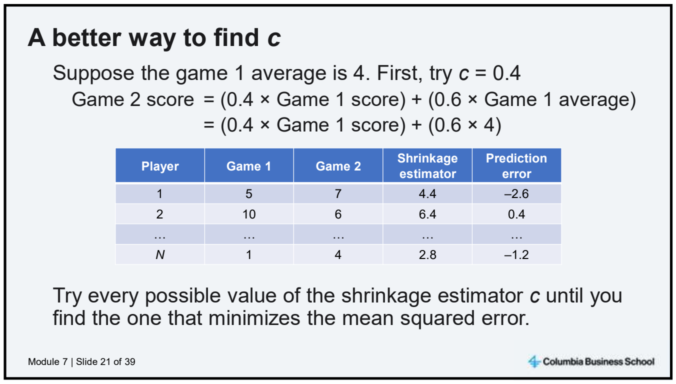
How might we find this shrinkage coefficient *c*?



One way to find the shrinkage coefficient *c*



The slope of the line of the game 2 score against the game 1 score is roughly equal to the shrinkage coefficient



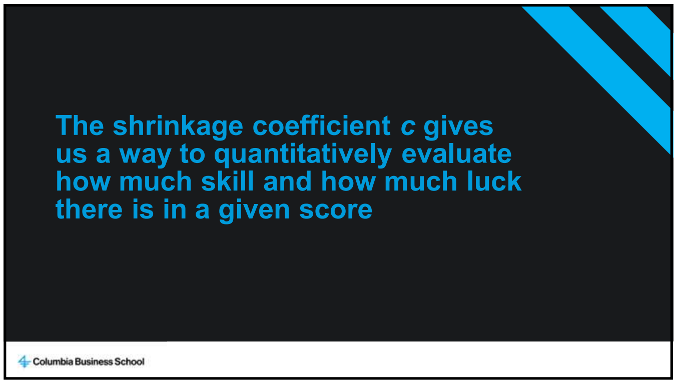
A better way to find *c*

Suppose the game 1 average is 4. First, try *c* = 0.4

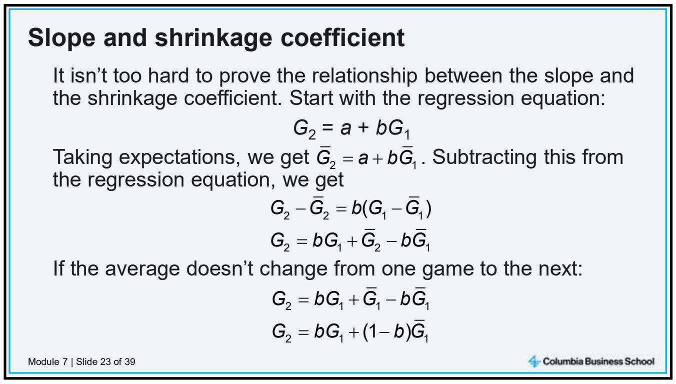
Game 2 score = (0.4 × Game 1 score) + (0.6 × Game 1 average)
= (0.4 × Game 1 score) + (0.6 × 4)

Player	Game 1	Game 2	Shrinkage estimator	Prediction error
1	5	7	4.4	–2.6
2	10	6	6.4	0.4
…	…	…	…	…
N	1	4	2.8	–1.2

Try every possible value of the shrinkage estimator *c* until you find the one that minimizes the mean squared error.



The shrinkage coefficient *c* gives us a way to quantitatively evaluate how much skill and how much luck there is in a given score



Slope and shrinkage coefficient

It isn't too hard to prove the relationship between the slope and the shrinkage coefficient. Start with the regression equation:

$$G_2 = a + bG_1$$

Taking expectations, we get $\bar{G}_2 = a + b\bar{G}_1$. Subtracting this from the regression equation, we get

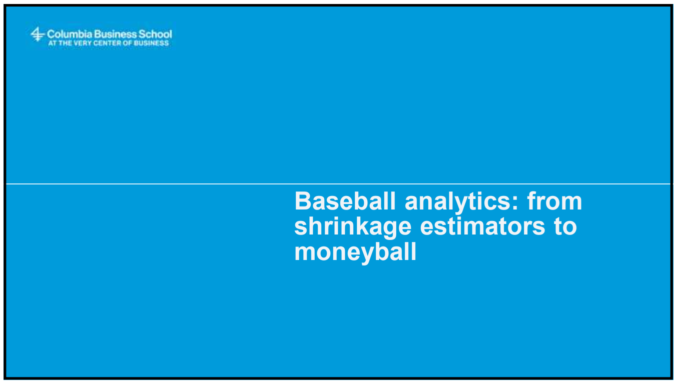
$$G_2 - \bar{G}_2 = b(G_1 - \bar{G}_1)$$

$$G_2 = bG_1 + \bar{G}_2 - b\bar{G}_1$$

If the average doesn't change from one game to the next:

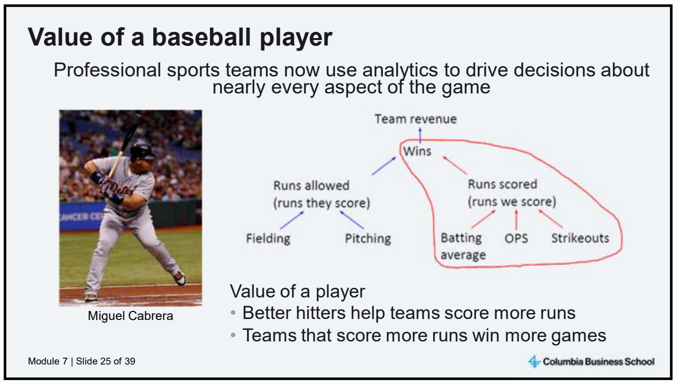
$$G_2 = bG_1 + \bar{G}_1 - b\bar{G}_1$$

$$G_2 = bG_1 + (1-b)\bar{G}_1$$



Columbia Business School
AT THE VERY CENTER OF BUSINESS

Baseball analytics: from shrinkage estimators to moneyball



Value of a baseball player

Professional sports teams now use analytics to drive decisions about nearly every aspect of the game

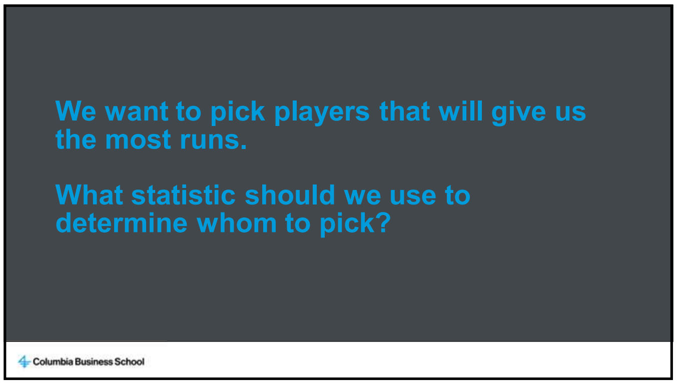
Miguel Cabrera

Value of a player

- Better hitters help teams score more runs
- Teams that score more runs win more games

Module 7 | Slide 25 of 39

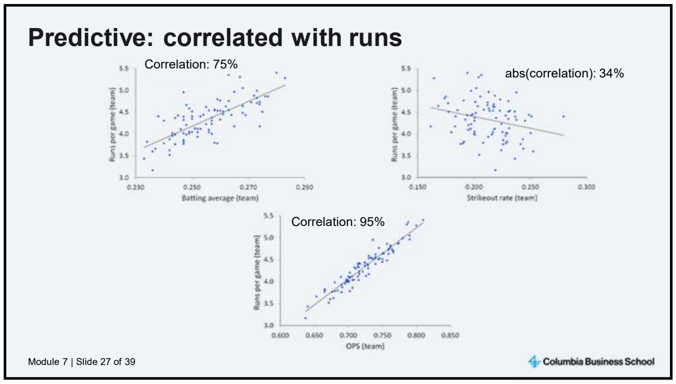
Columbia Business School



We want to pick players that will give us the most runs.

What statistic should we use to determine whom to pick?

Columbia Business School



Predictive: correlated with runs

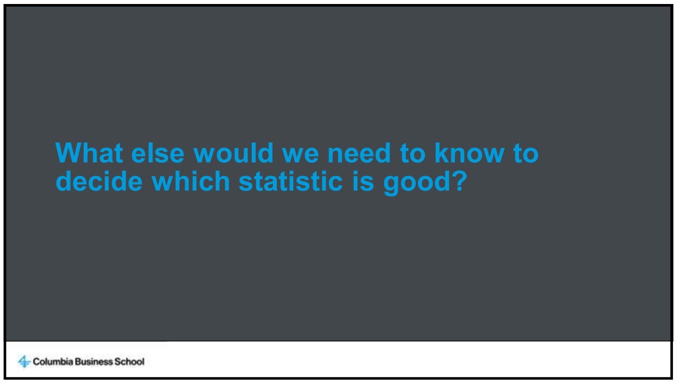
Correlation: 75%

abs(correlation): 34%

Correlation: 95%

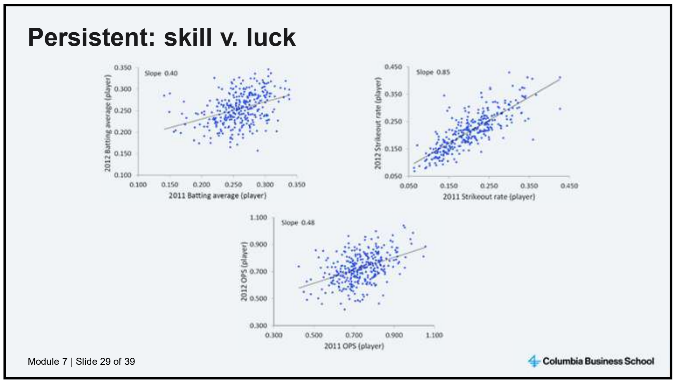
Module 7 | Slide 27 of 39

Columbia Business School



What else would we need to know to decide which statistic is good?

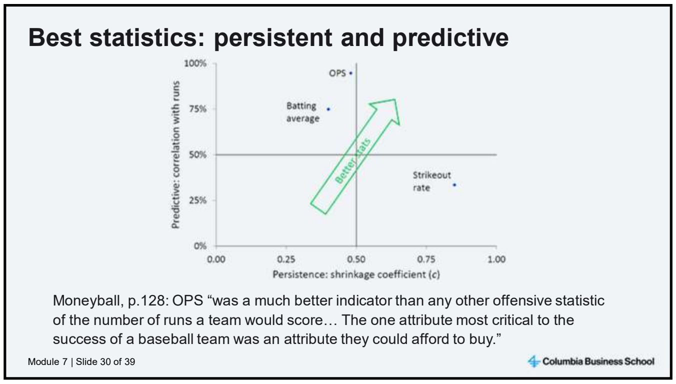
Columbia Business School



Persistent: skill v. luck

Module 7 | Slide 29 of 39

Columbia Business School

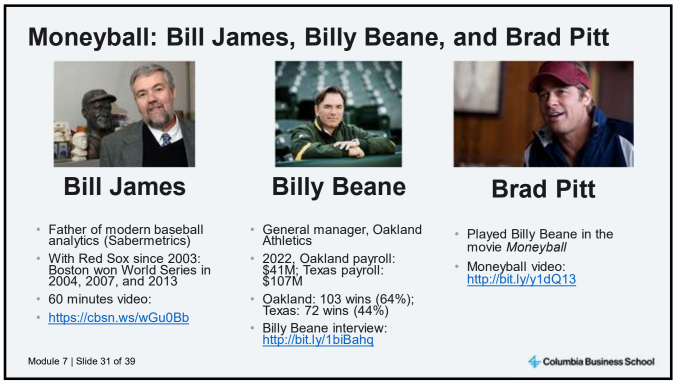


Best statistics: persistent and predictive

Moneyball, p.128: OPS “was a much better indicator than any other offensive statistic of the number of runs a team would score… The one attribute most critical to the success of a baseball team was an attribute they could afford to buy.”

Module 7 | Slide 30 of 39

Columbia Business School



Moneyball: Bill James, Billy Beane, and Brad Pitt

Bill James

- Father of modern baseball analytics (Sabermetrics)
- With Red Sox since 2003: Boston won World Series in 2004, 2007, and 2013
- 60 minutes video:
- https://cbsn.ws/wGu0Bb

Billy Beane

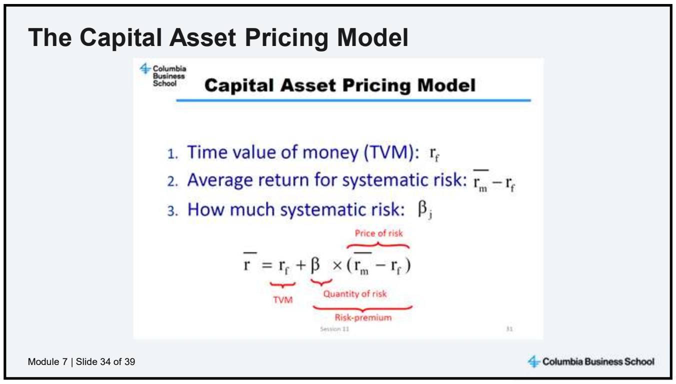
- General manager, Oakland Athletics
- 2022, Oakland payroll: $41M; Texas payroll: $107M
- Oakland: 103 wins (64%); Texas: 72 wins (44%)
- Billy Beane interview: http://bit.ly/1biBahq

Brad Pitt

- Played Billy Beane in the movie *Moneyball*
- Moneyball video: http://bit.ly/y1dQ13

Moneyball

Other applications of shrinkage estimators: predicting future stock *β*



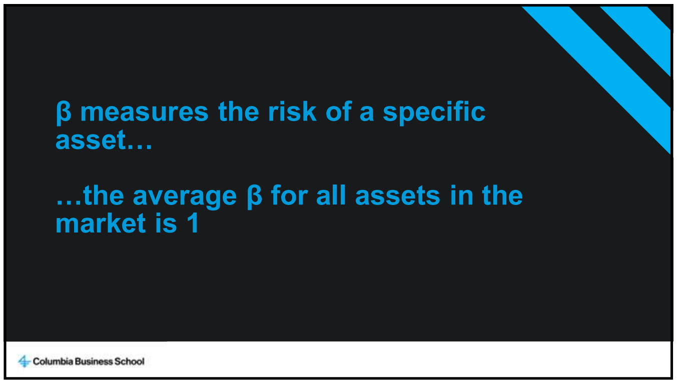
The Capital Asset Pricing Model

Capital Asset Pricing Model

1. Time value of money (TVM): r_f
2. Average return for systematic risk: $\overline{r_m} - r_f$
3. How much systematic risk: β_j

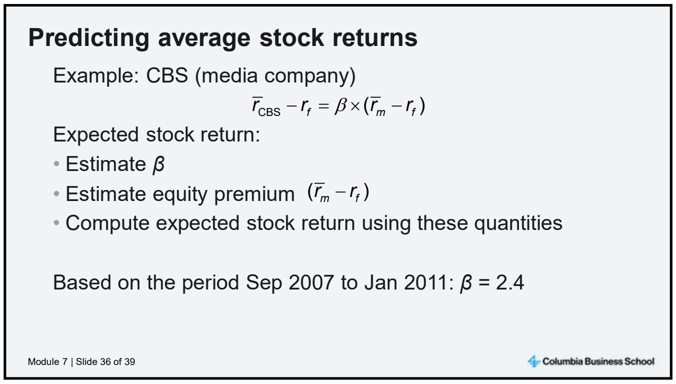
$$\overline{r} = \underbrace{r_f}_{\text{TVM}} + \underbrace{\underbrace{\beta}_{\text{Quantity of risk}} \times \overbrace{(\overline{r_m} - r_f)}^{\text{Price of risk}}}_{\text{Risk-premium}}$$

Session 11 31



β measures the risk of a specific asset…

…the average β for all assets in the market is 1



Predicting average stock returns

Example: CBS (media company)

$$\overline{r}_{CBS} - r_f = \beta \times (\overline{r}_m - r_f)$$

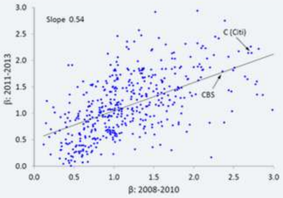
Expected stock return:

- Estimate β
- Estimate equity premium $(\overline{r}_m - r_f)$
- Compute expected stock return using these quantities

Based on the period Sep 2007 to Jan 2011: β = 2.4

Can we assume this β is representative of the future β we might observe?

Predicing β: shrinkage estimator

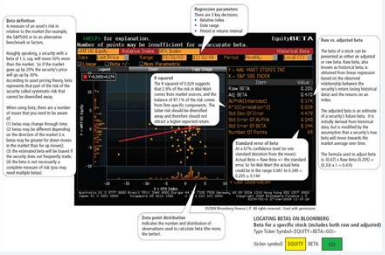


c^* = 0.54
minimizes the MSE

$$\begin{aligned}\text{Predicted } \beta \text{ for CBS } &= c \times (\text{Observed } \beta) + (1 - c) \times (\text{Average } \beta) \\ &= 0.54 \times 2.4 + 0.46 \times \mathit{1.0} \\ &= 1.8\end{aligned}$$

Bloomberg (Merrill Lynch) adjusted β

$$\text{Adjusted } \beta = (2/3) \times \text{Raw } \beta + (1/3) \times 1.0$$



Beta definition
A measure of an asset's risk in relation to the market (for example, the S&P500) or to an alternative benchmark or factors.

Roughly speaking, a security with a beta of 1.5, say, will move 50% more than the market. So if the market goes up by 20% the security's price will go up by 30%.
According to asset pricing theory, beta represents that part of the risk of the security called systematic risk that cannot be diversified away.

When using beta, there are a number of issues that you need to be aware of:
(1) betas may change through time;
(2) betas may be different depending on the direction of the market (i.e. betas may be greater for down moves in the market than for up moves);
(3) the estimated beta will be biased if the security does not frequently trade;
(4) the beta is not necessarily a complete measure of risk (you may need multiple betas).

Regression parameters
There are 3 key decisions:
- Relative index
- Date range
- Period or returns interval

<HELP> for explanation.
Number of points may be insufficient for an accurate beta.
EquityBETA

Y = WAL-MART STORES INC
X = S&P 500 INDEX

Item	Value
Raw BETA	0.205
Adj BETA	0.470
ALPHA(Intercept)	0.174
R^2(Correlation^2)	0.029
Std Dev Of Error	4.479
Std Error Of ALPHA	0.540
Std Error Of BETA	0.144
Number Of Points	60

R squared
The R squared of 0.029 suggests that 2.9% of the risk in Wal-Mart comes from market sources, and the balance of 97.1% of the risk comes from firm specific components. The latter risk should be diversified away and therefore should not attract a higher expected return.

Standard error of beta
At a 67% confidence level (or one standard deviation from the mean):
Actual Beta = Raw Beta +/- the standard error. So for Wal-Mart the actual beta could be in the range 0.061 to 0.349 = 0.205 ± 0.144

Raw vs. adjusted beta
The beta of a stock can be presented as either an adjusted or raw beta. Raw beta, also known as historical beta, is obtained from linear regression based on the observed relationship between the security's return (using historical data) and the returns on an index.

The adjusted beta is an estimate of a security's future beta. It is initially derived from historical data, but is modified by the assumption that a security's true beta will move towards the market average over time.

The formula used to adjust beta is: (0.67) x Raw Beta (0.205) + (0.33) x 1 = 0.470

Data point distribution
Indicates the number and distribution of observations used to calculate beta (the more, the better).

©2009 Bloomberg Finance L.P. All rights reserved. Used with permission.

LOCATING BETAS ON BLOOMBERG
Beta for a specific stock (includes both raw and adjusted)
Type Ticker Symbol<EQUITY>BETA<GO>

[ticker symbol] EQUITY BETA GO

Fall 2024

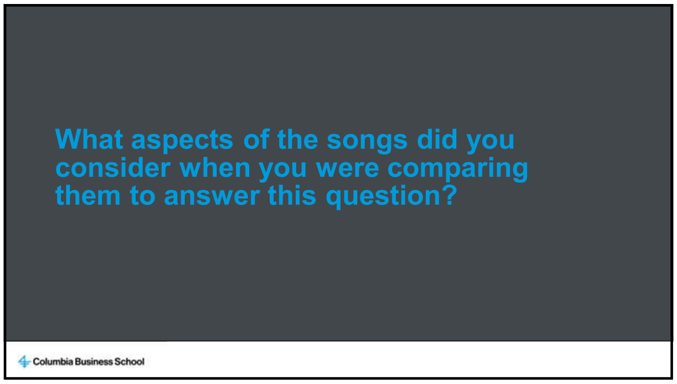
Recommendation Analytics; Music Streaming Services

Module 8

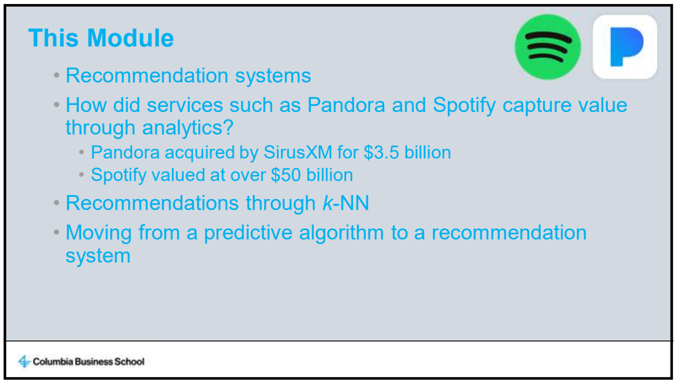
Professor Daniel Guetta

A musical start…

I'm going to play two songs, and then ask you how similar you think they are…

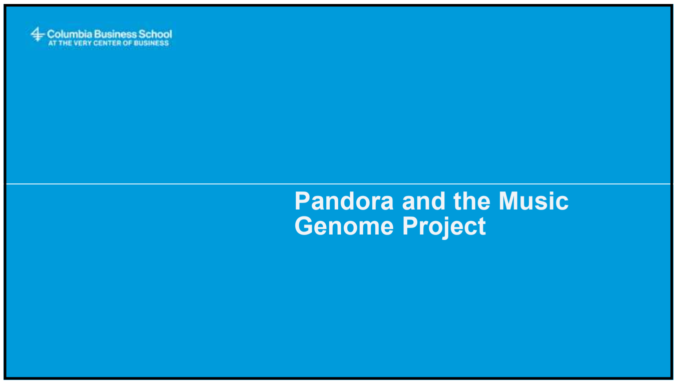


What aspects of the songs did you consider when you were comparing them to answer this question?

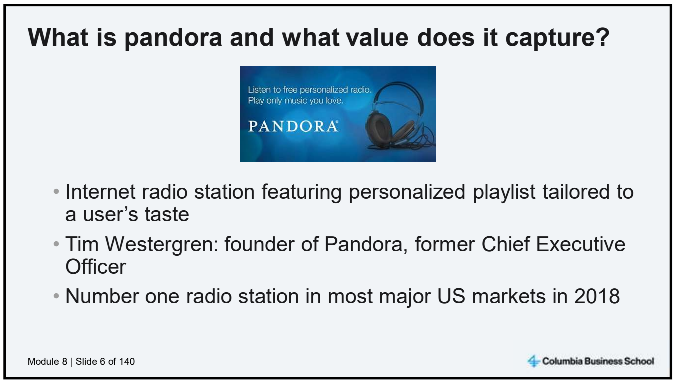


This Module

- Recommendation systems
- How did services such as Pandora and Spotify capture value through analytics?
  - Pandora acquired by SirusXM for $3.5 billion
  - Spotify valued at over $50 billion
- Recommendations through *k*-NN
- Moving from a predictive algorithm to a recommendation system



Pandora and the Music Genome Project

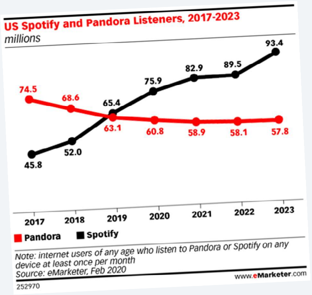


What is pandora and what value does it capture?

Listen to free personalized radio.
Play only music you love.
PANDORA

- Internet radio station featuring personalized playlist tailored to a user's taste
- Tim Westergren: founder of Pandora, former Chief Executive Officer
- Number one radio station in most major US markets in 2018

Pandora vs. Spotify



Note: internet users of any age who listen to Pandora or Spotify on any device at least once per month
Source: eMarketer, Feb 2020

https://www.emarketer.com/content/us-spotify-listeners-surpassed-pandora-listeners-in-2019-sooner-than-expected

The music genome project

- Conceived by Will Glaser and Tim Westergren in 1999; capture the essence of music at a fundamental level
- 5 genomes: pop/rock, hip-hop/electronica, jazz, world music, and classical
- Categories of attributes: melody, harmony, rhythm, form, sound (i.e., instrumentation and voice), lyrics
- Specific attributes (rated by analysts on a 0 to 5 scale)
  - Acid rock qualities, accordion playing, acousti-lectric sonority, acousti-synthetic sonority, …
- Example: For Led Zeppelin's song "Kashmir," the rating starts 4-0-3-3 (high on acid rock attributes, no accordion, medium sonorities)

Example: Norwegian Wood and Stayin' Alive

	Norwegian Wood (Beatles)	Stayin' Alive (Bee Gees)
Beat (fast/slow)	Slow	Fast
Strings	✓	×
Disco	×	✓
Electric guitar	×	✓
Vocals	✓	✓

- Other attributes: harmony, melody, rhythm, specific instruments, etc…
- Pandora introduced a scale for each attribute

What is the main challenge in getting the music genome project to work?

Creating the data!

How many songs can be rated in nine months? What is the cost?

- 450 musical attributes (250 attributes for a pop song)
- 50 song analysts; 20 minutes for one analyst to rate a pop song on 10 attributes
- each analyst works 8 hours/day, 20 days/month at 15 $/hour

Number of songs rated in 9 months

- 250 attributes requires 25 analysts working 20 minutes
- 50 analysts can rate 6 songs per hour; 48 songs per day; 960 songs/month
- Approximately 10,000 songs rated in 9 months

Cost

- 50 song analysts; 15 $/hour; 8 hour/day; 20 days/month; 9 months
- $1 million for 9 months to rate 10,000 songs

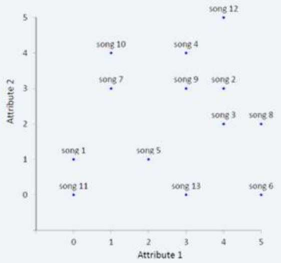
Columbia Business School
AT THE VERY CENTER OF BUSINESS

How does Pandora go from a genome to recommendations?

Distance metrics and the 1-NN algorithm

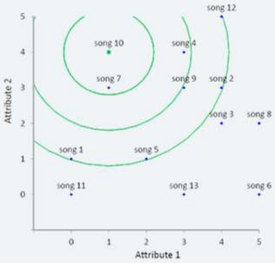
- User selects a favorite song
- We find the “weighted distance” of this song to every other song
- We recommend the song with the minimum weighted distance to the favorite song

1-NN algorithm: Pandora



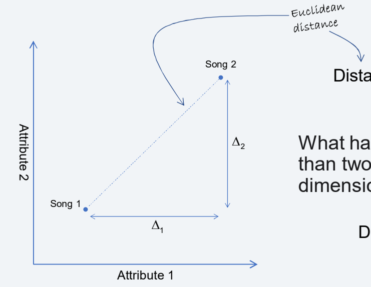
- A user chooses song 10 to listen to first, and rates it “like”
  - Assume the two attributes are equally important
- What song would 1-NN pick to play for the user next?

1-NN algorithm: Pandora



- A user chooses song 10 to listen to first, and rates it “like”
  - Assume the two attributes are equally important
- What song would 1-NN pick to play for the user next?
- **Song 7: it’s the closest song to song 10**

How is “distance” defined?



$$\text{Distance} = \sqrt{(\Delta_1)^2 + (\Delta_2)^2}$$

What happens if there are more than two dimensions? For *N* dimensions, the formula is

$$\text{Distance} = \sqrt{\sum_{i=1}^{N} (\Delta_i)^2}$$

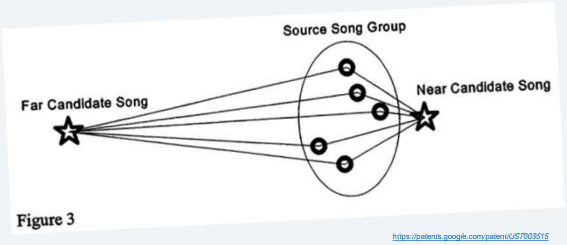
Pandora’s first test

[redacted]

(Optional) Complexities

- If attributes are on very different scales (eg: one rating on a 1-5 scale, and one on a 1-1,000 scale) this distance metric doesn’t work as expected; it helps to standardize columns.
- It is sometimes useful to *weight* different attributes differently (eg: loudness is much more important than tempo).
- In practice, there might be missing values in the data; handling these is a whole topic in its own right.

Pandora’s Patent



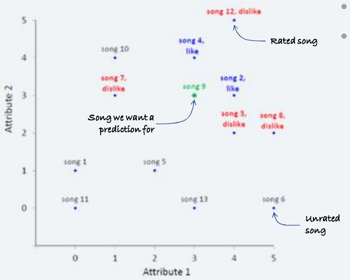
https://patents.google.com/patent/US7003515

Applying nearest-neighbors to predictions

k-NN algorithm

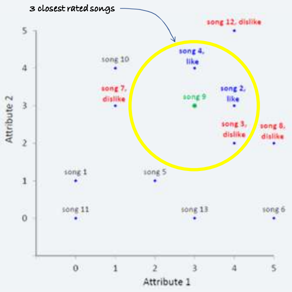
- The concept of a nearest-neighbor can be used for prediction
- This applies very generally
  - Response: take a loan or not
  - Response: like a song or not
  - Response: how much this diamond costs
- The *k*-NN algorithm works as follows
  - Take the individual for which we want to make a prediction
  - Find its *k*-closest neighbors
  - Average the response for these *k*-closest neighbors to get a prediction for our point

3-NN prediction algorithm



- New song: song 9
- Does the 3-NN algorithm predict the user will like or dislike it
  - Assume attributes are equally important
  - Note: the user has only rated 6 songs

3-NN prediction algorithm



- New song: song 9
- Does the 3-NN algorithm predict the user will like or dislike it
  - Assume attributes are equally important
  - Note: the user has only rated 6 songs
- 3 closest rated songs are 4 (like), 2 (like), 3 (dislike)
- P(Like song 9) = 2/3

k-NN vs. Linear Regression

k-NN is just a model like linear regression is – it takes in some variables, and it spits out a prediction

How are *k*-NN and linear regression different? When would you expect one to be better than the other?

Parametric vs. nonparametric models

- **Parametric models** assume a very specific relationship between variables (eg: they are linearly related)
  - If the data fits these assumptions, they're great!
  - If not, they will be sub-optimal
- **Nonparametric models** allow **any** relationship between variables
  - This gives them much more flexibility
  - But it might also make them unnecessarily **complex** and **opaque**

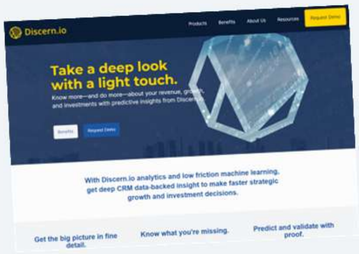
Another example – calculating CLV in B2B businesses

B2B CRM (Customer Relationship Management)

- A large part of the economy comprises businesses that sell their products to *other businesses* only
- Examples include Salesforce, Hubspot, MongoDB, Datadog, Toast, etc…, etc…
- As in every business, sales and customer acquisition are essential parts of growing a business successfully

In what ways to sales in a B2B business differ from sales in a B2C business? How might analytics help with B2B customer acquisition?

Example: Discern.io



Discern.io

Take a deep look with a light touch.

Know more—and do more—about your revenue, growth, and investments with predictive insights from Discern.io

With Discern.io analytics and low friction machine learning, get deep CRM data-backed insight to make faster strategic growth and investment decisions.

Get the big picture in fine detail.

Know what you're missing.

Predict and validate with proof.

Helen Lin
Founder and CEO

Ling Ling
Chief Data Scientist

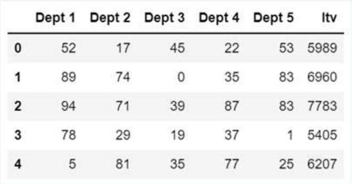
Disclaimer: I am one of discern.io's advisors and own a small amount of stock in the company. The details in this lecture are fictionalized based on Discern's work and do not use any proprietary info

The use case

- During customer acquisition, a B2B business talks to hundreds of businesses in their "sales pipeline"
- Some won't even convert, and of those that do, some will go on to have high CLV (customer lifetime value), some low
- The onboarding process for a new customer involves talking to five departments at the company. Each department gives the customer a score from 1 to 100
- The company then needs to decide which customers to follow up with – the process is time-consuming, and so the company doesn't follow up with everyone

The Data

We have data on 5,000 past customers of the company. In each case, we know the scores assigned by each of the five teams, and the customer lifetime value of the customer



	Dept 1	Dept 2	Dept 3	Dept 4	Dept 5	ltv
0	52	17	45	22	53	5989
1	89	74	0	35	83	6960
2	94	71	39	87	83	7783
3	78	29	19	37	1	5405
4	5	81	35	77	25	6207

Each row is one customer

Customer lifetime value. This will be 0 if the customer doesn't end up closing the deal

What analytic opportunities does this dataset create? What algorithms might we use?

Option 1: linear regression

We could use the past data to fit a linear regression of the form

$$\text{ltv} = \beta_0 + \sum_{i=1}^{5} \beta_i (\text{Department } i \text{ score})$$

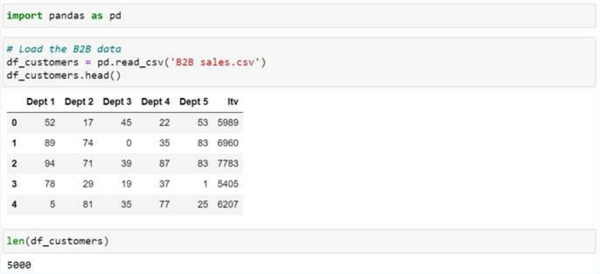
When faced with a new customer for whom we want to predict the lifetime value, we just multiply each of the scores by the relevant β, sum up the results, and get our prediction.

Option 2: *k*-NN

- We could use *k*-NN to make the prediction instead
- When faced with a new customer for whom we want to predict the lifetime value, we would
  - Look at all **past** customers for which we **know** the LTV
  - Find the *k* "closest" customers among those past ones, based on department ratings
  - These are "lookalike" customers that are most similar to our new customer
  - Find the average of these "lookalike" customers – this is the prediction for our new customer

k-NN in Python

Loading the data



```
import pandas as pd
```

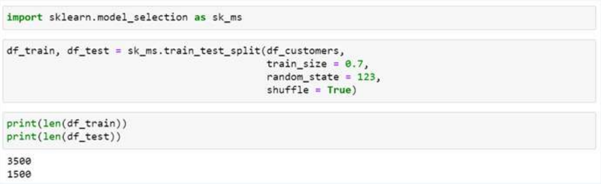
```
# Load the B2B data
df_customers = pd.read_csv('B2B sales.csv')
df_customers.head()
```

	Dept 1	Dept 2	Dept 3	Dept 4	Dept 5	ltv
0	52	17	45	22	53	5989
1	89	74	0	35	83	6960
2	94	71	39	87	83	7783
3	78	29	19	37	1	5405
4	5	81	35	77	25	6207

```
len(df_customers)
```

```
5000
```

Splitting the data into a training and test set



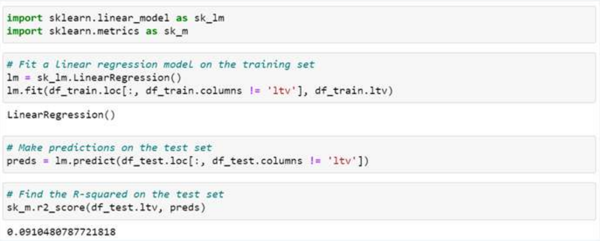
```
import sklearn.model_selection as sk_ms
```

```
df_train, df_test = sk_ms.train_test_split(df_customers,
                                           train_size = 0.7,
                                           random_state = 123,
                                           shuffle = True)
```

```
print(len(df_train))
print(len(df_test))
```

```
3500
1500
```

Let’s try linear regression



```
import sklearn.linear_model as sk_lm
import sklearn.metrics as sk_m
```

```
# Fit a linear regression model on the training set
lm = sk_lm.LinearRegression()
lm.fit(df_train.loc[:, df_train.columns != 'ltv'], df_train.ltv)
```

```
LinearRegression()
```

```
# Make predictions on the test set
preds = lm.predict(df_test.loc[:, df_test.columns != 'ltv'])
```

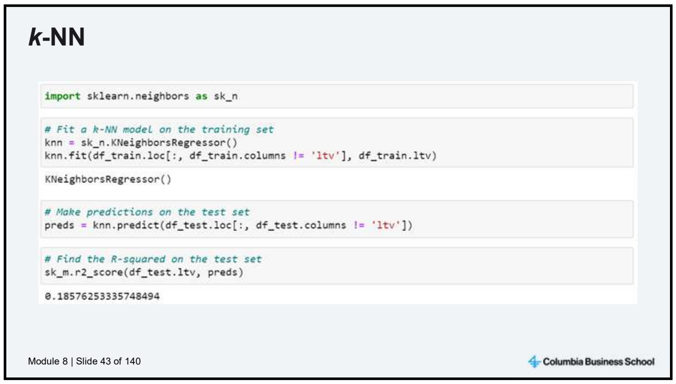
```
# Find the R-squared on the test set
sk_m.r2_score(df_test.ltv, preds)
```

```
0.0910480787721818
```

Linear regression doesn’t seem to be working so well on this dataset! Let’s try *k*-NN

k-NN in Python

- scikit-learn has an in-built model to make predictions using *k*-NN
- These reside in `sklearn.neighbors`
- As with other models, there are separate models for continuous outcomes (regression) and binary outcomes (classification)
  - `KNeighborsRegression()`
  - `KNeighborsClassifier()`
- Let’s see how it’s used



k-NN

```
import sklearn.neighbors as sk_n
```

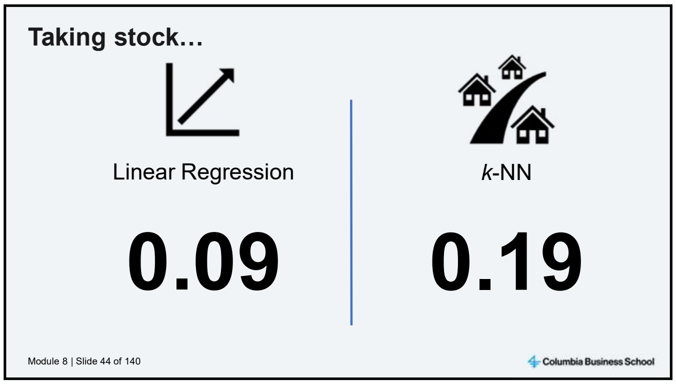
```
# Fit a k-NN model on the training set
knn = sk_n.KNeighborsRegressor()
knn.fit(df_train.loc[:, df_train.columns != 'ltv'], df_train.ltv)
```

```
KNeighborsRegressor()
```

```
# Make predictions on the test set
preds = knn.predict(df_test.loc[:, df_test.columns != 'ltv'])
```

```
# Find the R-squared on the test set
sk_m.r2_score(df_test.ltv, preds)
```

```
0.18576253335748494
```



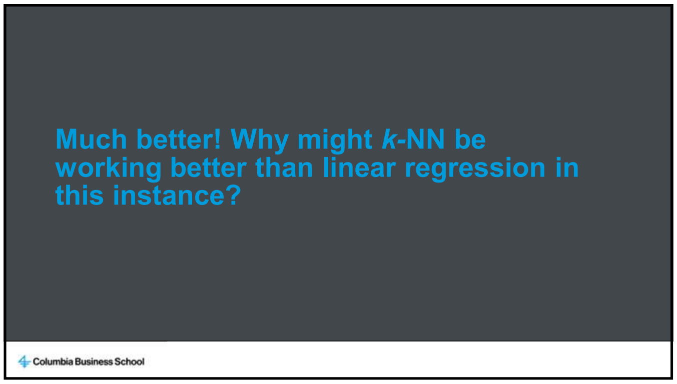
Taking stock…

Linear Regression

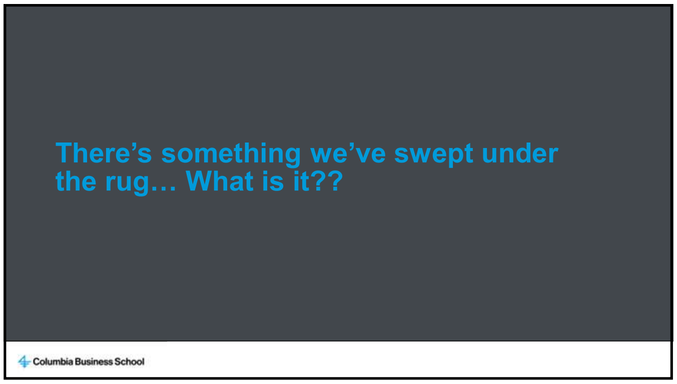
0.09

k-NN

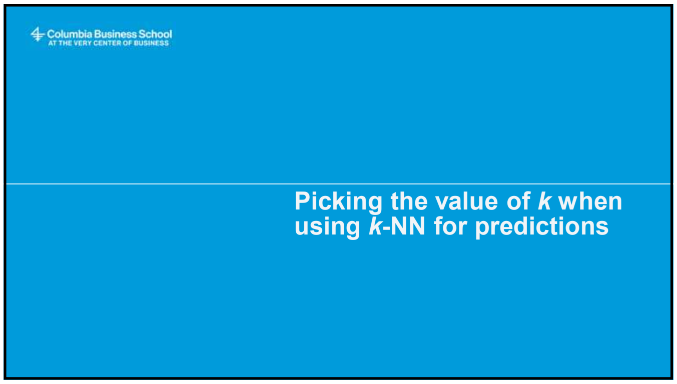
0.19



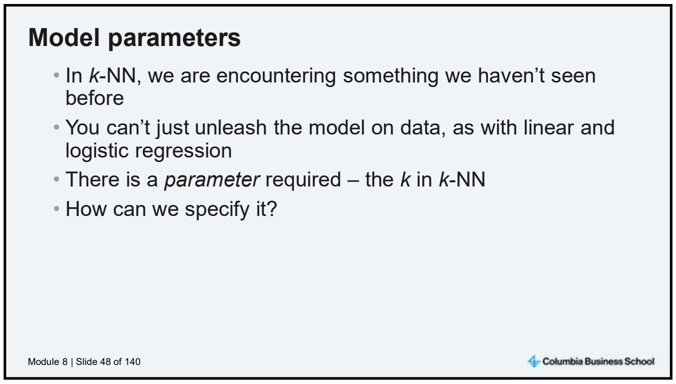
Much better! Why might *k*-NN be working better than linear regression in this instance?



There's something we've swept under the rug… What is it??



Picking the value of *k* when using *k*-NN for predictions



Model parameters

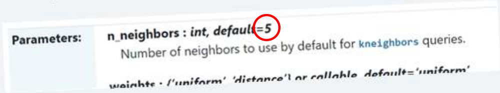
- In *k*-NN, we are encountering something we haven't seen before
- You can't just unleash the model on data, as with linear and logistic regression
- There is a *parameter* required – the *k* in *k*-NN
- How can we specify it?

k in Python

Whenever a model accepts a parameter, scikit-learn will usually allow you to specify it when you first create the model

```
sk_n.KNeighborsRegressor(n_neighbors=12)
```

If you *don't* specify the parameter, scikit-learn will usually use a default, specified in the documentation:

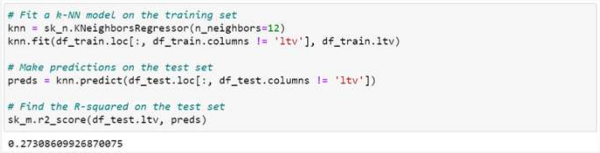


Parameters: **n_neighbors : *int, default=5***
Number of neighbors to use by default for `kneighbors` queries.

weights : *{'uniform', 'distance'} or callable, default='uniform'*

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html

k-NN with 12 neighbors



```
# Fit a k-NN model on the training set
knn = sk_n.KNeighborsRegressor(n_neighbors=12)
knn.fit(df_train.loc[:, df_train.columns != 'ltv'], df_train.ltv)

# Make predictions on the test set
preds = knn.predict(df_test.loc[:, df_test.columns != 'ltv'])

# Find the R-squared on the test set
sk_m.r2_score(df_test.ltv, preds)
```

```
0.27308609926870075
```

Picking the right value of *k* is essential – going from the default *k*=5 to *k*=12 increased our R^2 from 0.19 to 0.27

Never, ever, ever, use `sklearn` defaults

But why are some values of *k* "better" than others?

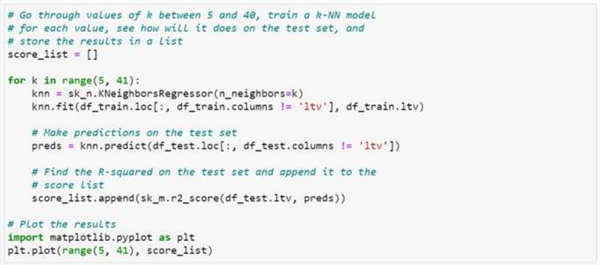
…and how do we pick the best one?

Picking the value of *k*

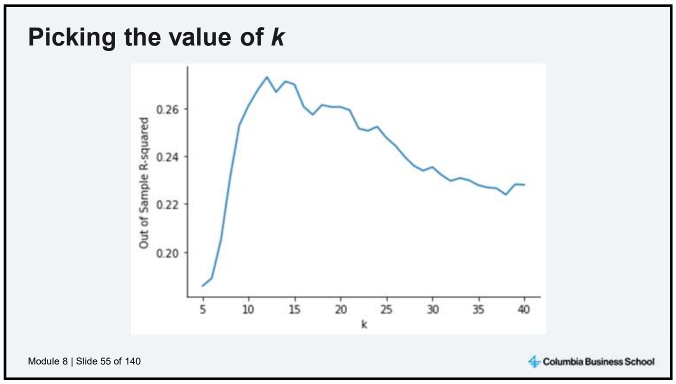
- The value of *k* controls the amount of overfitting in the model
  - If *k* is small (say *k*=1) we simply predict the value of the *closest* neighbor. This is a highly-tailored prediction, but very noisy
  - If *k* is large, *many* points are averaged – this won't be very tailored, but very stable

  We discuss how this relates to overfitting in greater detail in BA2
- To pick the best *k*, we try every value on the test set, and find the one that gives the best performance

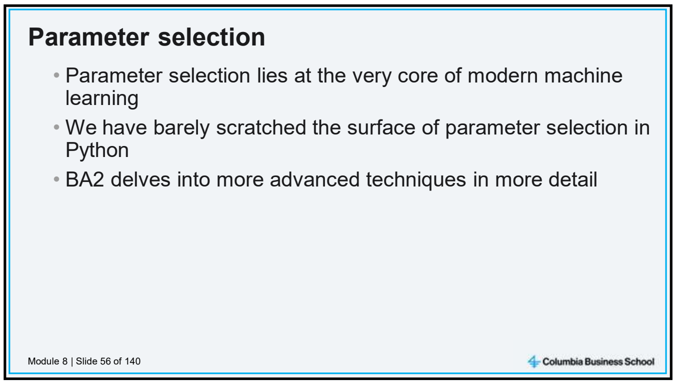
Picking the value of *k*



```
# Go through values of k between 5 and 40, train a k-NN model
# for each value, see how will it does on the test set, and
# store the results in a list
score_list = []

for k in range(5, 41):
    knn = sk_n.KNeighborsRegressor(n_neighbors=k)
    knn.fit(df_train.loc[:, df_train.columns != 'ltv'], df_train.ltv)

    # Make predictions on the test set
    preds = knn.predict(df_test.loc[:, df_test.columns != 'ltv'])

    # Find the R-squared on the test set and append it to the
    # score list
    score_list.append(sk_m.r2_score(df_test.ltv, preds))

# Plot the results
import matplotlib.pyplot as plt
plt.plot(range(5, 41), score_list)
```

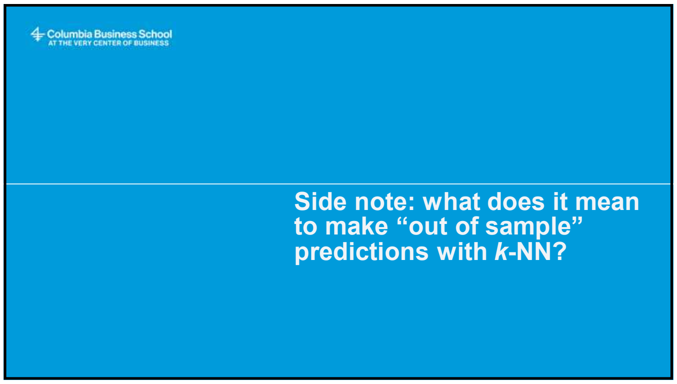


Picking the value of *k*

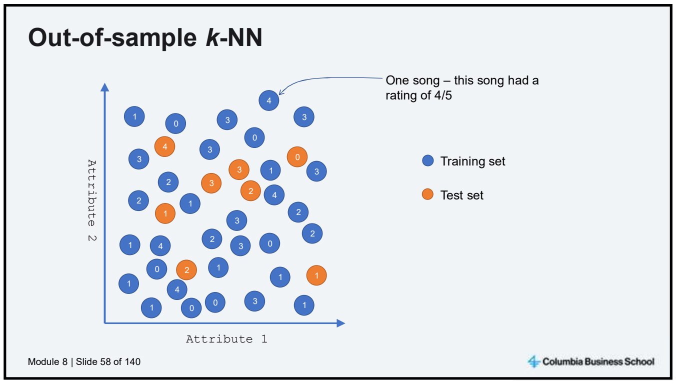


Parameter selection

- Parameter selection lies at the very core of modern machine learning
- We have barely scratched the surface of parameter selection in Python
- BA2 delves into more advanced techniques in more detail

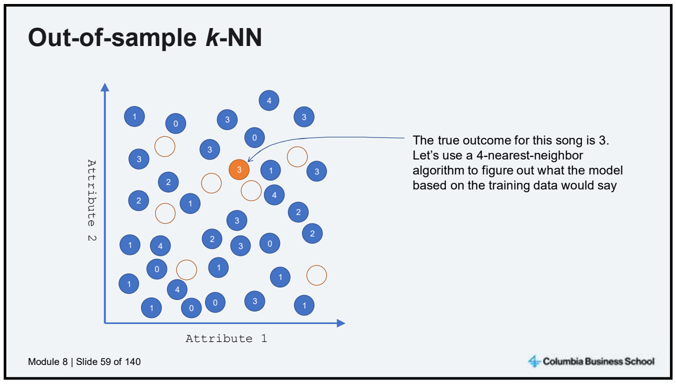


Side note: what does it mean to make "out of sample" predictions with *k*-NN?



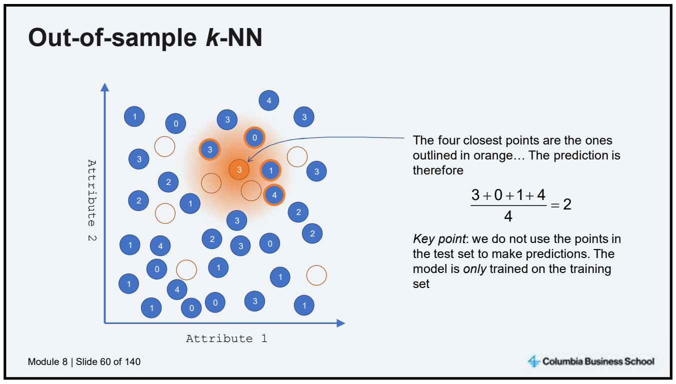
Out-of-sample *k*-NN

One song – this song had a rating of 4/5



Out-of-sample *k*-NN

The true outcome for this song is 3. Let's use a 4-nearest-neighbor algorithm to figure out what the model based on the training data would say

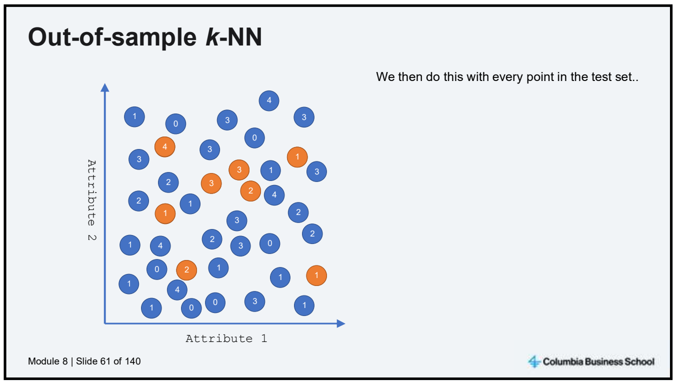


Out-of-sample *k*-NN

The four closest points are the ones outlined in orange… The prediction is therefore

$$\frac{3+0+1+4}{4}=2$$

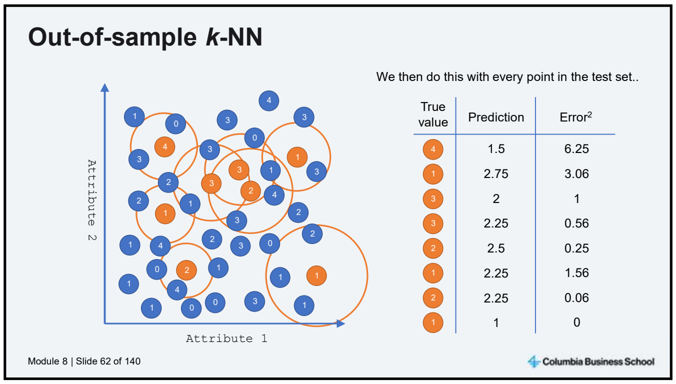
Key point: we do not use the points in the test set to make predictions. The model is *only* trained on the training set



Out-of-sample *k*-NN

We then do this with every point in the test set..

Module 8 | Slide 61 of 140

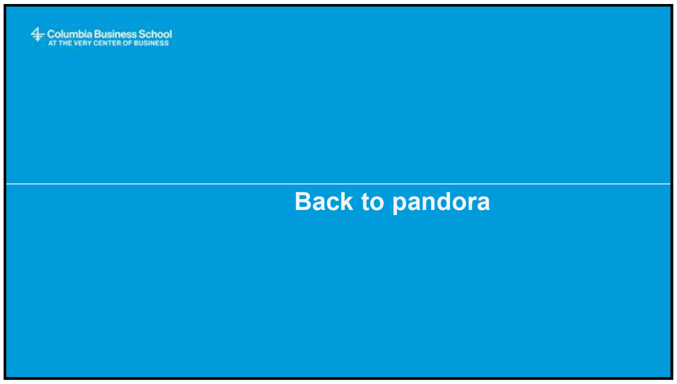


Out-of-sample *k*-NN

We then do this with every point in the test set..

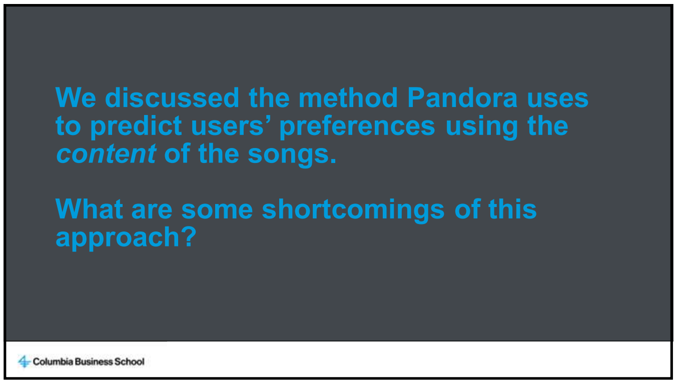
True value	Prediction	Error^2
4	1.5	6.25
1	2.75	3.06
3	2	1
3	2.25	0.56
2	2.5	0.25
1	2.25	1.56
2	2.25	0.06
1	1	0

Module 8 | Slide 62 of 140



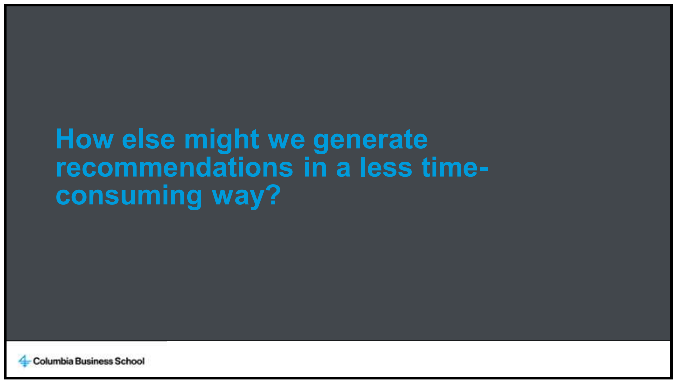
Columbia Business School
AT THE VERY CENTER OF BUSINESS

Back to pandora

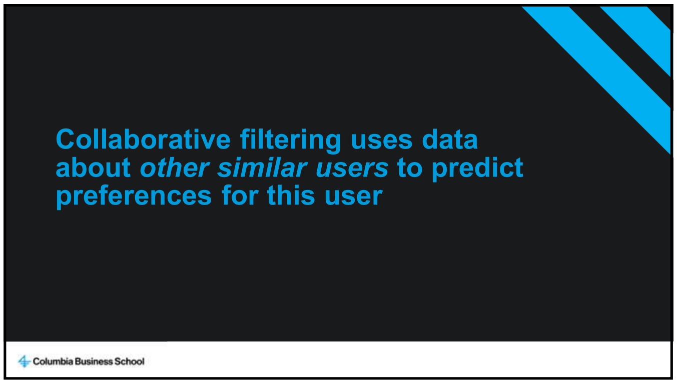


We discussed the method Pandora uses to predict users' preferences using the *content* of the songs.

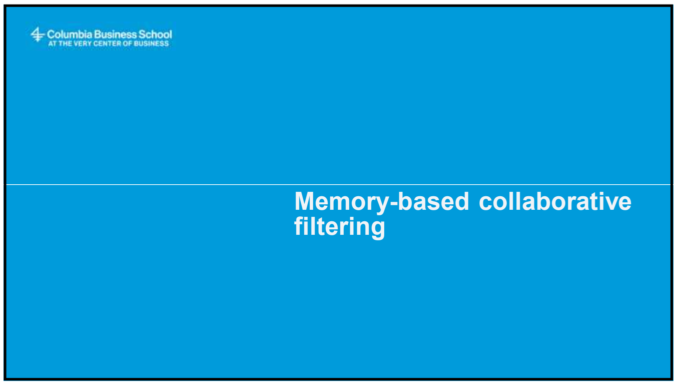
What are some shortcomings of this approach?



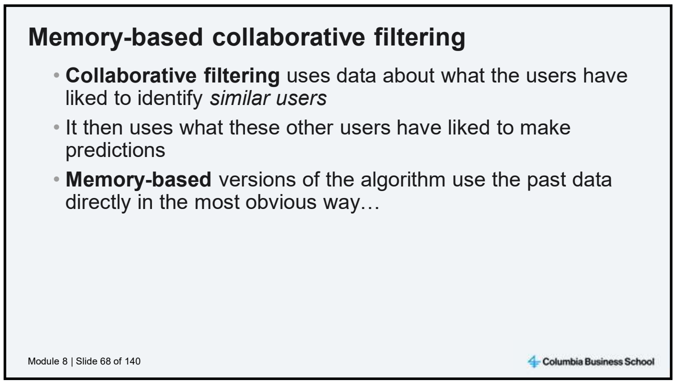
How else might we generate recommendations in a less time-consuming way?



Collaborative filtering uses data about *other similar users* to predict preferences for this user

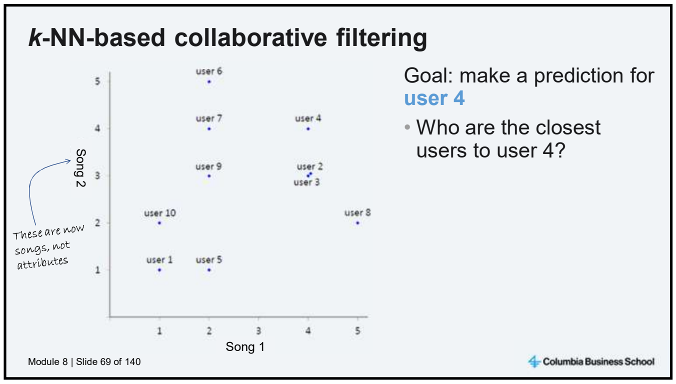


Memory-based collaborative filtering



Memory-based collaborative filtering

- **Collaborative filtering** uses data about what the users have liked to identify *similar users*
- It then uses what these other users have liked to make predictions
- **Memory-based** versions of the algorithm use the past data directly in the most obvious way…

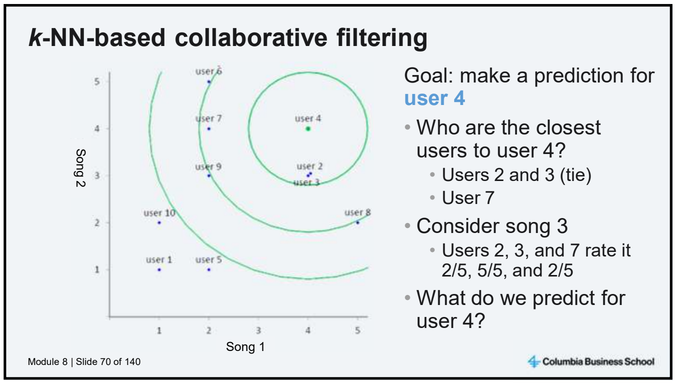


k-NN-based collaborative filtering

These are now songs, not attributes

Goal: make a prediction for **user 4**

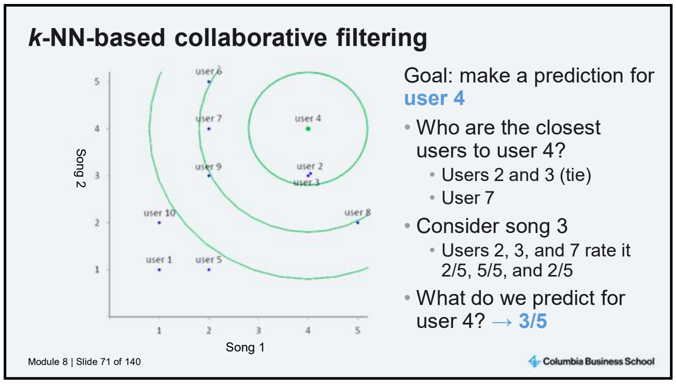
- Who are the closest users to user 4?



k-NN-based collaborative filtering

Goal: make a prediction for **user 4**

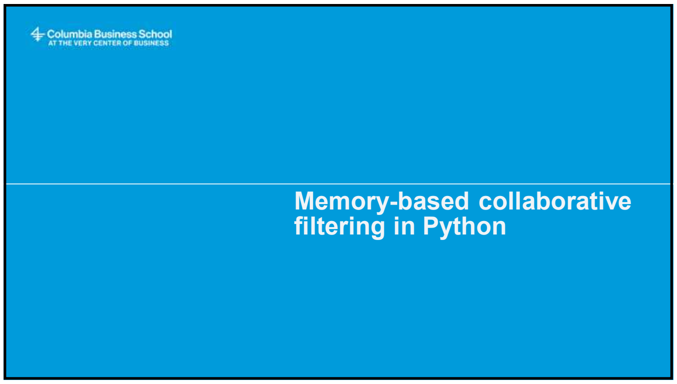
- Who are the closest users to user 4?
  - Users 2 and 3 (tie)
  - User 7
- Consider song 3
  - Users 2, 3, and 7 rate it 2/5, 5/5, and 2/5
- What do we predict for user 4?



k-NN-based collaborative filtering

Goal: make a prediction for **user 4**

- Who are the closest users to user 4?
  - Users 2 and 3 (tie)
  - User 7
- Consider song 3
  - Users 2, 3, and 7 rate it 2/5, 5/5, and 2/5
- What do we predict for user 4? → **3/5**



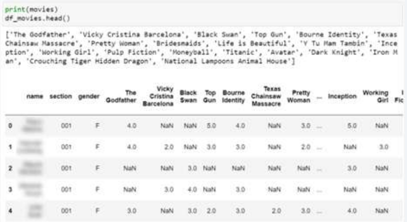
Memory-based collaborative filtering in Python

Important note

The most efficient way of carrying out these operations is using high-performance Python libraries like `numpy`. We will use **much slower** – but easier to understand – techniques to cover these concepts without too many prerequisites.

Loading the canvas survey results

We first load the Canvas survey results; see the optional cell in the notebook for the code. The data looks like this:

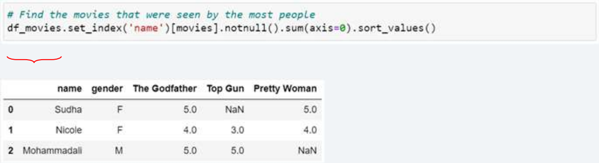


```
print(movies)
df_movies.head()
```

```
['The Godfather', 'Vicky Cristina Barcelona', 'Black Swan', 'Top Gun', 'Bourne Identity', 'Texas
Chainsaw Massacre', 'Pretty Woman', 'Bridesmaids', 'Life is Beautiful', 'Y Tu Mam Tambin', 'Ince
ption', 'Working Girl', 'Pulp Fiction', 'Moneyball', 'Titanic', 'Avatar', 'Dark Knight', 'Iron M
an', 'Crouching Tiger Hidden Dragon', 'National Lampoons Animal House']
```

	name	section	gender	The Godfather	Vicky Cristina Barcelona	Black Swan	Top Gun	Bourne Identity	Texas Chainsaw Massacre	Pretty Woman	...	Inception	Working Girl	Fic
0	[redacted]	001	F	4.0	NaN	NaN	5.0	4.0	NaN	3.0	...	5.0	NaN	
1	[redacted]	001	F	4.0	2.0	NaN	3.0	3.0	NaN	2.0	...	NaN	3.0	
2	[redacted]	001	F	NaN	NaN	3.0	NaN	NaN	NaN	NaN	...	3.0	NaN	
3	[redacted]	001	F	NaN	3.0	4.0	NaN	3.0	NaN	NaN	...	NaN	NaN	
4	[redacted]	001	F	3.0	NaN	3.0	2.0	3.0	2.0	3.0	...	4.0	NaN	

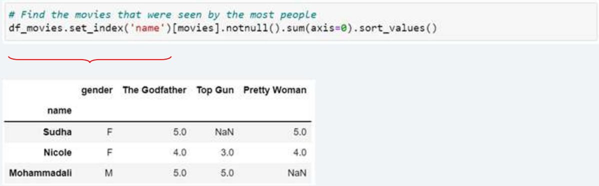
Summary statistics (most seen movies)



```
# Find the movies that were seen by the most people
df_movies.set_index('name')[movies].notnull().sum(axis=0).sort_values()
```

	name	gender	The Godfather	Top Gun	Pretty Woman
0	Sudha	F	5.0	NaN	5.0
1	Nicole	F	4.0	3.0	4.0
2	Mohammadali	M	5.0	5.0	NaN

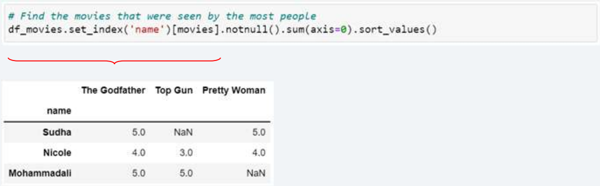
Summary statistics (most seen movies)



```
# Find the movies that were seen by the most people
df_movies.set_index('name')[movies].notnull().sum(axis=0).sort_values()
```

	gender	The Godfather	Top Gun	Pretty Woman
name				
Sudha	F	5.0	NaN	5.0
Nicole	F	4.0	3.0	4.0
Mohammadali	M	5.0	5.0	NaN

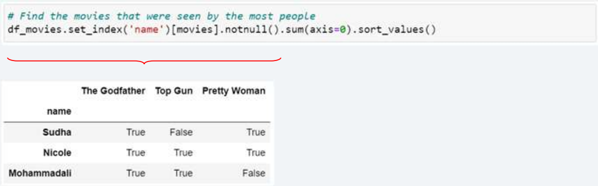
Summary statistics (most seen movies)



```
# Find the movies that were seen by the most people
df_movies.set_index('name')[movies].notnull().sum(axis=0).sort_values()
```

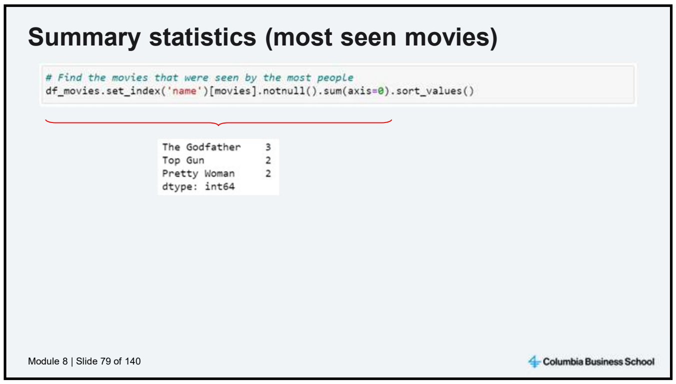
	The Godfather	Top Gun	Pretty Woman
name			
Sudha	5.0	NaN	5.0
Nicole	4.0	3.0	4.0
Mohammadali	5.0	5.0	NaN

Summary statistics (most seen movies)



```
# Find the movies that were seen by the most people
df_movies.set_index('name')[movies].notnull().sum(axis=0).sort_values()
```

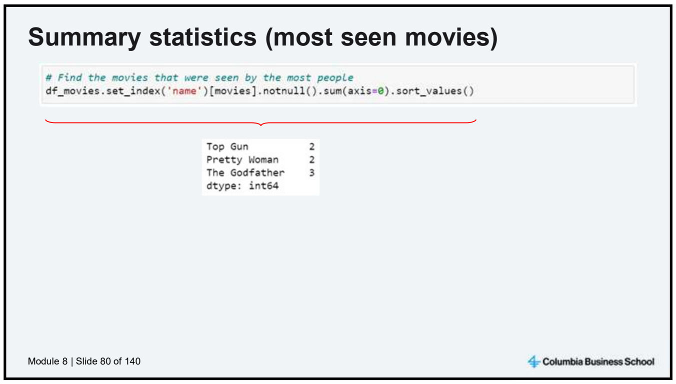
	The Godfather	Top Gun	Pretty Woman
name			
Sudha	True	False	True
Nicole	True	True	True
Mohammadali	True	True	False



Summary statistics (most seen movies)

```
# Find the movies that were seen by the most people
df_movies.set_index('name')[movies].notnull().sum(axis=0).sort_values()
```

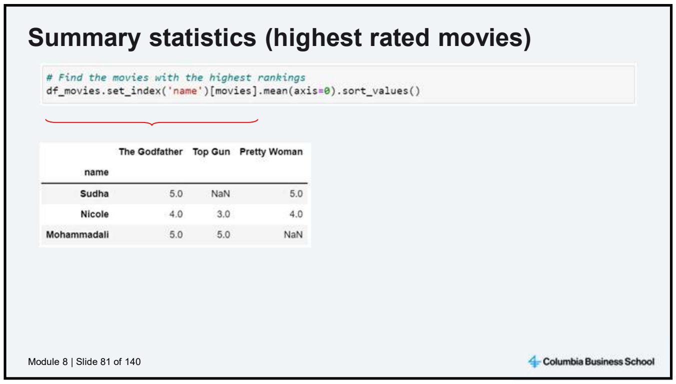
```
The Godfather    3
Top Gun          2
Pretty Woman     2
dtype: int64
```



Summary statistics (most seen movies)

```
# Find the movies that were seen by the most people
df_movies.set_index('name')[movies].notnull().sum(axis=0).sort_values()
```

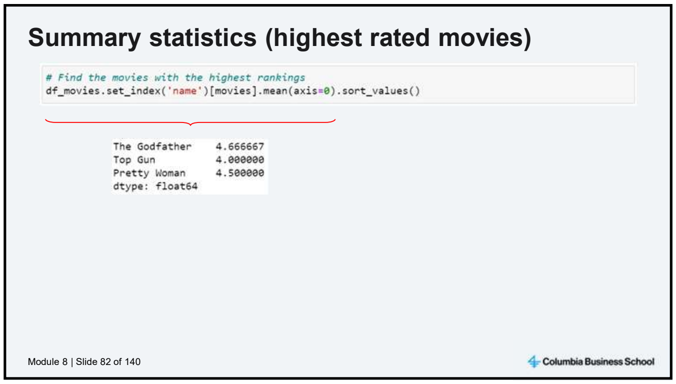
```
Top Gun          2
Pretty Woman     2
The Godfather    3
dtype: int64
```



Summary statistics (highest rated movies)

```
# Find the movies with the highest rankings
df_movies.set_index('name')[movies].mean(axis=0).sort_values()
```

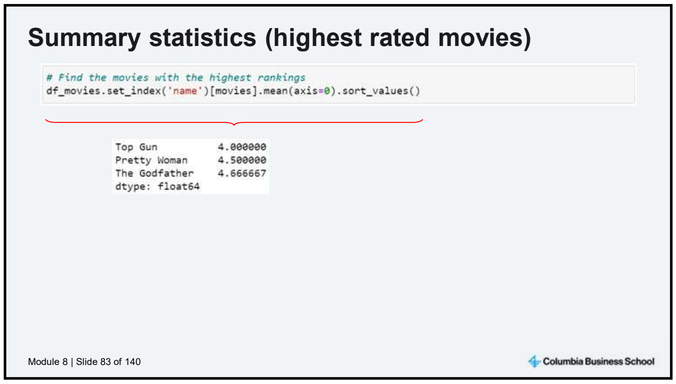
name	The Godfather	Top Gun	Pretty Woman
Sudha	5.0	NaN	5.0
Nicole	4.0	3.0	4.0
Mohammadali	5.0	5.0	NaN



Summary statistics (highest rated movies)

```
# Find the movies with the highest rankings
df_movies.set_index('name')[movies].mean(axis=0).sort_values()
```

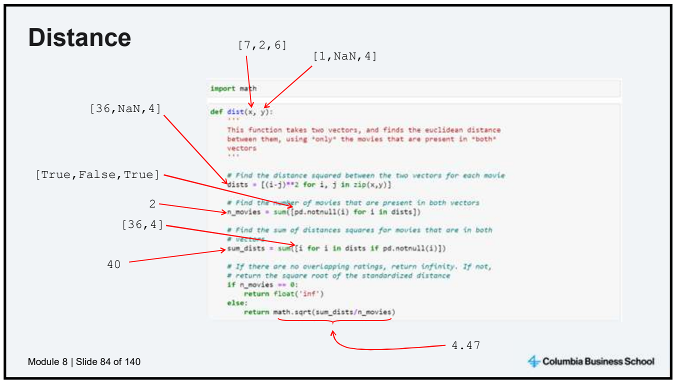
```
The Godfather    4.666667
Top Gun          4.000000
Pretty Woman     4.500000
dtype: float64
```



Summary statistics (highest rated movies)

```
# Find the movies with the highest rankings
df_movies.set_index('name')[movies].mean(axis=0).sort_values()
```

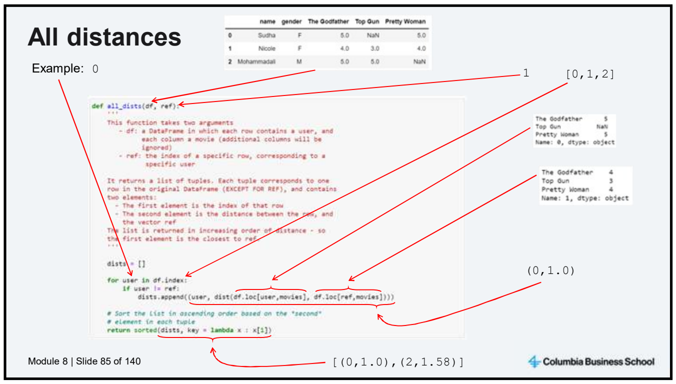
```
Top Gun          4.000000
Pretty Woman     4.500000
The Godfather    4.666667
dtype: float64
```



Distance

```
import math
```

```
def dist(x, y):
    '''
    This function takes two vectors, and finds the euclidean distance
    between them, using *only* the movies that are present in *both*
    vectors
    '''

    # Find the distance squared between the two vectors for each movie
    dists = [(i-j)**2 for i, j in zip(x,y)]

    # Find the number of movies that are present in both vectors
    n_movies = sum([pd.notnull(i) for i in dists])

    # Find the sum of distances squares for movies that are in both
    # vectors
    sum_dists = sum([i for i in dists if pd.notnull(i)])

    # If there are no overlapping ratings, return infinity. If not,
    # return the square root of the standardized distance
    if n_movies == 0:
        return float('inf')
    else:
        return math.sqrt(sum_dists/n_movies)
```



All distances

	name	gender	The Godfather	Top Gun	Pretty Woman
0	Sudha	F	5.0	NaN	5.0
1	Nicole	F	4.0	3.0	4.0
2	Mohammadali	M	5.0	5.0	NaN

Example: 0

1

[0,1,2]

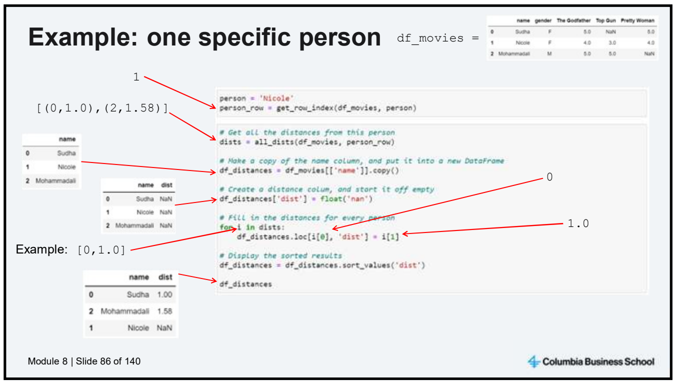
```
def all_dists(df, ref):
    '''
    This function takes two arguments
       - df: a DataFrame in which each row contains a user, and
             each column a movie (additional columns will be
             ignored)
       - ref: the index of a specific row, corresponding to a
              specific user

    It returns a list of tuples. Each tuple corresponds to one
    row in the original DataFrame (EXCEPT FOR REF), and contains
    two elements:
      - The first element is the index of that row
      - The second element is the distance between the row, and
        the vector ref
    The list is returned in increasing order of distance - so
    the first element is the closest to ref.
    '''

    dists = []

    for user in df.index:
        if user != ref:
            dists.append((user, dist(df.loc[user,movies], df.loc[ref,movies])))

    # Sort the list in ascending order based on the "second"
    # element in each tuple
    return sorted(dists, key = lambda x : x[1])
```

```
The Godfather      5
Top Gun          NaN
Pretty Woman       5
Name: 0, dtype: object
```

```
The Godfather    4
Top Gun          3
Pretty Woman     4
Name: 1, dtype: object
```

(0,1.0)

[(0,1.0),(2,1.58)]



Example: one specific person

df_movies =

	name	gender	The Godfather	Top Gun	Pretty Woman
0	Sudha	F	5.0	NaN	5.0
1	Nicole	F	4.0	3.0	4.0
2	Mohammadali	M	5.0	5.0	NaN

1

[(0,1.0),(2,1.58)]

```
person = 'Nicole'
person_row = get_row_index(df_movies, person)
```

```
# Get all the distances from this person
dists = all_dists(df_movies, person_row)

# Make a copy of the name column, and put it into a new DataFrame
df_distances = df_movies[['name']].copy()

# Create a distance colum, and start it off empty
df_distances['dist'] = float('nan')

# Fill in the distances for every person
for i in dists:
    df_distances.loc[i[0], 'dist'] = i[1]

# Display the sorted results
df_distances = df_distances.sort_values('dist')

df_distances
```

	name
0	Sudha
1	Nicole
2	Mohammadali

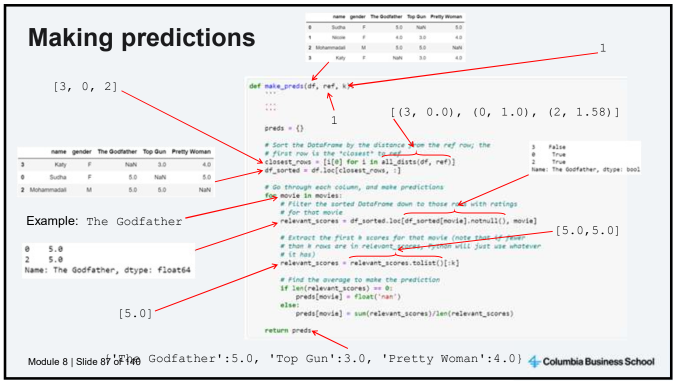
	name	dist
0	Sudha	NaN
1	Nicole	NaN
2	Mohammadali	NaN

0

1.0

Example: [0,1.0]

	name	dist
0	Sudha	1.00
2	Mohammadali	1.58
1	Nicole	NaN



Making predictions

	name	gender	The Godfather	Top Gun	Pretty Woman
0	Sudha	F	5.0	NaN	5.0
1	Nicole	F	4.0	3.0	4.0
2	Mohammadali	M	5.0	5.0	NaN
3	Katy	F	NaN	3.0	4.0

1

[3, 0, 2]

```
def make_preds(df, ref, k):
    '''
    ...
    '''

    preds = {}

    # Sort the DataFrame by the distance from the ref row; the
    # first row is the "closest" to ref
    closest_rows = [i[0] for i in all_dists(df, ref)]
    df_sorted = df.loc[closest_rows, :]

    # Go through each column, and make predictions
    for movie in movies:
        # Filter the sorted DataFrame down to those rows with ratings
        # for that movie
        relevant_scores = df_sorted.loc[df_sorted[movie].notnull(), movie]

        # Extract the first k scores for that movie (note that if fewer
        # than k rows are in relevant_scores, Python will just use whatever
        # it has)
        relevant_scores = relevant_scores.tolist()[:k]

        # Find the average to make the prediction
        if len(relevant_scores) == 0:
            preds[movie] = float('nan')
        else:
            preds[movie] = sum(relevant_scores)/len(relevant_scores)

    return preds
```

1

[(3, 0.0), (0, 1.0), (2, 1.58)]

	name	gender	The Godfather	Top Gun	Pretty Woman
3	Katy	F	NaN	3.0	4.0
0	Sudha	F	5.0	NaN	5.0
2	Mohammadali	M	5.0	5.0	NaN

```
3    False
0     True
2     True
Name: The Godfather, dtype: bool
```

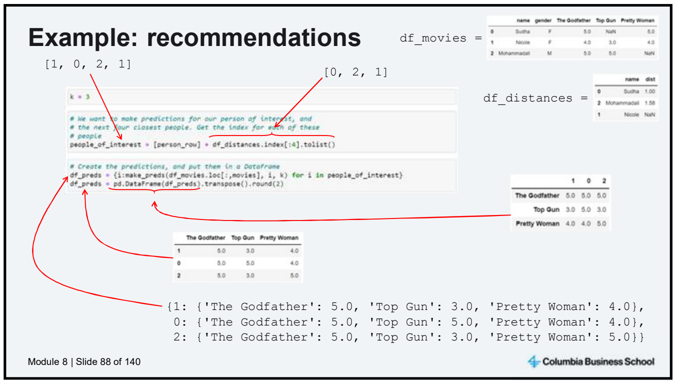
Example: The Godfather

```
0    5.0
2    5.0
Name: The Godfather, dtype: float64
```

[5.0,5.0]

[5.0]

{'The Godfather':5.0, 'Top Gun':3.0, 'Pretty Woman':4.0}



Example: recommendations

df_movies =

	name	gender	The Godfather	Top Gun	Pretty Woman
0	Sudha	F	5.0	NaN	5.0
1	Nicole	F	4.0	3.0	4.0
2	Mohammadali	M	5.0	5.0	NaN

[1, 0, 2, 1]

[0, 2, 1]

df_distances =

	name	dist
0	Sudha	1.00
2	Mohammadali	1.58
1	Nicole	NaN

```
k = 3
```

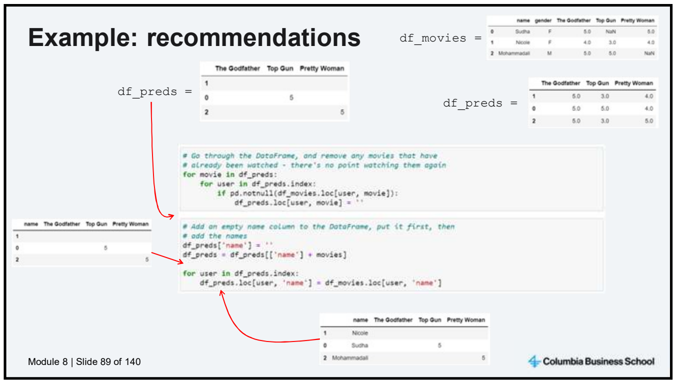
```
# We want to make predictions for our person of interest, and
# the next four closest people. Get the index for each of these
# people
people_of_interest = [person_row] + df_distances.index[:4].tolist()
```

```
# Create the predictions, and put them in a DataFrame
df_preds = {i:make_preds(df_movies.loc[:,movies], i, k) for i in people_of_interest}
df_preds = pd.DataFrame(df_preds).transpose().round(2)
```

	1	0	2
The Godfather	5.0	5.0	5.0
Top Gun	3.0	5.0	3.0
Pretty Woman	4.0	4.0	5.0

	The Godfather	Top Gun	Pretty Woman
1	5.0	3.0	4.0
0	5.0	5.0	4.0
2	5.0	3.0	5.0

```
{1: {'The Godfather': 5.0, 'Top Gun': 3.0, 'Pretty Woman': 4.0},
 0: {'The Godfather': 5.0, 'Top Gun': 5.0, 'Pretty Woman': 4.0},
 2: {'The Godfather': 5.0, 'Top Gun': 3.0, 'Pretty Woman': 5.0}}
```



Example: recommendations

df_movies =

	name	gender	The Godfather	Top Gun	Pretty Woman
0	Sudha	F	5.0	NaN	5.0
1	Nicole	F	4.0	3.0	4.0
2	Mohammadali	M	5.0	5.0	NaN

df_preds =

	The Godfather	Top Gun	Pretty Woman
1			
0		5	
2			5

df_preds =

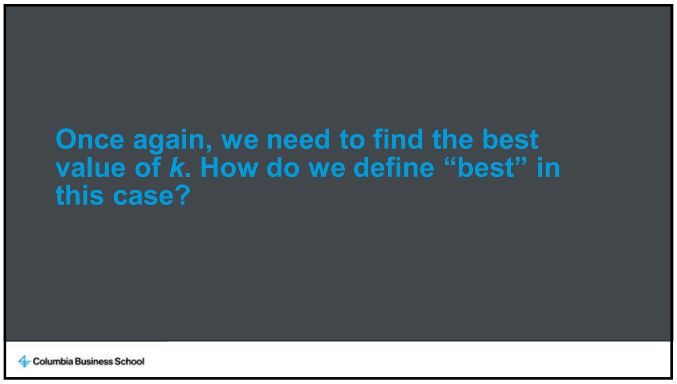
	The Godfather	Top Gun	Pretty Woman
1	5.0	3.0	4.0
0	5.0	5.0	4.0
2	5.0	3.0	5.0

```
# Go through the DataFrame, and remove any movies that have
# already been watched - there's no point watching them again
for movie in df_preds:
    for user in df_preds.index:
        if pd.notnull(df_movies.loc[user, movie]):
            df_preds.loc[user, movie] = ''
```

```
# Add an empty name column to the DataFrame, put it first, then
# add the names
df_preds['name'] = ''
df_preds = df_preds[['name'] + movies]

for user in df_preds.index:
    df_preds.loc[user, 'name'] = df_movies.loc[user, 'name']
```

	name	The Godfather	Top Gun	Pretty Woman
1				
0			5	
2				5

	name	The Godfather	Top Gun	Pretty Woman
1	Nicole			
0	Sudha		5	
2	Mohammadali			5

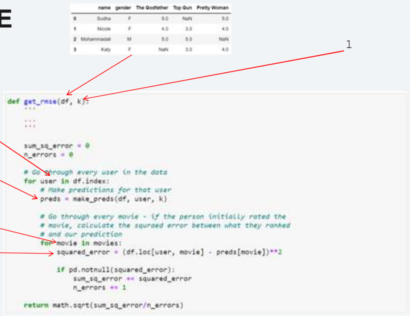


Once again, we need to find the best value of *k*. How do we define "best" in this case?

Finding the RMSE

- The concept of RMSE is a little more tricky here
- There isn't a set of "*y*" values that we are trying to predict with a set of "*x*" values. Everything is intertwined
- Instead, we will make predictions for every user (using every other user) and compare these predictions to the truth
- In reality, we should do this with a training/test set (keeping some of the movies as "test") but we'll leave that as an exercise…

Finding the RMSE



	name	gender	The Godfather	Top Gun	Pretty Woman
0	Sudha	F	5.0	NaN	5.0
1	Nicole	F	4.0	3.0	4.0
2	Mohammadali	M	5.0	5.0	NaN
3	Katy	F	NaN	3.0	4.0

```
def get_rmse(df, k):
    '''

    '''
    '''

    sum_sq_error = 0
    n_errors = 0

    # Go through every user in the data
    for user in df.index:
        # Make predictions for that user
        preds = make_preds(df, user, k)

        # Go through every movie - if the person initially rated the
        # movie, calculate the squraed error between what they ranked
        # and our prediction
        for movie in movies:
            squared_error = (df.loc[user, movie] - preds[movie])**2

            if pd.notnull(squared_error):
                sum_sq_error += squared_error
                n_errors += 1

    return math.sqrt(sum_sq_error/n_errors)
```

1

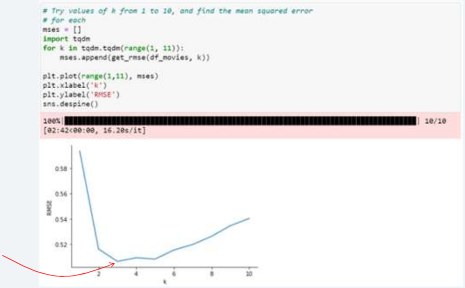
Example: `2`

`{'The Godfather': 5.0, 'Top Gun': 3.0, 'Pretty Woman': 5.0}`

Example: `Top Gun`

`4`

The best RMSE



```
# Try values of k from 1 to 10, and find the mean squared error
# for each
mses = []
import tqdm
for k in tqdm.tqdm(range(1, 11)):
    mses.append(get_rmse(df_movies, k))

plt.plot(range(1,11), mses)
plt.xlabel('k')
plt.ylabel('RMSE')
sns.despine()
```

```
100%|████████████████████████████████████████████████████████| 10/10
[02:42<00:00, 16.20s/it]
```

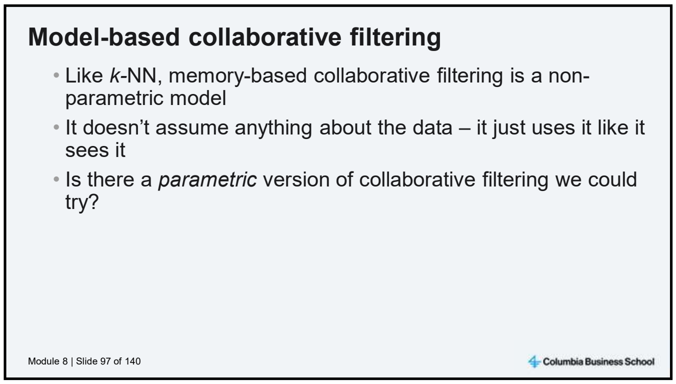
0.5065

Why is this not quite correct? What are we missing?

In theory, we should split the data into a training and test set, and only use the test set in determining *k*…

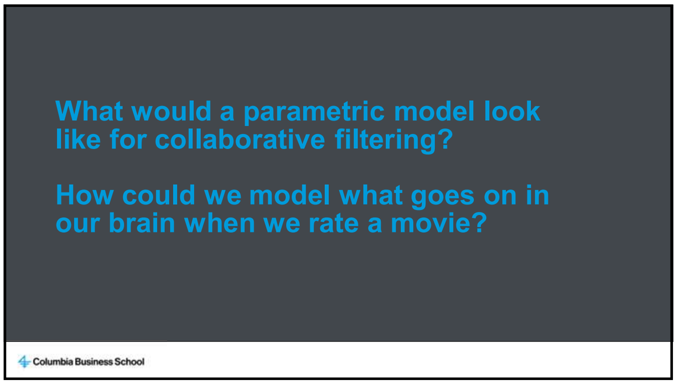
I leave this as an exercise…

Model-based collaborative filtering



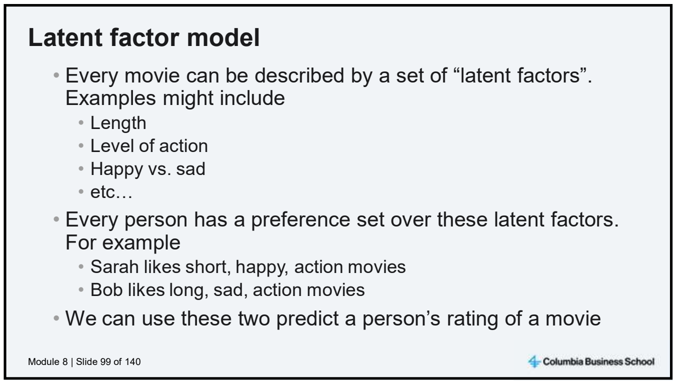
Model-based collaborative filtering

- Like *k*-NN, memory-based collaborative filtering is a non-parametric model
- It doesn't assume anything about the data – it just uses it like it sees it
- Is there a *parametric* version of collaborative filtering we could try?



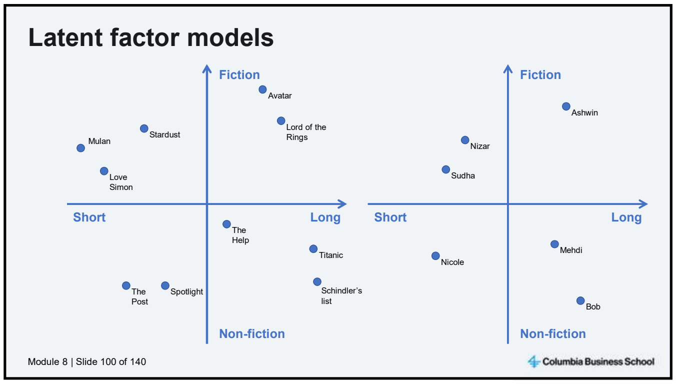
What would a parametric model look like for collaborative filtering?

How could we model what goes on in our brain when we rate a movie?

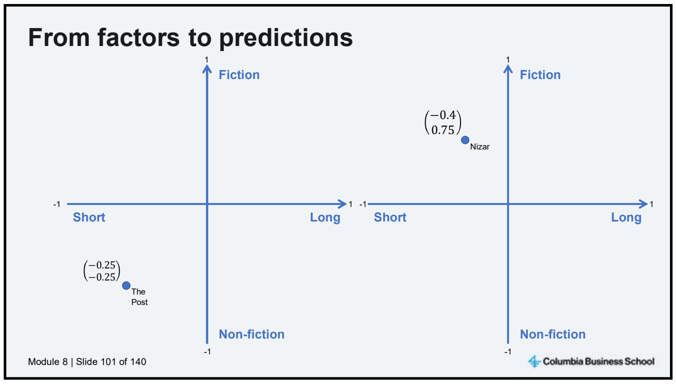


Latent factor model

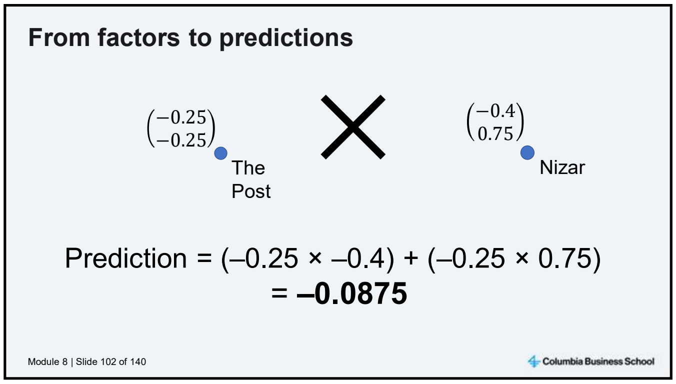
- Every movie can be described by a set of "latent factors". Examples might include
  - Length
  - Level of action
  - Happy vs. sad
  - etc…
- Every person has a preference set over these latent factors. For example
  - Sarah likes short, happy, action movies
  - Bob likes long, sad, action movies
- We can use these two predict a person's rating of a movie



Latent factor models



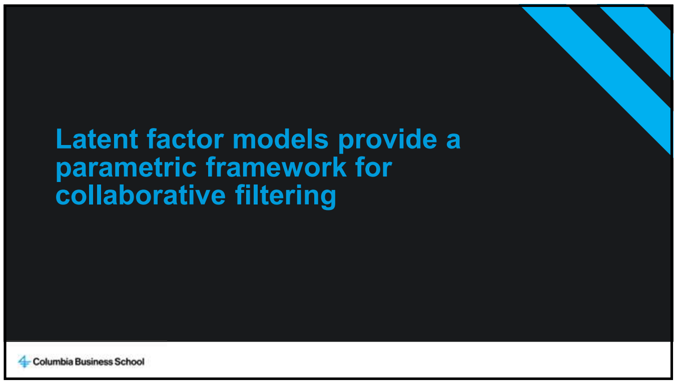
From factors to predictions



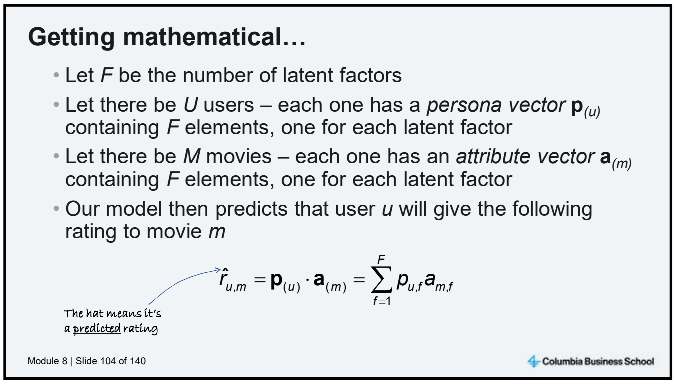
From factors to predictions

$\begin{pmatrix}-0.25\\-0.25\end{pmatrix}$
The Post
×
$\begin{pmatrix}-0.4\\0.75\end{pmatrix}$
Nizar

Prediction = (–0.25 × –0.4) + (–0.25 × 0.75)
= **–0.0875**



Latent factor models provide a parametric framework for collaborative filtering

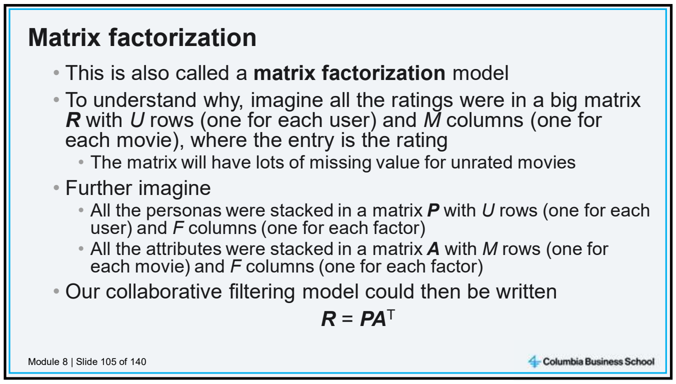


Getting mathematical…

- Let *F* be the number of latent factors
- Let there be *U* users – each one has a *persona vector* $\mathbf{p}_{(u)}$ containing *F* elements, one for each latent factor
- Let there be *M* movies – each one has an *attribute vector* $\mathbf{a}_{(m)}$ containing *F* elements, one for each latent factor
- Our model then predicts that user *u* will give the following rating to movie *m*

$$\hat{r}_{u,m} = \mathbf{p}_{(u)} \cdot \mathbf{a}_{(m)} = \sum_{f=1}^{F} p_{u,f} a_{m,f}$$

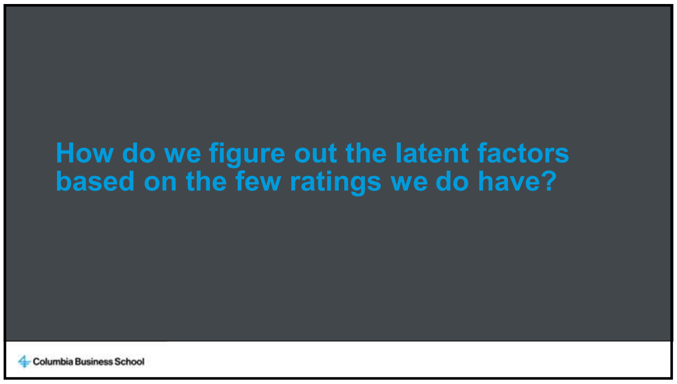
The hat means it's a predicted rating



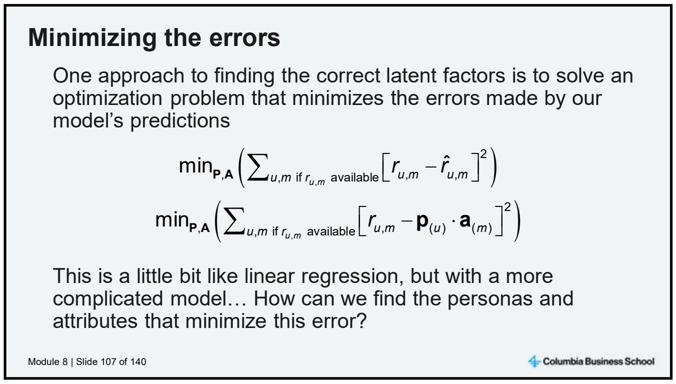
Matrix factorization

- This is also called a **matrix factorization** model
- To understand why, imagine all the ratings were in a big matrix ***R*** with *U* rows (one for each user) and *M* columns (one for each movie), where the entry is the rating
  - The matrix will have lots of missing value for unrated movies
- Further imagine
  - All the personas were stacked in a matrix ***P*** with *U* rows (one for each user) and *F* columns (one for each factor)
  - All the attributes were stacked in a matrix ***A*** with *M* rows (one for each movie) and *F* columns (one for each factor)
- Our collaborative filtering model could then be written

$$\boldsymbol{R} = \boldsymbol{P}\boldsymbol{A}^{\mathsf{T}}$$



How do we figure out the latent factors based on the few ratings we do have?



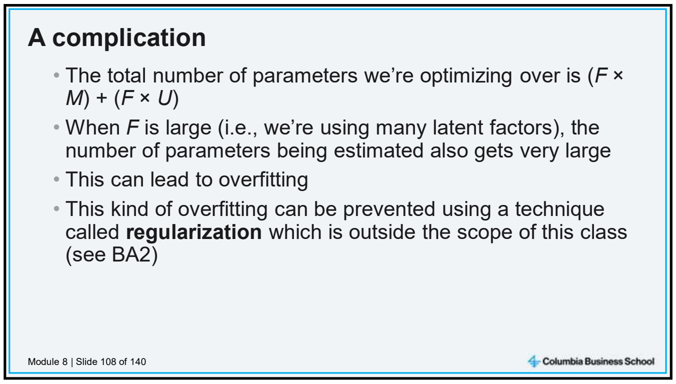
Minimizing the errors

One approach to finding the correct latent factors is to solve an optimization problem that minimizes the errors made by our model's predictions

$$\min_{\mathbf{P},\mathbf{A}} \left(\sum_{u,m \text{ if } r_{u,m} \text{ available}} \left[r_{u,m} - \hat{r}_{u,m} \right]^2 \right)$$

$$\min_{\mathbf{P},\mathbf{A}} \left(\sum_{u,m \text{ if } r_{u,m} \text{ available}} \left[r_{u,m} - \mathbf{p}_{(u)} \cdot \mathbf{a}_{(m)} \right]^2 \right)$$

This is a little bit like linear regression, but with a more complicated model… How can we find the personas and attributes that minimize this error?



A complication

- The total number of parameters we're optimizing over is (*F* × *M*) + (*F* × *U*)
- When *F* is large (i.e., we're using many latent factors), the number of parameters being estimated also gets very large
- This can lead to overfitting
- This kind of overfitting can be prevented using a technique called **regularization** which is outside the scope of this class (see BA2)

The latent number of factors needs to be specified manually in this model… More advanced models exist which can help us "detect" the "best" number of factors.

Columbia Business School

Columbia Business School
AT THE VERY CENTER OF BUSINESS

Stochastic gradient descent for matrix factorization (optional)

Gradient descent

- We can use gradient descent – which we saw when we discussed logistic regression – to solve this problem as well
- Note that gradient descent isn't guaranteed to work when the optimization problem is **nonconvex** (a concept you might cover in more advanced classes).
  - This problem is non-convex, but as we'll see, gradient descent will work fine

The gradient

What is the gradient with respect to the variables ***P*** and ***A*** in this case?

$$\min_{\mathbf{P},\mathbf{A}}\left(\sum\nolimits_{u,m \text{ if } r_{u,m} \text{ available}}\left[r_{u,m}-\mathbf{p}_{(u)}\cdot\mathbf{a}_{(m)}\right]^2\right)$$

$$\frac{\partial}{\partial\mathbf{p}_{(u)}}=-\sum\nolimits_{m \text{ if } r_{u,m} \text{ available}}2\left[r_{u,m}-\mathbf{p}_{(u)}\cdot\mathbf{a}_{(m)}\right]\mathbf{a}_{(m)}=-\sum\nolimits_{m \text{ if } r_{u,m} \text{ available}}2e_{u,m}\mathbf{a}_{(m)}$$

$$\frac{\partial}{\partial\mathbf{a}_{(m)}}=-\sum\nolimits_{u \text{ if } r_{u,m} \text{ available}}2\left[r_{u,m}-\mathbf{p}_{(u)}\cdot\mathbf{a}_{(m)}\right]\mathbf{p}_{(u)}=-\sum\nolimits_{u \text{ if } r_{u,m} \text{ available}}2e_{u,m}\mathbf{p}_{(u)}$$

The gradient descent update

In every step of the gradient descent, we will pick a learning rate/step size γ and update the parameters as follows

$$\mathbf{p}_{(u)}\leftarrow\mathbf{p}_{(u)}-\gamma\left(-\sum\nolimits_{m \text{ if } r_{u,m} \text{ available}}2e_{u,m}\mathbf{a}_{(m)}\right)=\mathbf{p}_{(u)}+\gamma\sum\nolimits_{m \text{ if } r_{u,m} \text{ available}}e_{u,m}\mathbf{a}_{(m)}$$

$$\mathbf{a}_{(m)}\leftarrow\mathbf{a}_{(m)}-\gamma\left(-\sum\nolimits_{u \text{ if } r_{u,m} \text{ available}}2e_{u,m}\mathbf{p}_{(u)}\right)=\mathbf{a}_{(m)}+\gamma\sum\nolimits_{u \text{ if } r_{u,m} \text{ available}}e_{u,m}\mathbf{p}_{(u)}$$

Stochastic gradient descent

- Notice that to compute the gradient, we need to take a sum over every rating in the dataset
- This is very common in ML problems
- When datasets are *massive*, it can take an enormous amount of time to calculate this gradient, making gradient descent very slow
- **Stochastic gradient descent** takes a different approach – it calculates the gradient using only **one** datapoint at a time
- This can make it much easier to apply gradient descent with massive datasets

The stochastic gradient descent update

In every step of the gradient descent, we will pick a learning rate/step size γ and update the parameters as follows

for every rating $r_{u,m}$:

$$\mathbf{p}_{(u)} \leftarrow \mathbf{p}_{(u)} + \gamma e_{u,m} \mathbf{a}_{(m)}$$
$$\mathbf{a}_{(m)} \leftarrow \mathbf{a}_{(m)} + \gamma e_{u,m} \mathbf{p}_{(u)}$$

Continue doing this again and again until the RMSE stops getting better.

Why is this called “stochastic gradient descent”?

- The idea is that instead of using the “true” gradient (calculated using every data point)…
- …we use an estimate of the gradient (the “expected” gradient), calculated by taking a small number of datapoints, and finding the gradient based on those
- In this case, the “small number of datapoints” is just 1
- It can be shown that under certain conditions, this works just as well as normal gradient descent
- In practice, we use more than 1 datapoint in each step – this is called **minibatch stochastic gradient descent**.

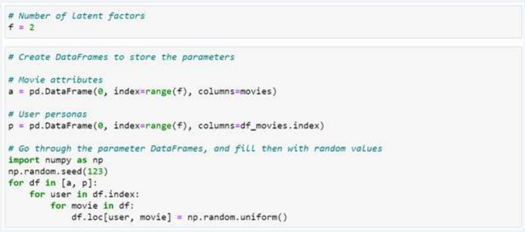
Stochastic gradient descent makes gradient descent easier to apply on massive datasets by updating the variables one datapoint at a time

Stochastic gradient descent for matrix factorization in Python (optional)

Important note

The most efficient way of carrying out these operations is using high-performance Python libraries like `numpy`. We will use **much slower** – but easier to understand – techniques to cover these concepts without too many prerequisites.

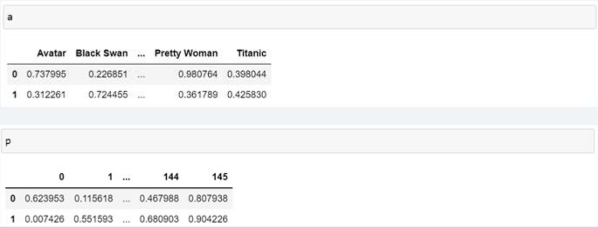
Initializing the algorithm with random parameters



```
# Number of latent factors
f = 2
```

```
# Create DataFrames to store the parameters

# Movie attributes
a = pd.DataFrame(0, index=range(f), columns=movies)

# User personas
p = pd.DataFrame(0, index=range(f), columns=df_movies.index)

# Go through the parameter DataFrames, and fill then with random values
import numpy as np
np.random.seed(123)
for df in [a, p]:
    for user in df.index:
        for movie in df:
            df.loc[user, movie] = np.random.uniform()
```

Initializing the algorithm with random parameters



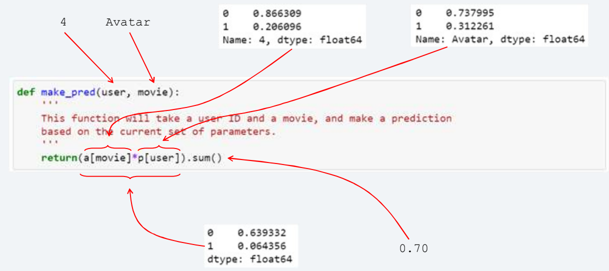
```
a
```

	Avatar	Black Swan	...	Pretty Woman	Titanic
0	0.737995	0.226851	...	0.980764	0.398044
1	0.312261	0.724455	...	0.361789	0.425830

```
p
```

	0	1	...	144	145
0	0.623953	0.115618	...	0.467988	0.807938
1	0.007426	0.551593	...	0.680903	0.904226

Making predictions



```
4        Avatar
```

```
0    0.866309
1    0.206096
Name: 4, dtype: float64
```

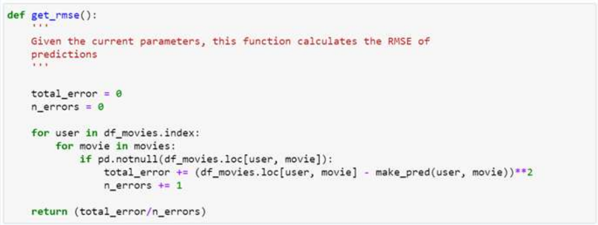
```
0    0.737995
1    0.312261
Name: Avatar, dtype: float64
```

```
def make_pred(user, movie):
    '''
    This function will take a user ID and a movie, and make a prediction
    based on the current set of parameters.
    '''
    return(a[movie]*p[user]).sum()
```

```
0    0.639332
1    0.064356
dtype: float64
```

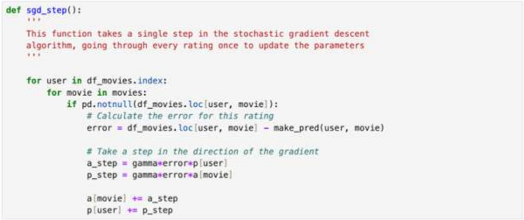
```
0.70
```

Getting the RMSE



```
def get_rmse():
    '''
    Given the current parameters, this function calculates the RMSE of
    predictions
    '''

    total_error = 0
    n_errors = 0

    for user in df_movies.index:
        for movie in movies:
            if pd.notnull(df_movies.loc[user, movie]):
                total_error += (df_movies.loc[user, movie] - make_pred(user, movie))**2
                n_errors += 1

    return (total_error/n_errors)
```

Stochastic gradient descent



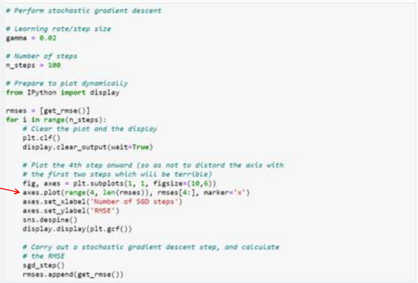
```
def sgd_step():
    '''
    This function takes a single step in the stochastic gradient descent
    algorithm, going through every rating once to update the parameters
    '''

    for user in df_movies.index:
        for movie in movies:
            if pd.notnull(df_movies.loc[user, movie]):
                # Calculate the error for this rating
                error = df_movies.loc[user, movie] - make_pred(user, movie)

                # Take a step in the direction of the gradient
                a_step = gamma*error*p[user]
                p_step = gamma*error*a[movie]

                a[movie] += a_step
                p[user] += p_step
```

$$\mathbf{p}_{(u)} \leftarrow \mathbf{p}_{(u)} + \gamma e_{u,m} \mathbf{a}_{(m)}$$

$$\mathbf{a}_{(m)} \leftarrow \mathbf{a}_{(m)} + \gamma e_{u,m} \mathbf{p}_{(u)}$$

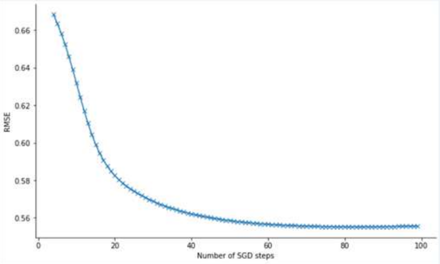
Stochastic gradient descent

Only plot the fourth step of the algorithm onwards. The early errors will be very large, so if we plot them the y-axis will be so large that we won't see the variation in the later steps...



```
# Perform stochastic gradient descent

# Learning rate/step size
gamma = 0.02

# Number of steps
n_steps = 100

# Prepare to plot dynamically
from IPython import display

rmses = [get_rmse()]
for i in range(n_steps):
    # Clear the plot and the display
    plt.clf()
    display.clear_output(wait=True)

    # Plot the 4th step onward (so as not to distord the axis with
    # the first two steps which will be terrible)
    fig, axes = plt.subplots(1, 1, figsize=(10,6))
    axes.plot(range(4, len(rmses)), rmses[4:], marker='x')
    axes.set_xlabel('Number of SGD steps')
    axes.set_ylabel('RMSE')
    sns.despine()
    display.display(plt.gcf())

    # Carry out a stochastic gradient descent step, and calculate
    # the RMSE
    sgd_step()
    rmses.append(get_rmse())
```

Stochastic gradient descent



0.5557

Adding fixed effects

Model limitations

- The current model only allows us to capture user preferences *as a function of the latent factors*.
- But in some cases, the movie is more liked just because it's a better movie – not because it's more "fiction" or more "long" or some other factor
- Similarly, in some cases, a user might like a movie just because they're an easier "grader"

Model limitations part 2

- The model also doesn't allow any "side information" to be used
- For example, we know whether each of our users are men or women
- Can we use that information to capture more signal in the model?

Matrix factorization with fixed effects and side info

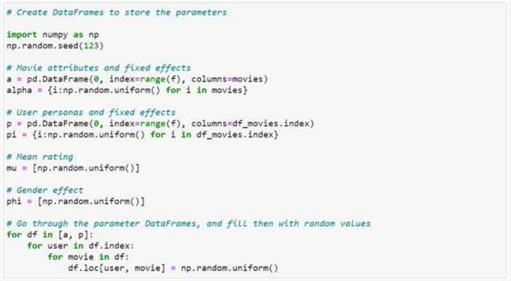
$$\hat{r}_{u,m} = \mu + \pi_u + \alpha_m + \mathbf{p}_{(u)} \cdot \mathbf{a}_{(m)} + \begin{cases} \phi & \text{if the } u \text{ is a woman} \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial}{\partial \mathbf{p}_{(u)}} = -2\sum_{m \text{ if } r_{u,m} \text{ available}} \mathbf{e}_{u,m}\mathbf{a}_{(m)} \qquad \frac{\partial}{\partial \mu} = -2\sum_{u,m \text{ if } r_{u,m} \text{ available}} \mathbf{e}_{u,m}$$

$$\frac{\partial}{\partial \mathbf{a}_{(m)}} = -2\sum_{u \text{ if } r_{u,m} \text{ available}} \mathbf{e}_{u,m}\mathbf{p}_{(u)} \qquad \frac{\partial}{\partial \pi_u} = -2\sum_{m \text{ if } r_{u,m} \text{ available}} \mathbf{e}_{u,m}$$

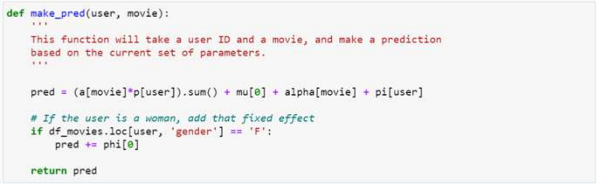
$$\frac{\partial}{\partial \phi} = -2\sum_{\substack{u,m \text{ if } r_{u,m} \text{ available} \\ \text{and } u \text{ is a woman}}} \mathbf{e}_{u,m} \qquad \frac{\partial}{\partial \alpha_m} = -2\sum_{u \text{ if } r_{u,m} \text{ available}} \mathbf{e}_{u,m}$$

Initializing the algorithm with random parameters



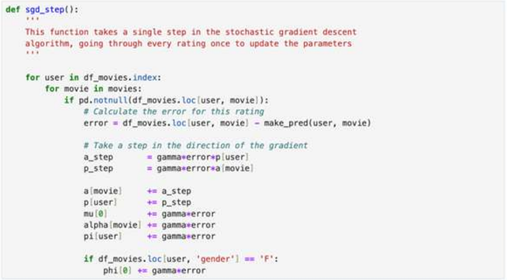
```
# Create DataFrames to store the parameters

import numpy as np
np.random.seed(123)

# Movie attributes and fixed effects
a = pd.DataFrame(0, index=range(f), columns=movies)
alpha = {i:np.random.uniform() for i in movies}

# User personas and fixed effects
p = pd.DataFrame(0, index=range(f), columns=df_movies.index)
pi = {i:np.random.uniform() for i in df_movies.index}

# Mean rating
mu = [np.random.uniform()]

# Gender effect
phi = [np.random.uniform()]

# Go through the parameter DataFrames, and fill then with random values
for df in [a, p]:
    for user in df.index:
        for movie in df:
            df.loc[user, movie] = np.random.uniform()
```

Making predictions



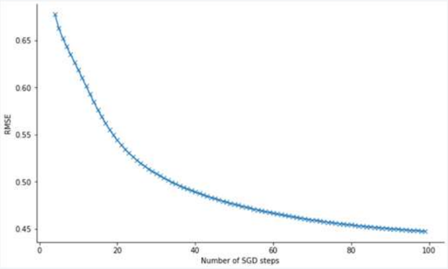
```
def make_pred(user, movie):
    '''
    This function will take a user ID and a movie, and make a prediction
    based on the current set of parameters.
    '''

    pred = (a[movie]*p[user]).sum() + mu[0] + alpha[movie] + pi[user]

    # If the user is a woman, add that fixed effect
    if df_movies.loc[user, 'gender'] == 'F':
        pred += phi[0]

    return pred
```

Stochastic gradient descent



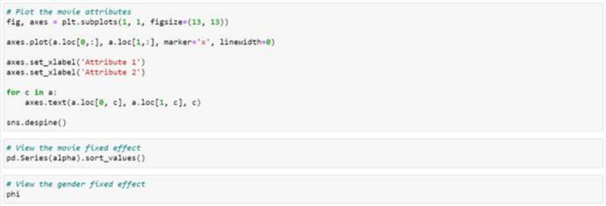
```
def sgd_step():
    '''
    This function takes a single step in the stochastic gradient descent
    algorithm, going through every rating once to update the parameters
    '''

    for user in df_movies.index:
        for movie in movies:
            if pd.notnull(df_movies.loc[user, movie]):
                # Calculate the error for this rating
                error = df_movies.loc[user, movie] - make_pred(user, movie)

                # Take a step in the direction of the gradient
                a_step      = gamma*error*p[user]
                p_step      = gamma*error*a[movie]

                a[movie]     += a_step
                p[user]      += p_step
                mu[0]        += gamma*error
                alpha[movie] += gamma*error
                pi[user]     += gamma*error

                if df_movies.loc[user, 'gender'] == 'F':
                    phi[0] += gamma*error
```

Stochastic gradient descent



0.4471

Visualizing the latent factors



```
# Plot the movie attributes
fig, axes = plt.subplots(1, 1, figsize=(13, 13))

axes.plot(a.loc[0,:], a.loc[1,:], marker='x', linewidth=0)

axes.set_xlabel('Attribute 1')
axes.set_xlabel('Attribute 2')

for c in a:
    axes.text(a.loc[0, c], a.loc[1, c], c)

sns.despine()
```

```
# View the movie fixed effect
pd.Series(alpha).sort_values()
```

```
# View the gender fixed effect
phi
```

Columbia Business School
AT THE VERY CENTER OF BUSINESS

Other examples of recommendation systems

The Netflix prize

Netflix offered $1,000,000 to anyone who could improve their recommendation algorithms

> *"The RMSE of Cinematch on the test subset, based on training the Cinematch algorithm using the training set alone, was **0.9525** …*
>
> *The qualify for the Grand Prize, the RMSE of Participant's submitted predictions on the test subset much be less than or equal to 90% of 0.9525, or **0.8572**"*

Netflix Prize winner: BellKor's Pragmatic Chaos. RMSE **0.8567**. They used many of the techniques we discussed here.

Amazon item-to-item collaborative filtering

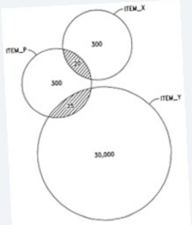
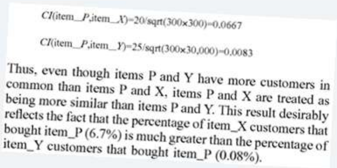


FIG. 4



$CI(\text{item\_P},\text{item\_X})=20/\text{sqrt}(300\times300)=0.0667$

$CI(\text{item\_P},\text{item\_Y})=25/\text{sqrt}(300\times30{,}000)=0.0083$

Thus, even though items P and Y have more customers in common than items P and X, items P and X are treated as being more similar than items P and Y. This result desirably reflects the fact that the percentage of item_X customers that bought item_P (6.7%) is much greater than the percentage of item_Y customers that bought item_P (0.08%).

https://patents.google.com/patent/US7113917

Amazon item-to-item collaborative filtering

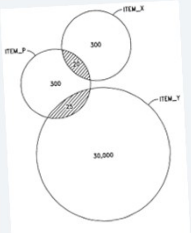
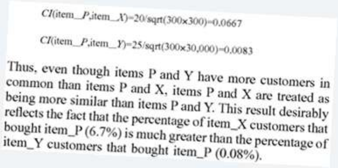


FIG.4



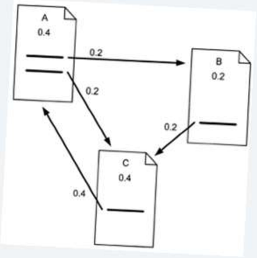
CI(item_*P*,item_*X*)=20/sqrt(300×300)=0.0667

CI(item_*P*,item_*Y*)=25/sqrt(300×30,000)=0.0083

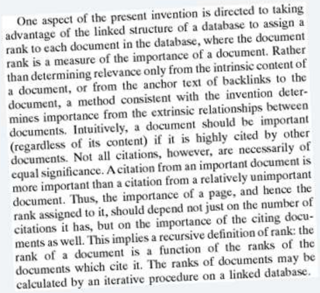
Thus, even though items P and Y have more customers in common than items P and X, items P and X are treated as being more similar than items P and Y. This result desirably reflects the fact that the percentage of item_X customers that bought item_P (6.7%) is much greater than the percentage of item_Y customers that bought item_P (0.08%).

https://patents.google.com/patent/US7113917

Google PageRank



https://patents.google.com/patent/US6285999



One aspect of the present invention is directed to taking advantage of the linked structure of a database to assign a rank to each document in the database, where the document rank is a measure of the importance of a document. Rather than determining relevance only from the intrinsic content of a document, or from the anchor text of backlinks to the document, a method consistent with the invention determines importance from the extrinsic relationships between documents. Intuitively, a document should be important (regardless of its content) if it is highly cited by other documents. Not all citations, however, are necessarily of equal significance. A citation from an important document is more important than a citation from a relatively unimportant document. Thus, the importance of a page, and hence the rank assigned to it, should depend not just on the number of citations it has, but on the importance of the citing documents as well. This implies a recursive definition of rank: the rank of a document is a function of the ranks of the documents which cite it. The ranks of documents may be calculated by an iterative procedure on a linked database.

Fall 2024

Simulation; Medical Testing & Pension Analytics

Session 9

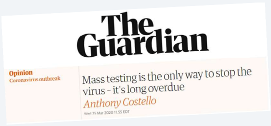
Professor Daniel Guetta

This Module

- COVID-19: every test counts
- Decision making through Monte Carlo simulation
- Evaluating GM's healthcare pension liabilities

COVID-19: every test counts

The importance of testing



The Guardian

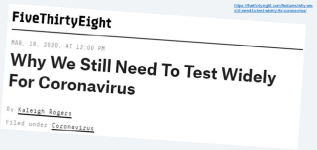
Opinion
Coronavirus outbreak

Mass testing is the only way to stop the virus – it's long overdue

Anthony Costello

Wed 25 Mar 2020 11.55 EDT

https://www.theguardian.com/commentisfree/2020/mar/25/mass-covid-19-testing-is-vital-but-the-data-must-be-localised



FiveThirtyEight

MAR. 18, 2020, AT 12:00 PM

Why We Still Need To Test Widely For Coronavirus

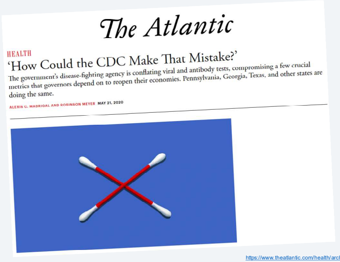
By Kaleigh Rogers

Filed under Coronavirus

https://fivethirtyeight.com/features/why-we-still-need-to-test-widely-for-coronavirus/

- When cases are still rare, allows for far less onerous social distancing
- Essential part of any "track and trace" approach
- Important part of protecting healthcare and other social workers
- Basis of pretty much everything we say, know, and decide about the virus

(At least) two types of COVID-19 tests



The Atlantic

HEALTH

'How Could the CDC Make That Mistake?'

The government's disease-fighting agency is conflating viral and antibody tests, compromising a few crucial metrics that governors depend on to reopen their economies. Pennsylvania, Georgia, Texas, and other states are doing the same.

ALEXIS C. MADRIGAL AND ROBINSON MEYER MAY 21, 2020

https://www.theatlantic.com/health/archive/2020/05/cdc-and-states-are-misreporting-covid-19-test-data-pennsylvania-georgia-texas/611935/

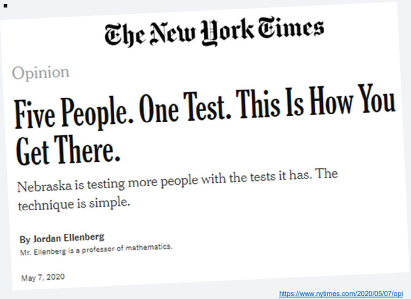
Viral tests are the most useful for track and trace – we'll focus on these today

Viral tests were in short supply

- The first viral test for COVID-19 was available in record time – early cases were reported in late December, the full gene sequence was submitted by China to the WHO on January 12th, and there were reports of viral testing happening on January 17th
- Many specific "recipes" for this test have emerged since, with varying degrees of success.
- Unfortunately, there are many hurdles between a test that works in principle and a test that can be applied usefully at scale – testing was plagued by a whole host of issues from the getgo
  - Shortage of collection kit (eg: nasal swabs)
  - Shortage of reagents and/or staff to analyze collected samples
  - Contaminated/flaws tests
  - Bureaucratic hurdles

How can we do more with the testing capacity we have?

An idea…



The New York Times

Opinion

Five People. One Test. This Is How You Get There.

Nebraska is testing more people with the tests it has. The technique is simple.

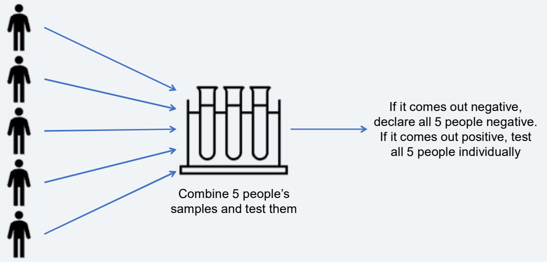
By Jordan Ellenberg

Mr. Ellenberg is a professor of mathematics.

May 7, 2020

https://www.nytimes.com/2020/05/07/opinion/coronavirus-group-testing.html

An idea…



Combine 5 people's samples and test them

If it comes out negative, declare all 5 people negative. If it comes out positive, test all 5 people individually

Any thoughts on this technique?

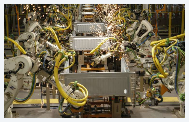
Pros and cons

- Pros
  - Test the population with fewer tests
  - Can vary the group size if needed
- Cons
  - Do diluted samples work? (c.f. Wassermann test for syphilis in WW2)
    - Nebraska required special permission to do this
  - Does it *really* result in fewer tests?
  - Does it affect the accuracy of the test?
  - What kind of shortage does this help?

Does this really reduce the number of tests needed, and by how much?

Simulation

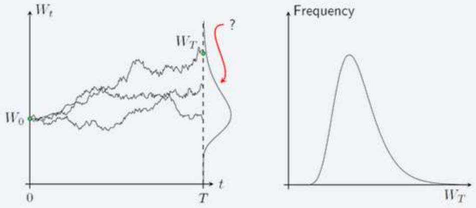
Monte Carlo simulation



- Simulation is the imitation of real-world process
- Analyze the consequences of decisions before real-world implementation
- Two reasons for simulation
  - Random events impact the outcome of interest and need to understand the range of future outcomes (**Monte Carlo simulation**)
  - Even in the absence of randomness, there might be no simple formula for the outcome of interest, and simulation is the only way to do testing (dynamical systems)

Simulation in the presence of randomness

Key idea: simulate "many" possible paths to understand the possible scenarios you could face.



Monte Carlo simulation process

Construct a model connecting inputs to outputs

- Output of interest and random inputs that impact the output
- How the random inputs impact the outputs
- Nature of random inputs: distribution

Run the simulation

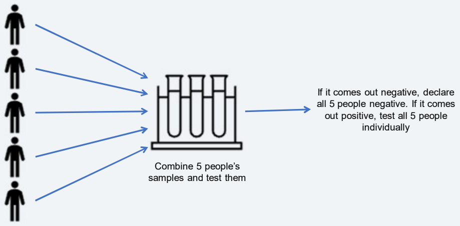
- Generate many possible values that random inputs may take
- For each sequence of events, record outputs

Analyze the output

- Simulation shows how random inputs lead to a range of outcomes for the random outputs
- Distribution of the outputs: average, standard deviation, percentiles…

Back to COVID

An idea…



Combine 5 people's samples and test them

If it comes out negative, declare all 5 people negative. If it comes out positive, test all 5 people individually

On average, how many tests does it take to test a single person conclusively?

What are the sources of randomness that might lead this number being different each time?

Sources of randomness

- Whether the patient of interest has COVID
  - This depends on how much of the population is infected with COVID
  - We'll denote this variable s_1; equal to 1 if the patient has COVID, and 0 otherwise
- Whether the other four patients being tested have COVID
  - We'll denote these variables s_2, s_3, s_4, and s_5; each variable will be 1 if the relevant patient has COVID, and 0 otherwise
- Whether the combined sample tests are positive or negative
  - This depends on the sensitivity and specificity of the test *and* on whether anyone in the sample is positive
  - We'll denote this T, equal to 1 if the test is positive, and 0 otherwise

Sensitivity and specificity of the test

The accuracy of diagnostic tests is encapsulated by two numbers

- **The sensitivity** (true positive rate): this is the probability someone tests positive if they do indeed have the condition
- **The specificity** (true negative rate): this is the probability someone tests negative if they do *not* have the condition

Many estimates of these two numbers exist for COVID tests; we'll go with fairly plausible **sensitivity = 0.9**, and **specificity = 0.98**, but we'll play with these later.

Sources of randomness

- The s variables depend on the proportion p of the population that is currently infected with COVID

$$s_1, s_2, s_3, s_4, s_5 \sim \text{Bernoulli}(p)$$

- The T variable depends on whether the sample had any COVID-positive samples

$$T = \begin{cases} \text{Bernoulli}(0.9) & \text{if } \max(s_1, s_2, s_3, s_4, s_5) = 1 \\ \text{Bernoulli}(0.02) & \text{if } \max(s_1, s_2, s_3, s_4, s_5) = 0 \end{cases}$$

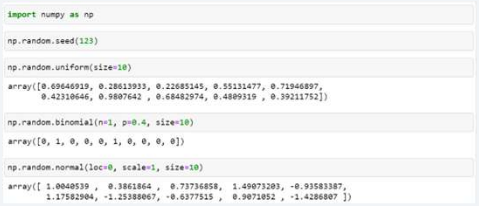
How do these sources of randomness affect the outcome we care about

The outcome

- If T = 0, we don't need to carry out any further test
  - The number of tests required for our person of interest is 0.2
- If T = 1, we need to re-test every one of the five people in the sample
  - The number of tests required per person is therefore 1.2

Random numbers in Python

Generating random numbers in Python



```
import numpy as np
```

```
np.random.seed(123)
```

```
np.random.uniform(size=10)
```

```
array([0.69646919, 0.28613933, 0.22685145, 0.55131477, 0.71946897,
       0.42310646, 0.9807642 , 0.68482974, 0.4809319 , 0.39211752])
```

```
np.random.binomial(n=1, p=0.4, size=10)
```

```
array([0, 1, 0, 0, 0, 1, 0, 0, 0, 0])
```

```
np.random.normal(loc=0, scale=1, size=10)
```

```
array([ 1.0040539 ,  0.3861864 ,  0.73736858,  1.49073203, -0.93583387,
        1.17582904, -1.25388067, -0.6377515 ,  0.9071052 , -1.4286807 ])
```

We'll discuss this shortly

`numpy` has in-built functions to generate random numbers from all common distributions. For others, we can use the inverse CDF method (but it'll be **slower** than the built-in ones).

The inverse CDF method

- Let U be a uniformly distributed random variable
- Suppose we have a distribution f with cumulative density function (CDF) $F(x)$… In other words, if a variable X has distribution f, then

$$P(X \leq x) = F(x)$$

- Let $F^{-1}(p)$ be the inverse function of $F(x)$
- Then it can be shown that $F^{-1}(U)$ has distribution f !
- To see why, recall that the CDF of a uniform distribution is $F(x) = x$, and so

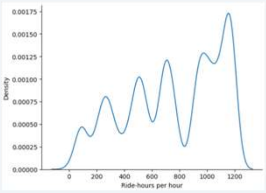
$$P(F^{-1}(U) \leq x) = P(U \leq F(x)) = F(x)$$

The inverse CDF method – an example

- You oversee operations at a ridesharing company
- You are given a file, `demand.csv` that contains one column – `total_trip_hours` – which contains historical ride demand
- Every row corresponds to one hour in the last four years, and it lists the number of trip-hours that were taken during that hour
  - For example, if during that hour, 10 people took 10 minute rides, the total number of ride-minutes in that hour is 10×10 = 100 ride-minutes = 1.67 ride-hours – so that row would contain 1.67
- You want to be able to simulate an "average day" in your company's operations – in particular, you want to be able to simulate a random variable that represents the number of ride-hours in any given hour

The inverse CDF method – an example

Let's have a look at the distribution of hours



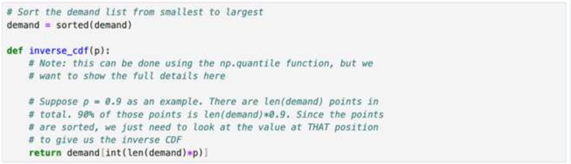
This is a complicated distribution! How do we generate variables from it?

The inverse CDF method – an example

- To use the inverse-CDF method, we need to calculate $F^{-1}(p)$
- Remember; $F(x) = P(\text{Ride-hours per hour} \leq x)$
- Thus, $F^{-1}(p)$ is the number of ride-hours such that a proportion p of ride-hours is less than that number
  - $F^{-1}(0.5)$ is the *median* number of ride-hours
  - $F^{-1}(0.9)$ is the number of ride-hours such that 90% of ride-hours is less than that
- We can calculate this in Python!

The inverse CDF method – an example

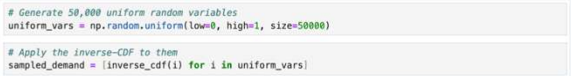
Calculating $F^{-1}(p)$



```
# Sort the demand list from smallest to largest
demand = sorted(demand)

def inverse_cdf(p):
    # Note: this can be done using the np.quantile function, but we
    # want to show the full details here

    # Suppose p = 0.9 as an example. There are len(demand) points in
    # total. 90% of those points is len(demand)*0.9. Since the points
    # are sorted, we just need to look at the value at THAT position
    # to give us the inverse CDF
    return demand[int(len(demand)*p)]
```

The inverse CDF method – an example

We can now use the inverse CDF method to generate variables from this distribution

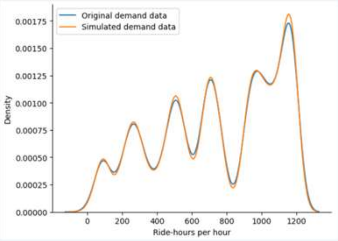


```
# Generate 50,000 uniform random variables
uniform_vars = np.random.uniform(low=0, high=1, size=50000)
```

```
# Apply the inverse-CDF to them
sampled_demand = [inverse_cdf(i) for i in uniform_vars]
```

The inverse CDF method – an example

We can verify it worked…



The randomization seed

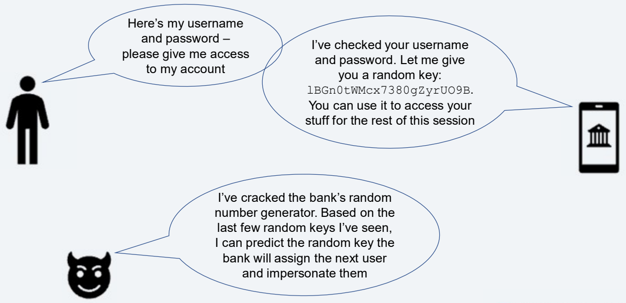
- Computers cannot generate random numbers.
- Instead, we give the computer a **seed**. It then uses complex mathematical formulas to generate **pseudo-random numbers**.
- If you give a computer the same seed, the same sequence of random numbers will be generated.
- In `numpy`, the randomization seed can be set using

`np.random.seed()`

Generating random numbers

- We will heavily rely on Python's ability to generate sequences of pseudo-random numbers
- We require a sequence that is **completely unpredictable** – even if you see every number in the sequence so far, there should be **no way to predict the next number**.
- Computers have no way to generate random numbers – so instead they use **complicated mathematical functions** to generate these sequences.
- Doing this properly is **hard**.
- Doing this is **really, really, really important**.

Why are random numbers so important?



Here's my username and password – please give me access to my account

I've checked your username and password. Let me give you a random key: `1BGn0tWMcx7380gZyrUO9B`. You can use it to access your stuff for the rest of this session

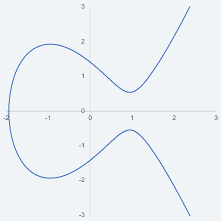
I've cracked the bank's random number generator. Based on the last few random keys I've seen, I can predict the random key the bank will assign the next user and impersonate them

Elliptic curves

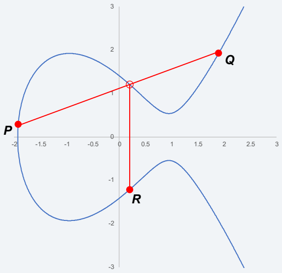
Elliptic curves are graphs that satisfy

$$y^2 = x^3 + ax + b$$

Example with
a = –2.7 and *b* = 2



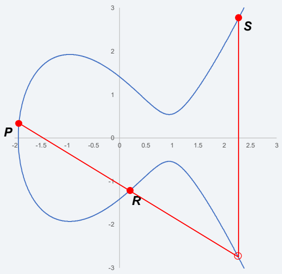
Summing points on elliptic curves



- Suppose we have two points ***P*** and ***Q*** on an elliptic curve
- We **define** summation on an elliptic curve in a weird way
  - Draw the line from ***P*** to ***Q***
  - This line will cross the graph once at a single point
  - Reflect that point on the *x*-axis to get ***R***; the sum
- So we say

$$\boldsymbol{P} + \boldsymbol{Q} = \boldsymbol{R}$$

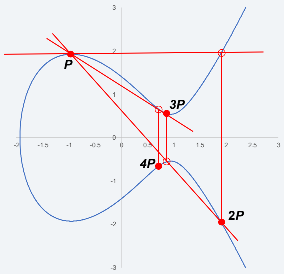
Summing points on elliptic curves



- Another example: let's find ***P*** + ***R***
  - Draw the line between ***P*** and ***R***
  - Find the point at which it crosses the curve
  - Reflect it in the *x*-axis
- So

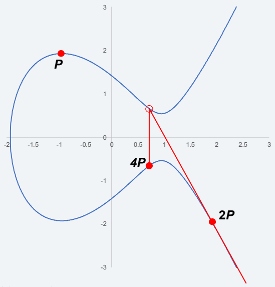
$$\boldsymbol{P} + \boldsymbol{R} = \boldsymbol{S}$$

Summing points on elliptic curves



- What if we want to add a point to itself (i.e. ***P*** + ***P***)?
- We simply find the tangent line to ***P***, and reflect the crossing point in the *x*-axis
- We can then find 2***P*** + ***P*** = 3***P***
- And 3***P*** + ***P*** = 4***P***

Summing points on elliptic curves



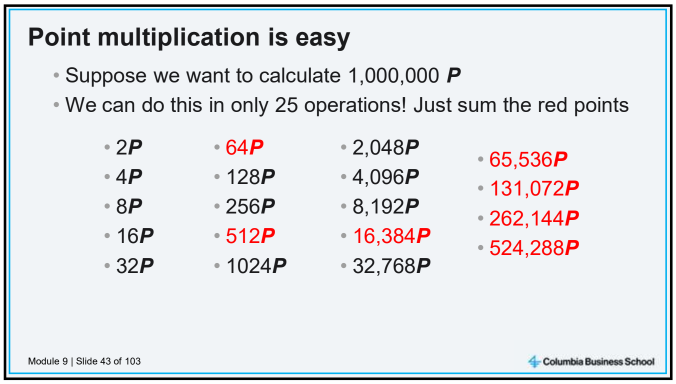
- But there's another way to find 4***P***; we can just add 2***P*** to itself
- If our concept of addition is "consistent", this should lead to the same point
- And it does!

Nerd notes (very, very, optional)

- In practice, cryptography uses elliptic curves mod *p* over the integers; it can be shown that – as long as we add an identity element at infinity – these integers form a finite abelian group
- A lot of the underlying math behind this was developed by Evariste Gallois, a French mathematician who died in a duel in 1832 (possibly over a love affair) at the age of 20; he was also a political firebrand, spent time in jail, and we know a lot of this because Alexandre Dumas (who wrote *The Count of Monte Cristo*) talked about it in his diaries… No biggie…

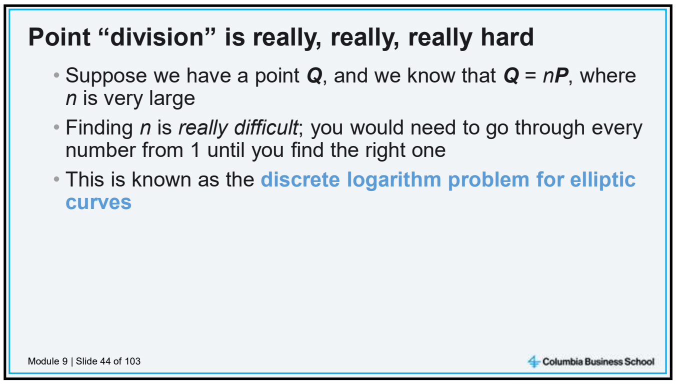
MAN THE PAST IS SO BIG
HOW DO HISTORIANS EVEN COPE?
I HAVE NO IDEA.
I HONESTLY HAVE ENOUGH TROUBLE JUST WITH THE PRESENT.

https://xkcd.com/1979/



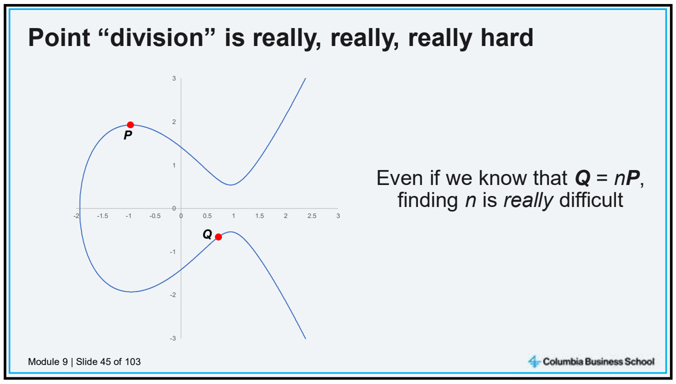
Point multiplication is easy

- Suppose we want to calculate 1,000,000 ***P***
- We can do this in only 25 operations! Just sum the red points
  - 2***P***
  - 4***P***
  - 8***P***
  - 16***P***
  - 32***P***
  - 64***P***
  - 128***P***
  - 256***P***
  - 512***P***
  - 1024***P***
  - 2,048***P***
  - 4,096***P***
  - 8,192***P***
  - 16,384***P***
  - 32,768***P***
  - 65,536***P***
  - 131,072***P***
  - 262,144***P***
  - 524,288***P***



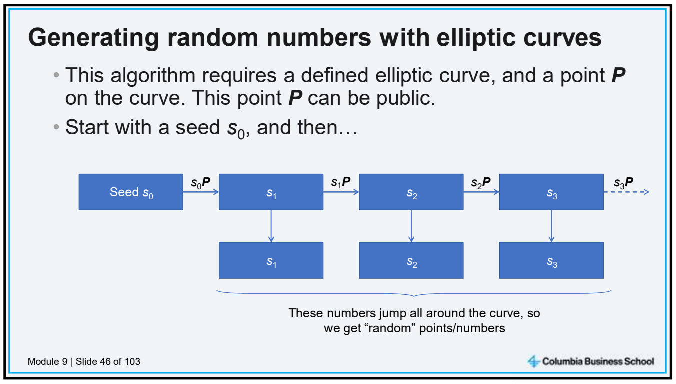
Point "division" is really, really, really hard

- Suppose we have a point ***Q***, and we know that ***Q*** = *n****P***, where *n* is very large
- Finding *n* is *really difficult*; you would need to go through every number from 1 until you find the right one
- This is known as the **discrete logarithm problem for elliptic curves**



Point "division" is really, really, really hard

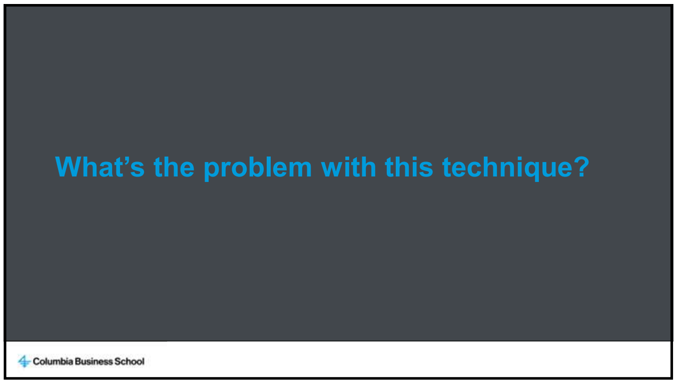
Even if we know that ***Q*** = *n****P***, finding *n* is *really* difficult



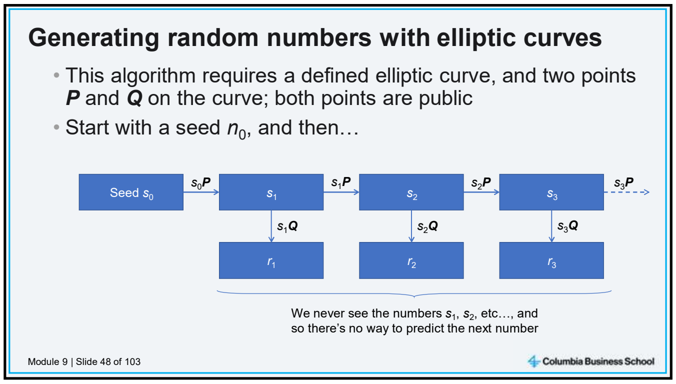
Generating random numbers with elliptic curves

- This algorithm requires a defined elliptic curve, and a point ***P*** on the curve. This point ***P*** can be public.
- Start with a seed s_0, and then…

These numbers jump all around the curve, so we get "random" points/numbers



What's the problem with this technique?

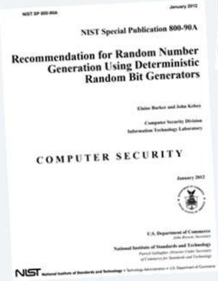


Generating random numbers with elliptic curves

- This algorithm requires a defined elliptic curve, and two points ***P*** and ***Q*** on the curve; both points are public
- Start with a seed n_0, and then…

We never see the numbers s_1, s_2, etc…, and so there's no way to predict the next number

This is real…



NIST SP 800-90A January 2012

NIST Special Publication 800-90A

Recommendation for Random Number Generation Using Deterministic Random Bit Generators

Elaine Barker and John Kelsey

Computer Security Division
Information Technology Laboratory

COMPUTER SECURITY

January 2012

U.S. Department of Commerce
John Bryson, Secretary

National Institute of Standards and Technology
Patrick Gallagher, Director Under Secretary
of Commerce for Standards and Technology

NIST National Institute of Standards and Technology • Technology Administration • U.S. Department of Commerce

https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-90a.pdf

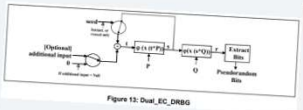


Figure 13: Dual_EC_DRBG

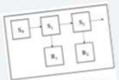


Figure 14: Dual_EC_DRBG Backtracking Resistance

Each of following curves is given by the equation:

$$y^2 = x^3 - 3x + b \pmod{p}$$

Notation:

p - Order of the field F_p, given in decimal

n - Order of the Elliptic Curve Group, in decimal .

a – (-3) in the above equation

b - Coefficient above

A.1.1 Curve P-256

```
p = 11579208921035624876269744694940757353008614\
    3415290314195533631308867097853951
n = 11579208921035624876269744694940757352999695\
    5224135760342422259061068512044369
b = 5ac635d8 aa3a93e7 b3ebbd55 769886bc 651d06b0 cc53b0f6 3bce3c3e
    27d2604b
Px = 6b17d1f2 e12c4247 f8bce6e5 63a440f2 77037d81 2deb33a0
     f4a13945 d898c296
Py = 4fe342e2 fe1a7f9b 8ee7eb4a 7c0f9e16 2bce3357 6b315ece
     cbb64068 37bf51f5
Qx = c97445f4 5cdef9f0 d3e05e1e 585fc297 235b82b5 be8ff3ef
     ca67c598 52018192
Qy = b28ef557 ba31dfcb dd21ac46 e2a91e3c 304f44cb 87058ada
     2cb81515 1e610046
```

Except…

The New York Times

N.S.A. Able to Foil Basic Safeguards of Privacy on Web

By Nicole Perlroth, Jeff Larson and Scott Shane

Sept. 5, 2013

https://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html

Many users assume — or have been assured by Internet companies — that their data is safe from prying eyes, including those of the government, and the N.S.A. wants to keep it that way. The **agency** treats its **recent successes** in **deciphering protected information** as among its most closely guarded secrets, restricted to those cleared for a highly classified program code-named Bullrun, according to the **documents, provided by Edward J. Snowden**, the former N.S.A. contractor.

…

One goal in the agency's 2013 budget request was to "**influence policies**, standards and specifications for commercial public key technologies," the most common encryption method.

Cryptographers have long suspected that the agency **planted vulnerabilities** in a standard adopted in 2006 by the **National Institute of Standards and Technology** and later by the International Organization for Standardization, which has 163 countries as members.

REUTERS

EDITOR'S PICKS DECEMBER 20, 2013 / 4:05 PM / UPDATED 8 YEARS AGO

Exclusive: Secret contract tied NSA and security industry pioneer

By Joseph Menn

https://www.reuters.com/article/us-usa-security-rsa-idUSBRE9BJ1C220131220

Documents **leaked** by former NSA contractor **Edward Snowden** show that the NSA created and promulgated a **flawed formula for generating random numbers** to create a "back door" in encryption products, the New York Times reported in September. Reuters later reported that RSA became the most important distributor of that formula by rolling it into a software tool called Bsafe that is used to enhance security in personal computers and many other products.

Undisclosed until now was that **RSA received $10 million in a deal that set the NSA formula as the preferred, or default, method for number generation in the BSafe software**, according to two sources familiar with the contract. Although that sum might seem paltry, it represented more than a third of the revenue that the relevant division at RSA had taken in during the entire previous year, securities filings show.

RSA SPEAKING OF SECURITY

RSA RESPONSE TO MEDIA CLAIMS REGARDING NSA RELATIONSHIP

Home » News Media » RSA Response to Media Claims Regarding NSA Relationship

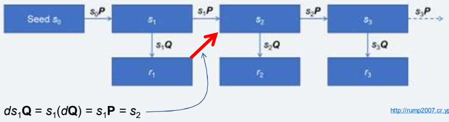
December 22, 2013

https://web.archive.org/web/20131223121638/http://blogs.rsa.com/news-media-2/rsa-response/

Recent press coverage has asserted that RSA entered into a "secret contract" with the NSA to incorporate a known flawed random number generator into its BSAFE encryption libraries. **We categorically deny this allegation.**

Explaining the backdoor

- In theory, ***P*** and ***Q*** should be picked completely randomly
- It can be shown that whatever ***P*** and ***Q*** are, there is always a *d* such that ***P*** = *d****Q***
- Because of the discrete logarithm problem, it's almost impossible to compute *d* – but suppose you choose a ***Q*** and *d*, and then write the NIST standard to pick a ***P*** equal to *d****Q*** (which is easy to compute)
- Then given the last "random" number, you can predict the next…



$dr_1 = ds_1\mathbf{Q} = s_1(d\mathbf{Q}) = s_1\mathbf{P} = s_2$

http://rump2007.cr.yp.to/15-shumow.pdf

How might we *P* and *Q* in a way that ensures this hasn't happened?

The dénouement

The New York Times

Government Announces Steps to Restore Confidence on Encryption Standards

BY NICOLE PERLROTH SEPTEMBER 10, 2013 7:02 PM 9

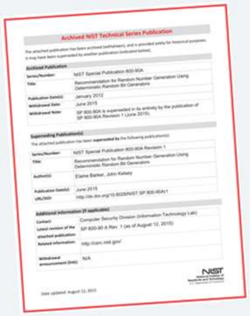
https://bits.blogs.nytimes.com/2013/09/10/government-announces-steps-to-restore-confidence-on-encryption-standards/

Nothing-up-my-sleeve number

From Wikipedia, the free encyclopedia

https://en.wikipedia.org/wiki/Nothing-up-my-sleeve_number

In cryptography, nothing-up-my-sleeve numbers are any numbers which, by their construction, are above suspicion of hidden properties… An example would be the use of initial digits from the number π as the constants. Using digits of π millions of places after the decimal point would not be considered trustworthy because the algorithm designer might have selected that starting point because it created a secret weakness the designer could later exploit.



Archived NIST Technical Series Publication

The attached publication has been archived (withdrawn), and is provided solely for historical purposes. It may have been superseded by another publication (indicated below).

Archived Publication

Series/Number:	NIST Special Publication 800-90A
Title:	Recommendation for Random Number Generation Using Deterministic Random Bit Generators
Publication Date(s):	January 2012
Withdrawal Date:	June 2015
Withdrawal Note:	SP 800-90A is superseded in its entirety by the publication of SP 800-90A Revision 1 (June 2015).

Superseding Publication(s)

The attached publication has been **superseded by** the following publication(s):

Series/Number:	NIST Special Publication 800-90A Revision 1
Title:	Recommendation for Random Number Generation Using Deterministic Random Bit Generators
Author(s):	Elaine Barker, John Kelsey
Publication Date(s):	June 2015
URL/DOI:	http://dx.doi.org/10.6028/NIST.SP.800-90Ar1

Additional Information (if applicable)

Contact:	Computer Security Division (Information Technology Lab)
Latest revision of the attached publication:	SP 800-90 A Rev. 1 (as of August 12, 2015)
Related information:	http://csrc.nist.gov/
Withdrawal announcement (link):	N/A

NIST

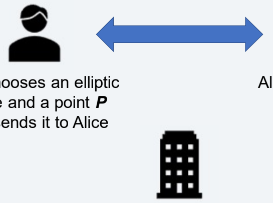
Date updated: August 12, 2015

https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-90a.pdf

Elliptic curves are everywhere: key exchange

- Suppose Alice and Bob live on opposite sides of the world and want to exchange a message
- The NSA is eager to know what Alice wants to tell Bob, and can read any messages they send to each other
- Alice and Bob could encrypt the message, but what encryption key would they use? If they exchange the encryption key, the NSA will intercept it too
- Astonishingly, the **Elliptic-Curve Diffie-Hellman algorithm** will allow them to do this securely!
- This algorithm is in use today – when you go to an https site, you might be using it!

Elliptic curves are everywhere: key exchange

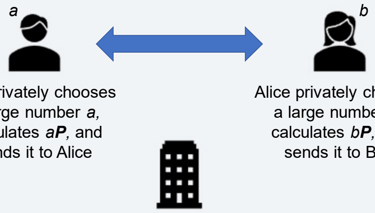


Bob chooses an elliptic curve and a point ***P*** and sends it to Alice

Alice stores this curve and the point ***P***

The NSA intercepts both

Elliptic curves are everywhere: key exchange

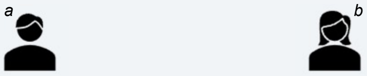


Bob privately chooses a large number *a*, calculates ***aP***, and sends it to Alice

Alice privately chooses a large number *b*, calculates ***bP***, and sends it to Bob

The NSA intercepts both. Even though it knows ***P***, ***aP***, and ***bP***, it can't figure out *a* and *b* because of the discrete logarithm problem

Elliptic curves are everywhere: key exchange



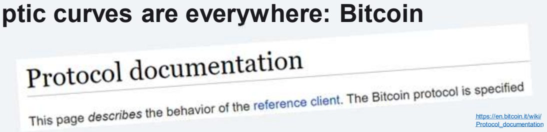
Bob calculates *a* × the point he received, and gets ***abP***

Alice calculates *b* × the point she received, and gets ***baP***

Alice and Bob now have the same point ***abP*** = ***baP***, which they can use as their encryption key!

The NSA doesn't know *a* or *b*, and so they can't find this number

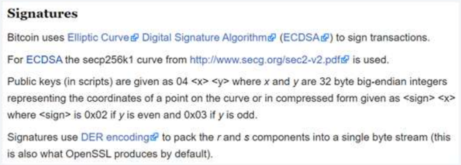
Elliptic curves are everywhere: Bitcoin



Protocol documentation

This page *describes* the behavior of the reference client. The Bitcoin protocol is specified

https://en.bitcoin.it/wiki/Protocol_documentation



Signatures

Bitcoin uses Elliptic Curve Digital Signature Algorithm (ECDSA) to sign transactions.

For ECDSA the secp256k1 curve from http://www.secg.org/sec2-v2.pdf is used.

Public keys (in scripts) are given as 04 <x> <y> where *x* and *y* are 32 byte big-endian integers representing the coordinates of a point on the curve or in compressed form given as <sign> <x> where <sign> is 0x02 if *y* is even and 0x03 if *y* is odd.

Signatures use DER encoding to pack the *r* and *s* components into a single byte stream (this is also what OpenSSL produces by default).

Back to COVID; simulation in Python

Reminder

- The *s* variables depend on the proportion *p* of the population that is currently infected with COVID

$$s_1, s_2, s_3, s_4, s_5 \sim \text{Bernoulli}(p)$$

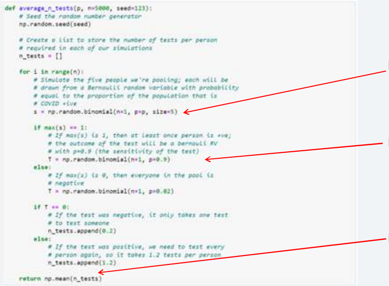
- The *T* variable depends on whether the sample had any COVID-positive samples

$$T = \begin{cases} \text{Bernoulli}(0.9) & \text{if } \max(s_1, s_2, s_3, s_4, s_5) = 1 \\ \text{Bernoulli}(0.02) & \text{if } \max(s_1, s_2, s_3, s_4, s_5) = 0 \end{cases}$$

- If *T* = 0, we don't need to carry out any further test, and the number of tests for the person of interest is 0.2. If *T* = 1, we need to re-test everyone; the tests required per person is 1.2

How can we use Python to simulate thousands of instances of this test, to see what the average number of tests is?

Simulating the tests



```
def average_n_tests(p, n=5000, seed=123):
    # Seed the random number generator
    np.random.seed(seed)

    # Create a list to store the number of tests per person
    # required in each of our simulations
    n_tests = []

    for i in range(n):
        # Simulate the five people we're pooling; each will be
        # drawn from a Bernoulli random variable with probability
        # equal to the proportion of the population that is
        # COVID +ive
        s = np.random.binomial(n=1, p=p, size=5)

        if max(s) == 1:
            # If max(s) is 1, then at least once person is +ve;
            # the outcome of the test will be a bernouli RV
            # with p=0.9 (the sensitivity of the test)
            T = np.random.binomial(n=1, p=0.9)
        else:
            # If max(s) is 0, then everyone in the pool is
            # negative
            T = np.random.binomial(n=1, p=0.02)

        if T == 0:
            # If the test was negative, it only takes one test
            # to test someone
            n_tests.append(0.2)
        else:
            # If the test was positive, we need to test every
            # person again, so it takes 1.2 tests per person
            n_tests.append(1.2)

    return np.mean(n_tests)
```

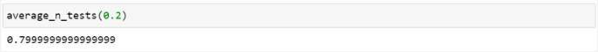
Example: `[0, 0, 1, 0, 1]`

Example: `1`

Example: `[0.2, 0.2, 1.2, ..., 1.2, 0.2]`

Running the simulation

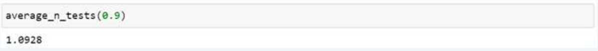
Suppose 20% of the population is currently infected with COVID. How many tests will the pooled procedure require on average?



```
average_n_tests(0.2)
```

```
0.7999999999999999
```

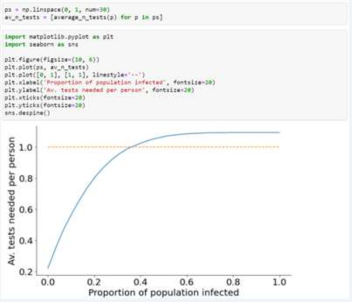
What if 90% of the population is infected?



```
average_n_tests(0.9)
```

```
1.0928
```

Managerial insights



```
ps = np.linspace(0, 1, num=30)
av_n_tests = [average_n_tests(p) for p in ps]
```

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
plt.plot(ps, av_n_tests)
plt.plot([0, 1], [1, 1], linestyle='--')
plt.xlabel('Proportion of population infected', fontsize=20)
plt.ylabel('Av. tests needed per person', fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
sns.despine()
```

Simulation can provide valuable managerial insights that would otherwise take costly experiments to obtain

AT THE VERY CENTER OF BUSINESS

Simulation accuracy

Simulation accuracy

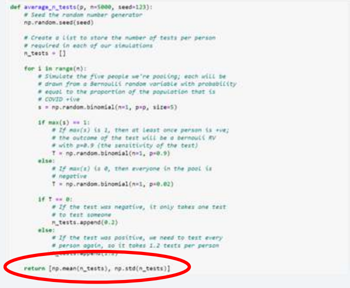
- Our simulation has told us if 20% of the population is infected, we will need 0.8 tests per person
- If we run it again with a different seed, we might get a slightly different number
- How can we get an estimate of roughly how accurate our result is?
- The key, it turns out, is the Central Limit Theorem
- If we use *n* simulation trials, and the standard deviation of the results from each trial is σ, the mean will be normally distributed with a standard deviation of $\sigma/\sqrt{n}$.

Simulation accuracy

This means that if the mean of all the simulations is μ, and the standard deviation is σ, we know that 95% of times we run this simulation, the mean will be in the interval

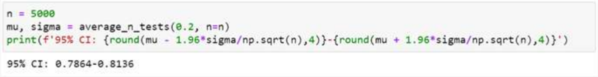
$$\mu \pm 1.96\frac{\sigma}{\sqrt{n}}$$

Simulation accuracy



```
def average_n_tests(p, n=5000, seed=123):
    # Seed the random number generator
    np.random.seed(seed)

    # Create a list to store the number of tests per person
    # required in each of our simulations
    n_tests = []

    for i in range(n):
        # Simulate the five people we're pooling; each will be
        # drawn from a Bernoulli random variable with probability
        # equal to the proportion of the population that is
        # COVID +ive
        s = np.random.binomial(n=1, p=p, size=5)

        if max(s) == 1:
            # If max(s) is 1, then at least once person is +ve;
            # the outcome of the test will be a bernoulli RV
            # with p=0.9 (the sensitivity of the test)
            T = np.random.binomial(n=1, p=0.9)
        else:
            # If max(s) is 0, then everyone in the pool is
            # negative
            T = np.random.binomial(n=1, p=0.02)

        if T == 0:
            # If the test was negative, it only takes one test
            # to test someone
            n_tests.append(0.2)
        else:
            # If the test was positive, we need to test every
            # person again, so it takes 1.2 tests per person
            n_tests.append(1.2)

    return [np.mean(n_tests), np.std(n_tests)]
```

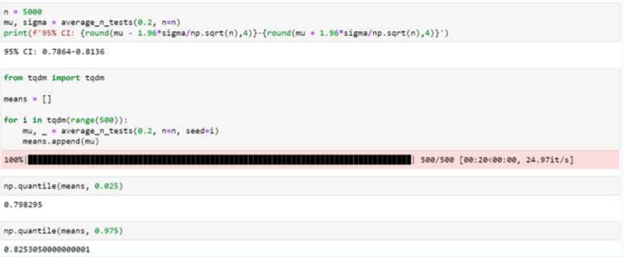
Simulation accuracy



```
n = 5000
mu, sigma = average_n_tests(0.2, n=n)
print(f'95% CI: {round(mu - 1.96*sigma/np.sqrt(n),4)}-{round(mu + 1.96*sigma/np.sqrt(n),4)}')
```

```
95% CI: 0.7864-0.8136
```

Verifying the central limit theorem



```
n = 5000
mu, sigma = average_n_tests(0.2, n=n)
print(f'95% CI: {round(mu - 1.96*sigma/np.sqrt(n),4)}-{round(mu + 1.96*sigma/np.sqrt(n),4)}')
```

```
95% CI: 0.7864-0.8136
```

```
from tqdm import tqdm

means = []

for i in tqdm(range(500)):
    mu, _ = average_n_tests(0.2, n=n, seed=i)
    means.append(mu)
```

```
100%|████████████████████████████████████████████████████████| 500/500 [00:20<00:00, 24.97it/s]
```

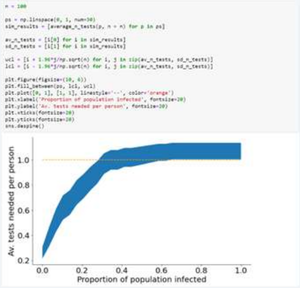
```
np.quantile(means, 0.025)
```

```
0.798295
```

```
np.quantile(means, 0.975)
```

```
0.8253050000000001
```

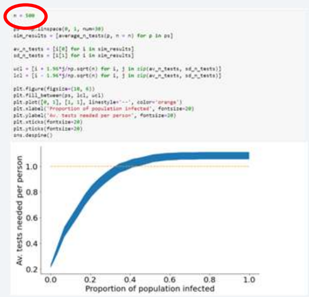
Simulation accuracy



```
n = 100

ps = np.linspace(0, 1, num=30)
sim_results = [average_n_tests(p, n = n) for p in ps]

av_n_tests = [i[0] for i in sim_results]
sd_n_tests = [i[1] for i in sim_results]

ucl = [i + 1.96*j/np.sqrt(n) for i, j in zip(av_n_tests, sd_n_tests)]
lcl = [i - 1.96*j/np.sqrt(n) for i, j in zip(av_n_tests, sd_n_tests)]

plt.figure(figsize=(10, 6))
plt.fill_between(ps, lcl, ucl)
plt.plot([0, 1], [1, 1], linestyle='--', color='orange')
plt.xlabel('Proportion of population infected', fontsize=20)
plt.ylabel('Av. tests needed per person', fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
sns.despine()
```

Av. tests needed per person
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0
Proportion of population infected

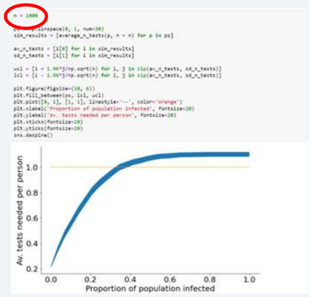
Simulation accuracy



```
n = 500

ps = np.linspace(0, 1, num=30)
sim_results = [average_n_tests(p, n = n) for p in ps]

av_n_tests = [i[0] for i in sim_results]
sd_n_tests = [i[1] for i in sim_results]

ucl = [i + 1.96*j/np.sqrt(n) for i, j in zip(av_n_tests, sd_n_tests)]
lcl = [i - 1.96*j/np.sqrt(n) for i, j in zip(av_n_tests, sd_n_tests)]

plt.figure(figsize=(10, 6))
plt.fill_between(ps, lcl, ucl)
plt.plot([0, 1], [1, 1], linestyle='--', color='orange')
plt.xlabel('Proportion of population infected', fontsize=20)
plt.ylabel('Av. tests needed per person', fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
sns.despine()
```

Av. tests needed per person
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0
Proportion of population infected

Simulation accuracy



```
n = 1000

ps = np.linspace(0, 1, num=30)
sim_results = [average_n_tests(p, n = n) for p in ps]

av_n_tests = [i[0] for i in sim_results]
sd_n_tests = [i[1] for i in sim_results]

ucl = [i + 1.96*j/np.sqrt(n) for i, j in zip(av_n_tests, sd_n_tests)]
lcl = [i - 1.96*j/np.sqrt(n) for i, j in zip(av_n_tests, sd_n_tests)]

plt.figure(figsize=(10, 6))
plt.fill_between(ps, lcl, ucl)
plt.plot([0, 1], [1, 1], linestyle='--', color='orange')
plt.xlabel('Proportion of population infected', fontsize=20)
plt.ylabel('Av. tests needed per person', fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
sns.despine()
```

Av. tests needed per person
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0
Proportion of population infected

Columbia Business School
AT THE VERY CENTER OF BUSINESS

Valuing the healthcare pension liability at GM

Valuing the healthcare pension liability at GM

- The UAW (United Auto Workers) union and GM are in negotiations
- The transfer of the healthcare liability from GM to the union for a fixed amount is being discussed

How should this liability be valued?

Pro-forma analysis

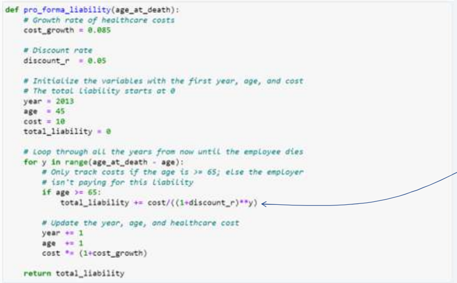
Data on the average employee

- Male
- Age: 45 years
- Age at retirement: 65 years
- Age at death: 78 years

Healthcare costs

- Current year: $10,000
- Annual increase in healthcare costs: 8.5%
- Discount rate assumption: 5%

Pro-forma analysis

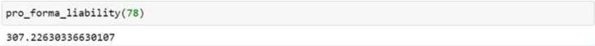


```
def pro_forma_liability(age_at_death):
    # Growth rate of healthcare costs
    cost_growth = 0.085

    # Discount rate
    discount_r  = 0.05

    # Initialize the variables with the first year, age, and cost
    # The total liability starts at 0
    year = 2013
    age  = 45
    cost = 10
    total_liability = 0

    # Loop through all the years from now until the employee dies
    for y in range(age_at_death - age):
        # Only track costs if the age is >= 65; else the employer
        # isn't paying for this liability
        if age >= 65:
            total_liability += cost/((1+discount_r)**y)

        # Update the year, age, and healthcare cost
        year += 1
        age  += 1
        cost *= (1+cost_growth)

    return total_liability
```

$$\frac{\text{cost}}{(1+\text{discount rate})^{\text{years}}}$$

Pro-forma analysis

The average employee dies at 78:

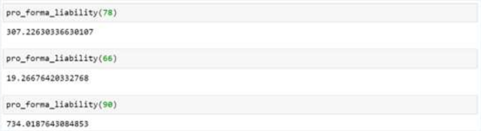


```
pro_forma_liability(78)
307.22630336630107
```

So the net present value of the liability is $307,000

Should we conclude that the UAW should settle for a $307K payment per worker from GM to take the liability off their books?

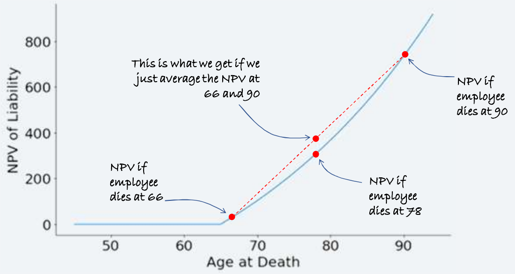
The impact of randomness



```
pro_forma_liability(78)
307.22630336630107

pro_forma_liability(66)
19.26676420332768

pro_forma_liability(90)
734.0187643084853
```

- Suppose employees die at 66 with probability ½ and at 90 with probability ½ (expected age of death: 78)
- The expected NPV is (½×$19.27K) + (½×$734.02K) = $377K
- This is much larger than the $307K obtained from just assuming an age of 78

The impact of randomness



Jensen's Inequality

- This is the result of a more general result called **Jensen's Inequality**
- Given a random variable X and a convex function f, Jensen's Inequality states that

$$E[f(X)] \geq f(E[X])$$

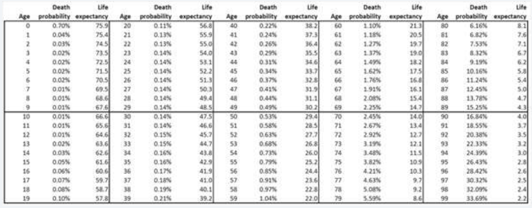
- This requires understanding the concept of a convex function, which we won't cover in this class
- In this instance, X is the age of death (which is random) and f is `pro_forma_liability`

Because of Jensen's Inequality, we need to take randomness into account when calculating the expected value of a function

Modelling the randomness in the death age – actuarial life tables

Actuarial life table: male

Actuarial life tables reports the remaining life expectancy at any specific age:

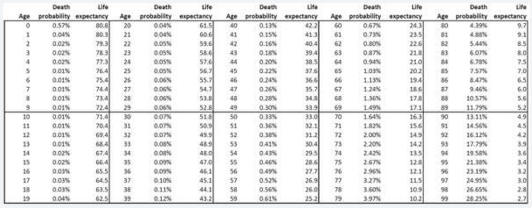


Age	Death probability	Life expectancy	Age	Death probability	Life expectancy	Age	Death probability	Life expectancy	Age	Death probability	Life expectancy	Age	Death probability	Life expectancy
0	0.70%	75.9	20	0.11%	56.8	40	0.22%	38.2	60	1.10%	21.3	80	6.16%	8.1
1	0.04%	75.4	21	0.13%	55.9	41	0.24%	37.3	61	1.18%	20.5	81	6.82%	7.6
2	0.03%	74.5	22	0.13%	55.0	42	0.26%	36.4	62	1.27%	19.7	82	7.53%	7.1
3	0.02%	73.5	23	0.14%	54.0	43	0.29%	35.5	63	1.37%	19.0	83	8.32%	6.7
4	0.02%	72.5	24	0.14%	53.1	44	0.31%	34.6	64	1.49%	18.2	84	9.19%	6.2
5	0.02%	71.5	25	0.14%	52.2	45	0.34%	33.7	65	1.62%	17.5	85	10.16%	5.8
6	0.02%	70.5	26	0.14%	51.3	46	0.37%	32.8	66	1.76%	16.8	86	11.24%	5.4
7	0.01%	69.5	27	0.14%	50.3	47	0.41%	31.9	67	1.91%	16.1	87	12.45%	5.0
8	0.01%	68.6	28	0.14%	49.4	48	0.44%	31.1	68	2.08%	15.4	88	13.78%	4.7
9	0.01%	67.6	29	0.14%	48.5	49	0.49%	30.2	69	2.25%	14.7	89	15.25%	4.3
10	0.01%	66.6	30	0.14%	47.5	50	0.53%	29.4	70	2.45%	14.0	90	16.84%	4.0
11	0.01%	65.6	31	0.14%	46.6	51	0.58%	28.5	71	2.67%	13.4	91	18.55%	3.7
12	0.01%	64.6	32	0.15%	45.7	52	0.63%	27.7	72	2.92%	12.7	92	20.38%	3.5
13	0.02%	63.6	33	0.15%	44.7	53	0.68%	26.8	73	3.19%	12.1	93	22.33%	3.2
14	0.03%	62.6	34	0.16%	43.8	54	0.73%	26.0	74	3.48%	11.5	94	24.39%	3.0
15	0.05%	61.6	35	0.16%	42.9	55	0.79%	25.2	75	3.82%	10.9	95	26.43%	2.8
16	0.06%	60.6	36	0.17%	41.9	56	0.85%	24.4	76	4.21%	10.3	96	28.42%	2.6
17	0.07%	59.7	37	0.18%	41.0	57	0.91%	23.6	77	4.63%	9.7	97	30.32%	2.5
18	0.08%	58.7	38	0.19%	40.1	58	0.97%	22.8	78	5.08%	9.2	98	32.09%	2.4
19	0.10%	57.8	39	0.21%	39.2	59	1.04%	22.0	79	5.59%	8.6	99	33.69%	2.2

https://www.ssa.gov/oact/STATS/table4c6.html

Actuarial life table: female

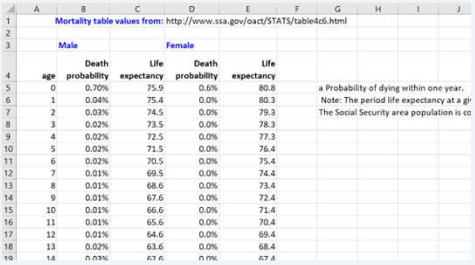
Actuarial life tables reports the remaining life expectancy at any specific age:



Age	Death probability	Life expectancy	Age	Death probability	Life expectancy	Age	Death probability	Life expectancy	Age	Death probability	Life expectancy	Age	Death probability	Life expectancy
0	0.57%	80.8	20	0.04%	61.5	40	0.13%	42.2	60	0.67%	24.3	80	4.39%	9.7
1	0.04%	80.3	21	0.04%	60.6	41	0.15%	41.3	61	0.73%	23.5	81	4.88%	9.1
2	0.02%	79.3	22	0.05%	59.6	42	0.16%	40.4	62	0.80%	22.6	82	5.44%	8.5
3	0.02%	78.3	23	0.05%	58.6	43	0.18%	39.4	63	0.87%	21.8	83	6.07%	8.0
4	0.02%	77.3	24	0.05%	57.6	44	0.20%	38.5	64	0.94%	21.0	84	6.78%	7.5
5	0.01%	76.4	25	0.05%	56.7	45	0.22%	37.6	65	1.03%	20.2	85	7.57%	7.0
6	0.01%	75.4	26	0.06%	55.7	46	0.24%	36.6	66	1.13%	19.4	86	8.47%	6.5
7	0.01%	74.4	27	0.06%	54.7	47	0.26%	35.7	67	1.24%	18.6	87	9.46%	6.0
8	0.01%	73.4	28	0.06%	53.8	48	0.28%	34.8	68	1.36%	17.8	88	10.57%	5.6
9	0.01%	72.4	29	0.06%	52.8	49	0.30%	33.9	69	1.49%	17.1	89	11.79%	5.2
10	0.01%	71.4	30	0.07%	51.8	50	0.33%	33.0	70	1.64%	16.3	90	13.11%	4.9
11	0.01%	70.4	31	0.07%	50.9	51	0.36%	32.1	71	1.82%	15.6	91	14.56%	4.5
12	0.01%	69.4	32	0.07%	49.9	52	0.38%	31.2	72	2.00%	14.9	92	16.12%	4.2
13	0.01%	68.4	33	0.08%	48.9	53	0.41%	30.4	73	2.20%	14.2	93	17.79%	3.9
14	0.02%	67.4	34	0.08%	48.0	54	0.43%	29.5	74	2.42%	13.5	94	19.58%	3.6
15	0.02%	66.4	35	0.09%	47.0	55	0.46%	28.6	75	2.67%	12.8	95	21.38%	3.4
16	0.03%	65.5	36	0.09%	46.1	56	0.49%	27.7	76	2.96%	12.1	96	23.19%	3.2
17	0.03%	64.5	37	0.10%	45.1	57	0.52%	26.9	77	3.27%	11.5	97	24.95%	3.0
18	0.03%	63.5	38	0.11%	44.1	58	0.56%	26.0	78	3.60%	10.9	98	26.65%	2.8
19	0.04%	62.5	39	0.12%	43.2	59	0.61%	25.2	79	3.97%	10.2	99	28.25%	2.7

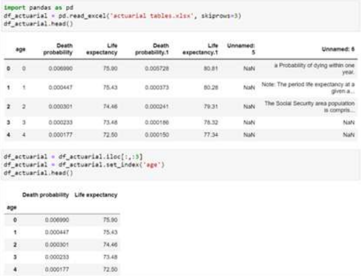
https://www.ssa.gov/oact/STATS/table4c6.html

Actuarial life tables



	A	B	C	D	E	F	G	H	I	J
1			Mortality table values from:	http://www.ssa.gov/oact/STATS/table4c6.html						
2										
3		Male		Female						
4	age	Death probability	Life expectancy	Death probability	Life expectancy					
5	0	0.70%	75.9	0.6%	80.8		a Probability of dying within one year.			
6	1	0.04%	75.4	0.0%	80.3		Note: The period life expectancy at a gi			
7	2	0.03%	74.5	0.0%	79.3		The Social Security area population is co			
8	3	0.02%	73.5	0.0%	78.3					
9	4	0.02%	72.5	0.0%	77.3					
10	5	0.02%	71.5	0.0%	76.4					
11	6	0.02%	70.5	0.0%	75.4					
12	7	0.01%	69.5	0.0%	74.4					
13	8	0.01%	68.6	0.0%	73.4					
14	9	0.01%	67.6	0.0%	72.4					
15	10	0.01%	66.6	0.0%	71.4					
16	11	0.01%	65.6	0.0%	70.4					
17	12	0.01%	64.6	0.0%	69.4					
18	13	0.02%	63.6	0.0%	68.4					
19	14	0.03%	62.6	0.0%	67.4					

Actuarial life tables



```
import pandas as pd
df_actuarial = pd.read_excel('actuarial tables.xlsx', skiprows=3)
df_actuarial.head()
```

	age	Death probability	Life expectancy	Death probability.1	Life expectancy.1	Unnamed: 5	Unnamed: 6
0	0	0.006990	75.90	0.005728	80.81	NaN	a Probability of dying within one year.
1	1	0.000447	75.43	0.000373	80.28	NaN	Note: The period life expectancy at a given a...
2	2	0.000301	74.46	0.000241	79.31	NaN	The Social Security area population is compris...
3	3	0.000233	73.48	0.000186	78.32	NaN	NaN
4	4	0.000177	72.50	0.000150	77.34	NaN	NaN

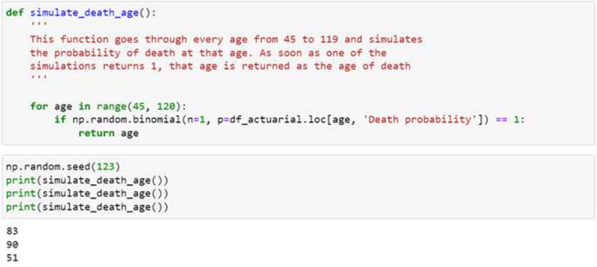
```
df_actuarial = df_actuarial.iloc[:,:3]
df_actuarial = df_actuarial.set_index('age')
df_actuarial.head()
```

	Death probability	Life expectancy
age		
0	0.006990	75.90
1	0.000447	75.43
2	0.000301	74.46
3	0.000233	73.48
4	0.000177	72.50

Simulating age of death

- For every year, generate a Bernoulli random variable with *p* equal to the probability of death at that age
- If the variable is 1, the person dies at that age. If it is 0, the person doesn't
- The age of death is the *minimum* age with an indicator of 1

Simulating age of death



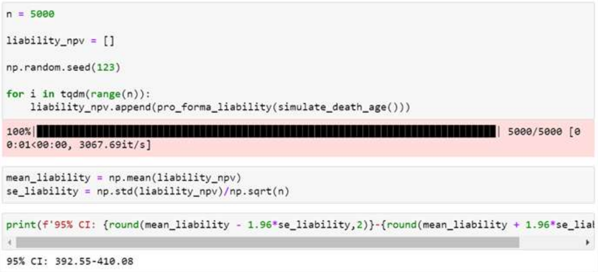
```
def simulate_death_age():
    '''
    This function goes through every age from 45 to 119 and simulates
    the probability of death at that age. As soon as one of the
    simulations returns 1, that age is returned as the age of death
    '''

    for age in range(45, 120):
        if np.random.binomial(n=1, p=df_actuarial.loc[age, 'Death probability']) == 1:
            return age
```

```
np.random.seed(123)
print(simulate_death_age())
print(simulate_death_age())
print(simulate_death_age())
```

```
83
90
51
```

Back to the GM case

Back to the GM case



```
n = 5000

liability_npv = []

np.random.seed(123)

for i in tqdm(range(n)):
    liability_npv.append(pro_forma_liability(simulate_death_age()))
```

```
100%|██████████████████████████████████████████████████| 5000/5000 [0
0:01<00:00, 3067.69it/s]
```

```
mean_liability = np.mean(liability_npv)
se_liability = np.std(liability_npv)/np.sqrt(n)
```

```
print(f'95% CI: {round(mean_liability - 1.96*se_liability,2)}-{round(mean_liability + 1.96*se_lia
```

```
95% CI: 392.55-410.08
```

Correctly incorporating randomness shows us the liability was not $307K (pro-forma assuming death age of 78) but $401K

Estimating probabilities

Estimating probabilities

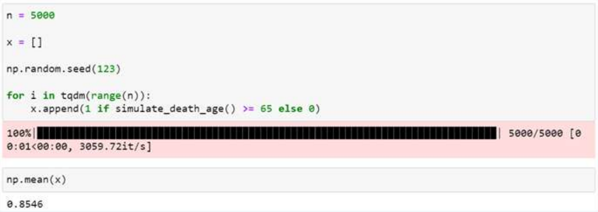
- Monte Carlo simulation can also be used to estimate the probability of an event
- Suppose we want to estimate the probability the age of death is $\geq$ 65
- Define

$$X = \begin{cases} 1 & \text{if age of death } \geq 65 \\ 0 & \text{otherwise} \end{cases}$$

- Then

$$E[X] = 0 \times P(X=0) + 1 \times P(X=1) = P(\text{age of death} \geq 65)$$

Estimating probabilities



```
n = 5000

x = []

np.random.seed(123)

for i in tqdm(range(n)):
    x.append(1 if simulate_death_age() >= 65 else 0)
```

```
100%|██████████████████████████████████████████████████████████| 5000/5000 [0
0:01<00:00, 3059.72it/s]
```

```
np.mean(x)
```

```
0.8546
```

Simulation applications

Traffic simulation

Traffic simulation

- Traffic light timing
- One-way versus two-way streets
- Impact of road closures

Simulation of epidemics

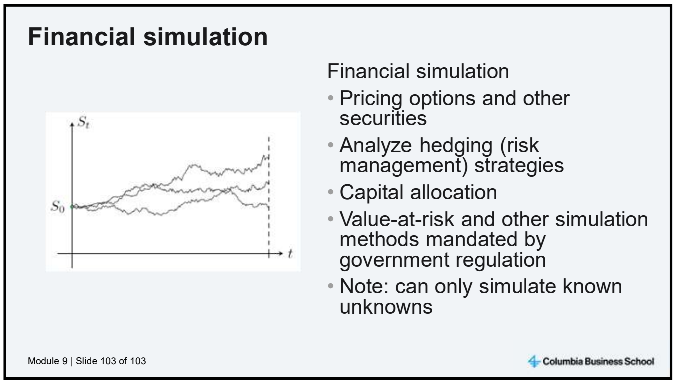
Simulation of epidemics

- Spread through air travel
- Analyze potential interventions
- Analyze vaccination policies

Call center simulation

Call center simulation

- Random arrival times of calls
- Analyze impact of staffing plans on key performance metrics



Financial simulation

Financial simulation

- Pricing options and other securities
- Analyze hedging (risk management) strategies
- Capital allocation
- Value-at-risk and other simulation methods mandated by government regulation
- Note: can only simulate known unknowns

Fall 2024

Prescriptive Analytics: Testing Channel Management in Retail

Module 10

Professor Daniel Guetta

This Module

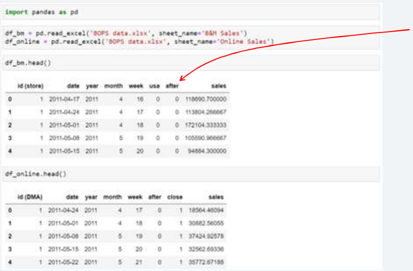
- Evaluating the Buy Online Pickup in Store (BOPS) program at *Home and Kitchen*
  - Analyzing the impact
  - Prescription (keep or drop)

  Source: "*Integration of Online and Offline Channels in Retail: The Impact of Sharing Reliable Inventory Availability Information*", 2014. Gallino, S., Moreno, A. *Management Science*
- Difference in Differences (DiD) method
- Evaluating the impact of Search Engine Marketing at eBay

Buy online pick up in store (BOPS) at Home & Kitchen

What is BOPS?

Data from the original pilot



```
import pandas as pd
```

```
df_bm = pd.read_excel('BOPS data.xlsx', sheet_name='B&M Sales')
df_online = pd.read_excel('BOPS data.xlsx', sheet_name='Online Sales')
```

```
df_bm.head()
```

```
   id (store)        date  year  month  week  usa  after          sales
0           1  2011-04-17  2011      4    16    0      0  118690.700000
1           1  2011-04-24  2011      4    17    0      0  113804.266667
2           1  2011-05-01  2011      4    18    0      0  172104.333333
3           1  2011-05-08  2011      5    19    0      0  105590.966667
4           1  2011-05-15  2011      5    20    0      0   94884.300000
```

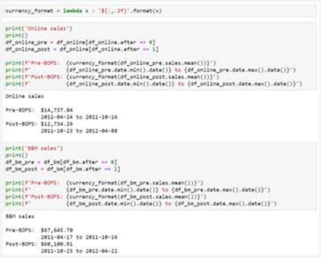
```
df_online.head()
```

```
   id (DMA)        date  year  month  week  after  close        sales
0         1  2011-04-24  2011      4    17      0      1  18564.46094
1         1  2011-05-01  2011      4    18      0      1  30682.56055
2         1  2011-05-08  2011      5    19      0      1  37424.92578
3         1  2011-05-15  2011      5    20      0      1  32562.69336
4         1  2011-05-22  2011      5    21      0      1  35772.67188
```

Was this week after the introduction of BOPS (1) or before (0)

Was the BOPS pilot a success? Should Home & Kitchen deploy it to Canada?

Initial analysis



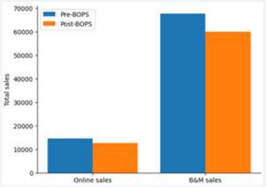
```
currency_format = lambda x : '${:,.2f}'.format(x)
```

```
print('Online sales')
print()
df_online_pre = df_online[df_online.after == 0]
df_online_post = df_online[df_online.after == 1]

print(f'Pre-BOPS:  {currency_format(df_online_pre.sales.mean())}')
print(f'           {df_online_pre.date.min().date()} to {df_online_pre.date.max().date()}')
print(f'Post-BOPS: {currency_format(df_online_post.sales.mean())}')
print(f'           {df_online_post.date.min().date()} to {df_online_post.date.max().date()}')
```

```
Online sales

Pre-BOPS:  $14,737.84
           2011-04-24 to 2011-10-16
Post-BOPS: $12,734.29
           2011-10-23 to 2012-04-08
```

```
print('B&M sales')
print()
df_bm_pre = df_bm[df_bm.after == 0]
df_bm_post = df_bm[df_bm.after == 1]

print(f'Pre-BOPS:  {currency_format(df_bm_pre.sales.mean())}')
print(f'           {df_bm_pre.date.min().date()} to {df_bm_pre.date.max().date()}')
print(f'Post-BOPS: {currency_format(df_bm_post.sales.mean())}')
print(f'           {df_bm_post.date.min().date()} to {df_bm_post.date.max().date()}')
```

```
B&M sales

Pre-BOPS:  $67,645.70
           2011-04-17 to 2011-10-16
Post-BOPS: $60,100.91
           2011-10-23 to 2012-04-22
```



What do we conclude? What might we be missing?

Factors we might be missing

Other factors might be causing the disparity between the "before" and "after" period

- Seasonality (holidays, back-to-school, summer moving season)
- Macro-economic factors (growth, shocks)
- Systemic company-wide factors (product selection, marketing)
- Systemic competitive factors (entrance of a new competitor)

How can we isolate the effect of BOPS from all these other confounding factors

The Differences in Differences (DiD) approach

Isolating the impact of BOPS

General idea of the difference in differences (DiD) approach

- Identify a control group
  - Similar to the test group and subject to the same common factors
  - **Not** exposed to the treatment
- Compare the **change in outcome** in the control group to the **change in outcome** in the test group

Example of DiD

A recent study at Columbia College reports that freshmen who participate in club sports gain an average of 3.6 pounds during their first year of college

Student group	Average starting weight	Average ending weight
Club sports	144.3	147.9
No club sports		

Does participating in sports cause weight gain?

Example of DiD

A recent study at Columbia College reports that freshmen who participate in club sports gain an average of 3.6 pounds during their first year of college

Student group	Average starting weight	Average ending weight	Difference
Club sports	144.3	147.9	3.6
No club sports	149.2	156.3	7.1
		DiD	–3.5

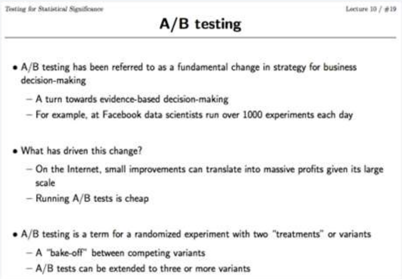
What fundamental assumption are we making in this analysis?

Testing

Decision/treatment → outcome

- How can we quantify the impact of a treatment?
- Looking at the outcome alone ignores *confounding factors*
- Testing seeks to isolate the treatment’s *causal effect*

A/B testing

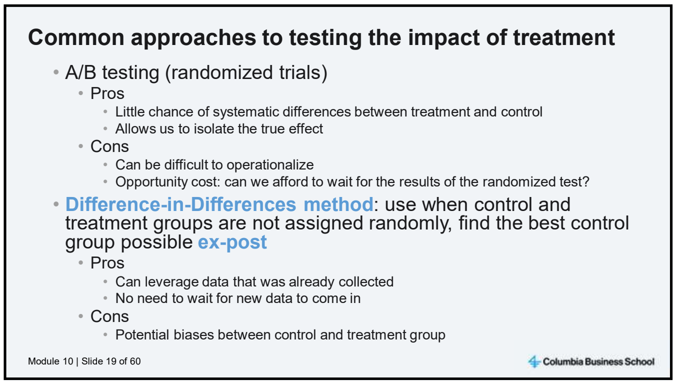


Testing for Statistical Significance Lecture 10 / #19

A/B testing

- A/B testing has been referred to as a fundamental change in strategy for business decision-making
  - A turn towards evidence-based decision-making
  - For example, at Facebook data scientists run over 1000 experiments each day
- What has driven this change?
  - On the Internet, small improvements can translate into massive profits given its large scale
  - Running A/B tests is cheap
- A/B testing is a term for a randomized experiment with two "treatments" or variants
  - A "bake-off" between competing variants
  - A/B tests can be extended to three or more variants

What would A/B testing have looked like for BOPS?

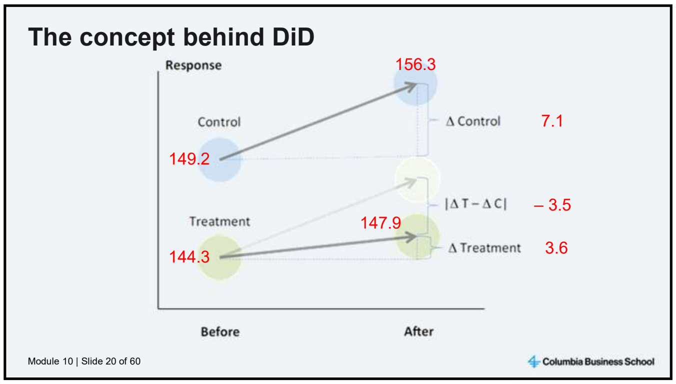


Common approaches to testing the impact of treatment

- A/B testing (randomized trials)
  - Pros
    - Little chance of systematic differences between treatment and control
    - Allows us to isolate the true effect
  - Cons
    - Can be difficult to operationalize
    - Opportunity cost: can we afford to wait for the results of the randomized test?
- **Difference-in-Differences method**: use when control and treatment groups are not assigned randomly, find the best control group possible **ex-post**
  - Pros
    - Can leverage data that was already collected
    - No need to wait for new data to come in
  - Cons
    - Potential biases between control and treatment group

Module 10 | Slide 19 of 60

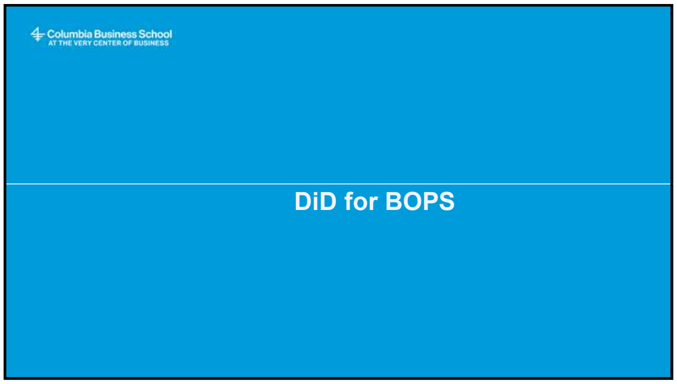
Columbia Business School



The concept behind DiD

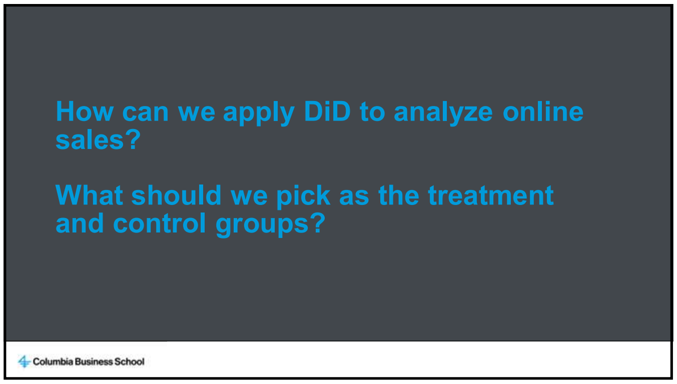
Module 10 | Slide 20 of 60

Columbia Business School



Columbia Business School
AT THE VERY CENTER OF BUSINESS

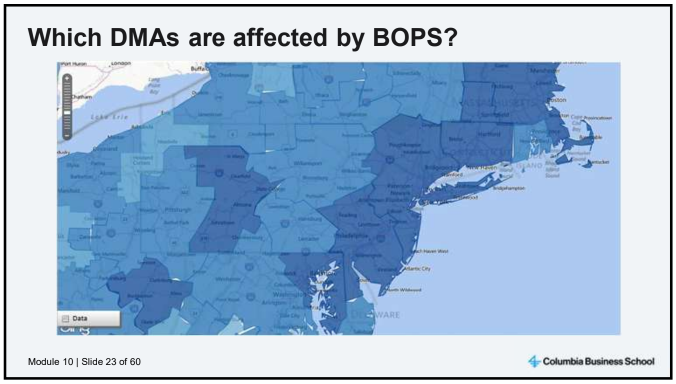
DiD for BOPS



How can we apply DiD to analyze online sales?

What should we pick as the treatment and control groups?

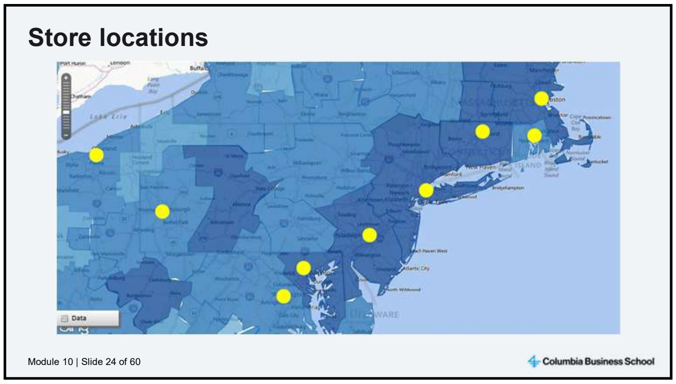
Columbia Business School



Which DMAs are affected by BOPS?

Module 10 | Slide 23 of 60

Columbia Business School

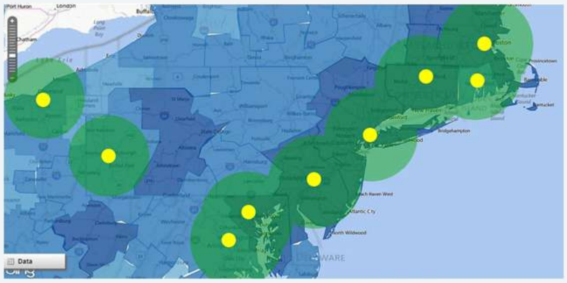


Store locations

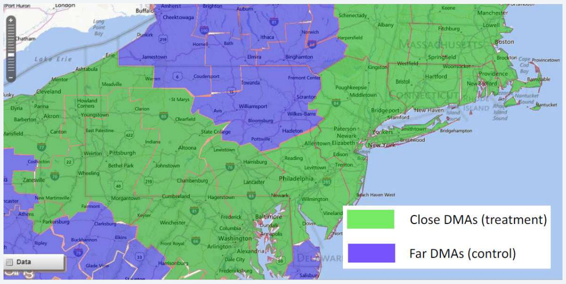
Module 10 | Slide 24 of 60

Columbia Business School

50 mile radius from stores



“Close” and “Far” DMAs

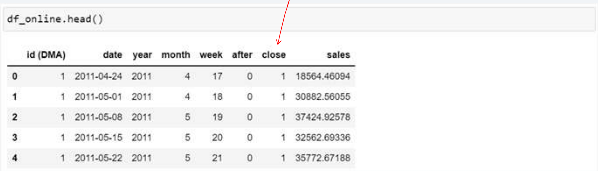


🟩 Close DMAs (treatment)

🟪 Far DMAs (control)

DiD for online sales analysis

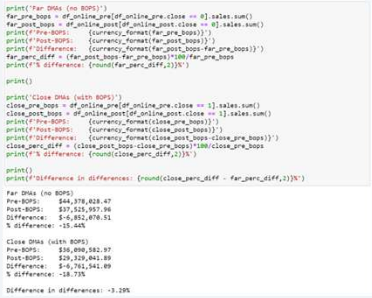
1 if this DMA was close to a home & kitchen store, 0 if it was far from a store



```
df_online.head()
```

	id (DMA)	date	year	month	week	after	close	sales
0	1	2011-04-24	2011	4	17	0	1	18564.46094
1	1	2011-05-01	2011	4	18	0	1	30882.56055
2	1	2011-05-08	2011	5	19	0	1	37424.92578
3	1	2011-05-15	2011	5	20	0	1	32562.69336
4	1	2011-05-22	2011	5	21	0	1	35772.67188

DiD for online sales analysis



```
print('Far DMAs (no BOPS)')
far_pre_bops = df_online_pre[df_online_pre.close == 0].sales.sum()
far_post_bops = df_online_post[df_online_post.close == 0].sales.sum()
print(f'Pre-BOPS:      {currency_format(far_pre_bops)}')
print(f'Post-BOPS:     {currency_format(far_post_bops)}')
print(f'Difference:    {currency_format(far_post_bops-far_pre_bops)}')
far_perc_diff = (far_post_bops-far_pre_bops)*100/far_pre_bops
print(f'% difference: {round(far_perc_diff,2)}%')

print()

print('Close DMAs (with BOPS)')
close_pre_bops = df_online_pre[df_online_pre.close == 1].sales.sum()
close_post_bops = df_online_post[df_online_post.close == 1].sales.sum()
print(f'Pre-BOPS:      {currency_format(close_pre_bops)}')
print(f'Post-BOPS:     {currency_format(close_post_bops)}')
print(f'Difference:    {currency_format(close_post_bops-close_pre_bops)}')
close_perc_diff = (close_post_bops-close_pre_bops)*100/close_pre_bops
print(f'% difference: {round(close_perc_diff,2)}%')

print()
print(f'Difference in differences: {round(close_perc_diff - far_perc_diff,2)}%')
```

```
Far DMAs (no BOPS)
Pre-BOPS:      $44,378,028.47
Post-BOPS:     $37,525,957.96
Difference:    $-6,852,070.51
% difference: -15.44%

Close DMAs (with BOPS)
Pre-BOPS:      $36,090,582.97
Post-BOPS:     $29,329,041.89
Difference:    $-6,761,541.09
% difference: -18.73%

Difference in differences: -3.29%
```

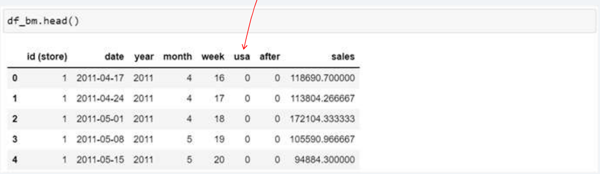
What are some potential caveats of this analysis?

Assuming it’s correct, what do we conclude? Any thoughts?

How could we apply DiD to brick & mortar sales – what’s our control group?

DiD for B&M sales analysis

1 if this store was in the USA, 0 if it was in Canada

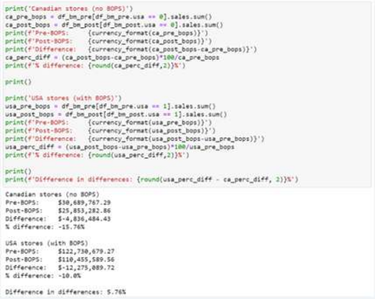


```
df_bm.head()
```

	id (store)	date	year	month	week	usa	after	sales
0	1	2011-04-17	2011	4	16	0	0	118690.700000
1	1	2011-04-24	2011	4	17	0	0	113804.266667
2	1	2011-05-01	2011	4	18	0	0	172104.333333
3	1	2011-05-08	2011	5	19	0	0	105590.966667
4	1	2011-05-15	2011	5	20	0	0	94884.300000

Columbia Business School

DiD for B&M sales analysis



```
print('Canadian stores (no BOPS)')
ca_pre_bops = df_bm_pre[df_bm_pre.usa == 0].sales.sum()
ca_post_bops = df_bm_post[df_bm_post.usa == 0].sales.sum()
print(f'Pre-BOPS:     {currency_format(ca_pre_bops)}')
print(f'Post-BOPS:    {currency_format(ca_post_bops)}')
print(f'Difference:   {currency_format(ca_post_bops-ca_pre_bops)}')
ca_perc_diff = (ca_post_bops-ca_pre_bops)*100/ca_pre_bops
print(f'% difference: {round(ca_perc_diff,2)}%')

print()

print('USA stores (with BOPS)')
usa_pre_bops = df_bm_pre[df_bm_pre.usa == 1].sales.sum()
usa_post_bops = df_bm_post[df_bm_post.usa == 1].sales.sum()
print(f'Pre-BOPS:     {currency_format(usa_pre_bops)}')
print(f'Post-BOPS:    {currency_format(usa_post_bops)}')
print(f'Difference:   {currency_format(usa_post_bops-usa_pre_bops)}')
usa_perc_diff = (usa_post_bops-usa_pre_bops)*100/usa_pre_bops
print(f'% difference: {round(usa_perc_diff,2)}%')

print()
print(f'Difference in differences: {round(usa_perc_diff - ca_perc_diff, 2)}%')
```

```
Canadian stores (no BOPS)
Pre-BOPS:     $30,689,767.29
Post-BOPS:    $25,853,282.86
Difference:   $-4,836,484.43
% difference: -15.76%

USA stores (with BOPS)
Pre-BOPS:     $122,730,679.27
Post-BOPS:    $110,455,589.56
Difference:   $-12,275,089.72
% difference: -10.0%

Difference in differences: 5.76%
```

Columbia Business School

What are some potential caveats of this analysis?

Assuming it's correct, what do we conclude? Any thoughts?

Columbia Business School

Aggregate impact of BOPS on sales

Estimated impact of BOPS on *Home and Kitchen* sales

- Online sales affected by BOPS: $36.1M × –3.3% = $ –1.2M
- B&M sales affected by BOPS: $123M × 5.8% = $7.1M

Estimated aggregate impact on sales

- $7.1M – $1.2M = $5.9M
- 2.9% increase in company-wide revenues

Columbia Business School

Should *Home and Kitchen* drop the BOPS initiative, or move ahead and deploy BOPS to Canada?

Columbia Business School

Columbia Business School
AT THE VERY CENTER OF BUSINESS

What unit to use?

An alternative approach to DiD

The DiD method computed the impact of the BOPS treatment according to

$$(\%\text{TotalSalesChange})_{\text{TREATED}} - (\%\text{TotalSalesChange})_{\text{CONTROL}}$$

The combines **all the units** (stores or DMAs) in the two groups. It then finds the difference between the two groups.

One shortcoming is that small units (small stores/small DMAs) will be dwarfed by large ones. To resolve this, we could first find the % change in each unit

An example

Store	Sales before	Sales after	USA?	Sales difference	% difference
1	$1M	$1M	0	$0	0%
2	$100M	$100M	0	$0	0%
Total Canada	$101M	$101M		$0	
3	$1M	$2M	1	$1M	100%
4	$100M	$101M	1	$1M	1%
Total USA	$101M	$103M		$2M	

Two ways to calculate these – either by averaging the percentage differences in each store, or by using the aggregates to calculate the percentage difference

An example; aggregated [what we did with BOPS]

Store	Sales before	Sales after	USA?	Sales difference	% difference
1	$1M	$1M	0	$0	0%
2	$100M	$100M	0	$0	0%
Total Canada	$101M	$101M		$0	$0 ÷ $101M = 0%
3	$1M	$2M	1	$1M	100%
4	$100M	$101M	1	$1M	1%
Total USA	$101M	$103M		$2M	$2M ÷ $101M = 1.98%

Each store gets counted proportionally to its size

DiD = 1.98% – 0% = 1.98%

An example;unit-wise

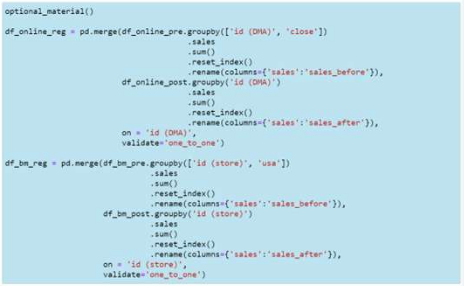
Store	Sales before	Sales after	USA?	Sales difference	% difference
1	$1M	$1M	0	$0	0%
2	$100M	$100M	0	$0	0%
Total Canada	$101M	$101M		$0	(0 + 0) ÷ 2 = 0%
3	$1M	$2M	1	$1M	100%
4	$100M	$101M	1	$1M	1%
Total USA	$101M	$103M		$2M	(100 + 1) ÷ 2 = 50.5%

Each store gets counted equally

DiD = 50.5% – 0% = 50.5%

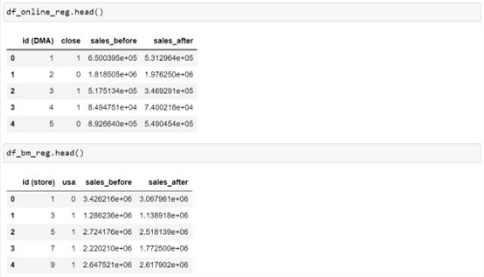
Which method should we use?

Preparing for unit-wise DiD



```
optional_material()

df_online_reg = pd.merge(df_online_pre.groupby(['id (DMA)', 'close'])
                                        .sales
                                        .sum()
                                        .reset_index()
                                        .rename(columns={'sales':'sales_before'}),
                         df_online_post.groupby('id (DMA)')
                                        .sales
                                        .sum()
                                        .reset_index()
                                        .rename(columns={'sales':'sales_after'}),
                         on = 'id (DMA)',
                         validate='one_to_one')

df_bm_reg = pd.merge(df_bm_pre.groupby(['id (store)', 'usa'])
                                .sales
                                .sum()
                                .reset_index()
                                .rename(columns={'sales':'sales_before'}),
                     df_bm_post.groupby('id (store)')
                                .sales
                                .sum()
                                .reset_index()
                                .rename(columns={'sales':'sales_after'}),
                     on = 'id (store)',
                     validate='one_to_one')
```

Preparing for unit-wise DiD



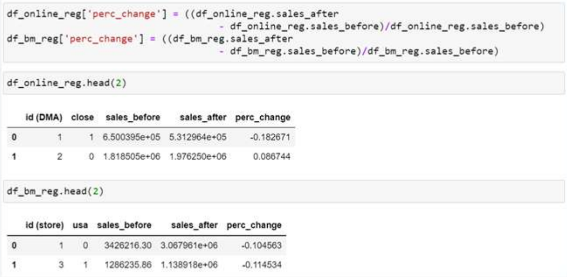
```
df_online_reg.head()
```

	id (DMA)	close	sales_before	sales_after
0	1	1	6.500395e+05	5.312964e+05
1	2	0	1.818505e+06	1.976250e+06
2	3	1	5.175134e+05	3.469291e+05
3	4	1	8.494751e+04	7.400218e+04
4	5	0	8.926640e+05	5.490454e+05

```
df_bm_reg.head()
```

	id (store)	usa	sales_before	sales_after
0	1	0	3.426216e+06	3.067961e+06
1	3	1	1.286236e+06	1.138918e+06
2	5	1	2.724176e+06	2.518139e+06
3	7	1	2.220210e+06	1.772500e+06
4	9	1	2.647521e+06	2.617902e+06

Preparing for unit-wise DiD



```
df_online_reg['perc_change'] = ((df_online_reg.sales_after
                                  - df_online_reg.sales_before)/df_online_reg.sales_before)
df_bm_reg['perc_change'] = ((df_bm_reg.sales_after
                                  - df_bm_reg.sales_before)/df_bm_reg.sales_before)
```

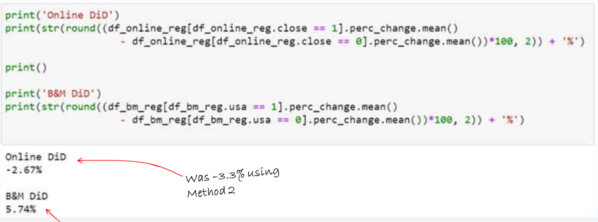
```
df_online_reg.head(2)
```

	id (DMA)	close	sales_before	sales_after	perc_change
0	1	1	6.500395e+05	5.312964e+05	-0.182671
1	2	0	1.818505e+06	1.976250e+06	0.086744

```
df_bm_reg.head(2)
```

	id (store)	usa	sales_before	sales_after	perc_change
0	1	0	3426216.30	3.067961e+06	-0.104563
1	3	1	1286235.86	1.138918e+06	-0.114534

Unit-wise DiD



```
print('Online DiD')
print(str(round((df_online_reg[df_online_reg.close == 1].perc_change.mean()
                 - df_online_reg[df_online_reg.close == 0].perc_change.mean())*100, 2)) + '%')

print()

print('B&M DiD')
print(str(round((df_bm_reg[df_bm_reg.usa == 1].perc_change.mean()
                 - df_bm_reg[df_bm_reg.usa == 0].perc_change.mean())*100, 2)) + '%')
```

```
Online DiD
-2.67%

B&M DiD
5.74%
```

Was -3.3% using Method 2

Was 5.8% using Method 2

Columbia Business School
AT THE VERY CENTER OF BUSINESS

DiD and linear regression

DiD using linear regression

Unit-wise DiD can we done using linear regression

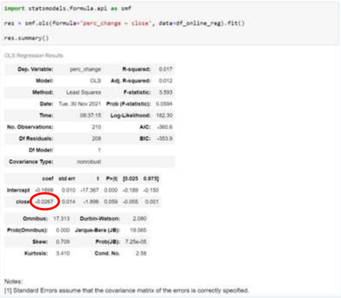
$$\%\text{SalesChange}_i = a + b \times \text{TREATED}_i + \text{error}$$

Each line in the data is one unit (DMA/store/etc…)

- *i*: the index of the unit (DMA or store)
- TREATED$_i$: equal to 1 if unit *i* received the treatment (i.e., if the DMA was close for online sales or the store was in the USA for B&M sales) and 0 otherwise
- *b*: measures the impact of the treatment (BOPS)

Why can we not do this using aggregated DiD?

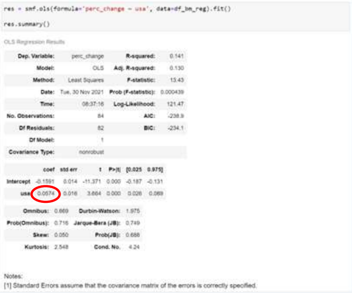
Unit-wise DiD (online sales)



```
import statsmodels.formula.api as smf

res = smf.ols(formula='perc_change ~ close', data=df_online_reg).fit()

res.summary()
```

```
OLS Regression Results
   Dep. Variable:       perc_change        R-squared:    0.017
           Model:               OLS   Adj. R-squared:    0.012
          Method:     Least Squares      F-statistic:    3.593
            Date:  Tue, 30 Nov 2021 Prob (F-statistic):  0.0594
            Time:          08:37:15   Log-Likelihood:   182.30
No. Observations:               210              AIC:   -360.6
    Df Residuals:               208              BIC:   -353.9
        Df Model:                 1
 Covariance Type:         nonrobust

             coef  std err        t  P>|t|  [0.025  0.975]
Intercept  -0.1698   0.010  -17.367  0.000  -0.189  -0.150
    close  -0.0267   0.014   -1.896  0.059  -0.055   0.001

      Omnibus:  17.313   Durbin-Watson:    2.080
Prob(Omnibus):   0.000 Jarque-Bera (JB):  18.065
         Skew:   0.709        Prob(JB):  7.25e-05
     Kurtosis:   3.410        Cond. No.     2.58

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Unit-wise DiD (B&M sales)



```
res = smf.ols(formula='perc_change ~ usa', data=df_bm_reg).fit()

res.summary()
```

```
OLS Regression Results
   Dep. Variable:       perc_change        R-squared:     0.141
           Model:               OLS   Adj. R-squared:     0.130
          Method:     Least Squares      F-statistic:     13.43
            Date:  Tue, 30 Nov 2021 Prob (F-statistic):  0.000439
            Time:          08:37:16   Log-Likelihood:    121.47
No. Observations:                84              AIC:    -238.9
    Df Residuals:                82              BIC:    -234.1
        Df Model:                 1
 Covariance Type:         nonrobust

             coef  std err        t  P>|t|  [0.025  0.975]
Intercept  -0.1591   0.014  -11.371  0.000  -0.187  -0.131
      usa   0.0574   0.016    3.664  0.000   0.026   0.089

      Omnibus:  0.669   Durbin-Watson:    1.975
Prob(Omnibus):  0.716 Jarque-Bera (JB):  0.749
         Skew:  0.050        Prob(JB):  0.688
     Kurtosis:  2.548        Cond. No.    4.24

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

What are some benefits of doing this over just finding the difference as we did before?

Adding explanatory variables

Suppose, in some stores, Home and Kitchen competes with Bed, Bath, and Beyond. This can be accounted for in the regressing

$$\%\text{SalesChange}_i = a + b \times \text{TREATED}_i + c \times \text{BBB}_i + \text{error}$$

Each line in the data is one unit (DMA/store/etc…)

- *i*: the index of the unit (DMA or store)
- TREATED_i: equal to 1 if unit *i* received the treatment (i.e., if the DMA was close for online sales or the store was in the USA for B&M sales) and 0 otherwise
- *b*: measures the impact of the treatment (BOPS)
- Other variables can be added (e.g.: store specific variables, etc…) to correct for confounding variables

DiD can discover causal impact where a basic analysis might have missed it

Columbia Business School
AT THE VERY CENTER OF BUSINESS

Another application: search engine marketing (SEM) at eBay

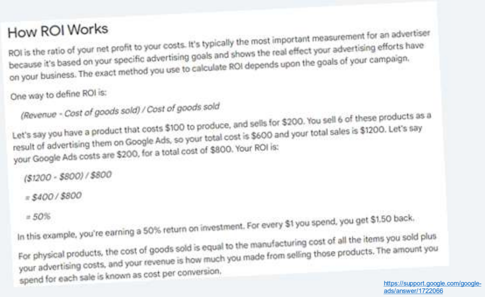
Search engine marketing at eBay

Context: in 2010, eBay spent $4 million per month on “search engine marketing” (SEM) (also known as sponsored search advertising)

Cost of SEM: pay a fee each time a customer clicks on an ad

How to compute the ROI of SEM?

Measuring the impact of advertising: Google’s Advice



How ROI Works

ROI is the ratio of your net profit to your costs. It's typically the most important measurement for an advertiser because it's based on your specific advertising goals and shows the real effect your advertising efforts have on your business. The exact method you use to calculate ROI depends upon the goals of your campaign.

One way to define ROI is:

(Revenue - Cost of goods sold) / Cost of goods sold

Let's say you have a product that costs $100 to produce, and sells for $200. You sell 6 of these products as a result of advertising them on Google Ads, so your total cost is $600 and your total sales is $1200. Let's say your Google Ads costs are $200, for a total cost of $800. Your ROI is:

($1200 - $800) / $800

= $400 / $800

= 50%

In this example, you're earning a 50% return on investment. For every $1 you spend, you get $1.50 back.

For physical products, the cost of goods sold is equal to the manufacturing cost of all the items you sold plus your advertising costs, and your revenue is how much you made from selling those products. The amount you spend for each sale is known as cost per conversion.

https://support.google.com/google-ads/answer/1722066

What do we think of Google’s advice?

Measuring the impact of SEM at eBay

Experiment (focus on non-branded keywords without the word “eBay”:

- Construct treatment/control groups through DMAs (it’s easy to restrict ads by geographic areas; serve ads only to *some* areas)
- **Treated group**: out of 210 DMAs, randomly select 65 where Google SEM would be turned off from two months
- **Control group**: create a control group of DMAs that match the previous 65 (similar traffic seasonality)

Measuring the impact of SEM at eBay

Estimate the impact of SEM on sales by using difference in differences

(Difference in Sales in treated group)

– (Difference in Sales in control group)

Result:

- SEM increases sales by about 0.44% (not statistically significant)
- **ROI estimate**: –63% (short term estimate)

Source: “*Consumer Heterogeneity and Paid Search Effectiveness: A Large Scale Field Experiment,*” *Econometrica*, 2015. Blake, T., Nosko, C., and Tadelis, S. http://conference.nber.org/confer/2013/EoDs13/Tadelis.pdf