INFO 2950: Intro to Data Science

Lecture 15
2023-10-18

Agenda

1. Bootstrapping
2. Binomial distributions of sequences
    a. Probabilities
    b. Counts
3. Polling: bootstrapping for margins of error

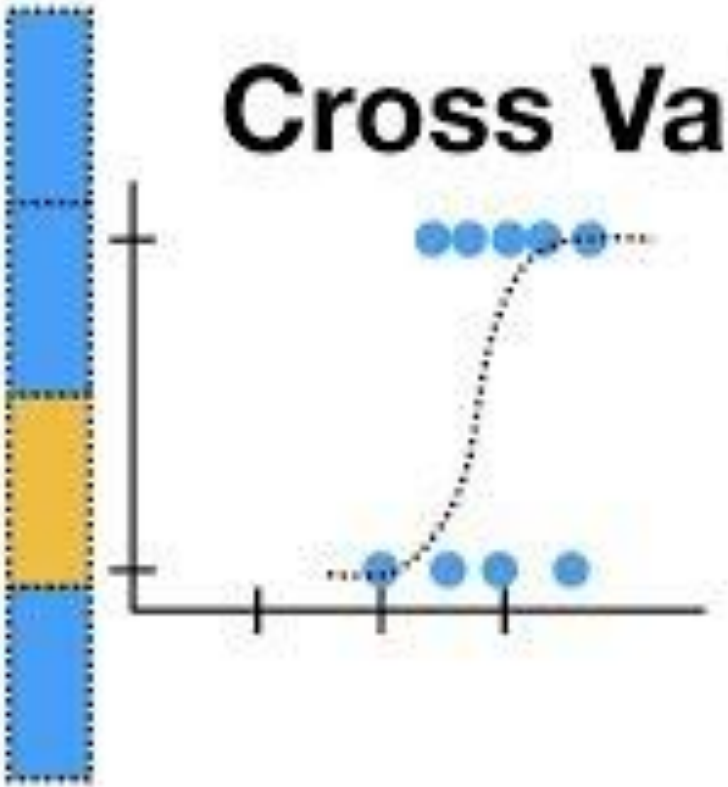
Refresher: Cross Validation

- Cross validation is used to measure generalized model performance (checks for overfitting)
- Can tell you the **variance** of your evaluation metrics across “out-of-sample” folds of data

Refresher: Cross Validation

- Cross validation is used to measure generalized model performance (checks for overfitting)
- Can tell you the **variance** of your evaluation metrics across “out-of-sample” folds of data
  - If our evaluation metrics have *high variance across CV folds*, we might not trust our model to perform consistently well across out-of-sample data

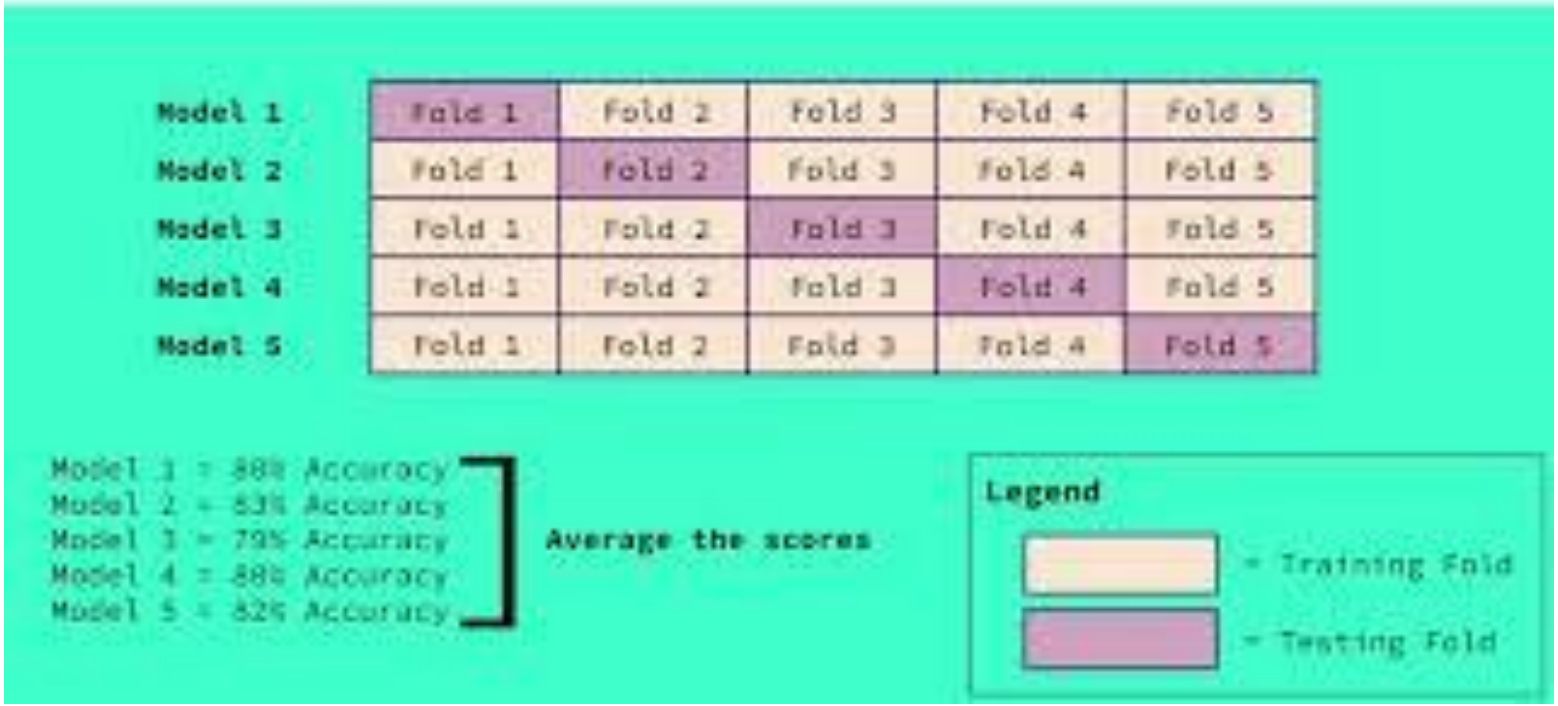
Cross Validation….



…it's no big deal!!!

https://www.youtube.com/watch?v=fSytzGwwBVw

What is Cross Validation?



https://www.youtube.com/watch?v=hoNpvry0370

What a regression model outputs

- α
- β's
- $\hat{y}$
- Evaluation metrics
  - Numeric (MAE, RMSE, etc.)
  - Binary (precision, recall, etc.)

What a regression model outputs

- α
- β's
- ŷ
- Evaluation metrics
  - Numeric (MAE, RMSE, etc.)
  - Binary (precision, recall, etc.)

Use cross validation to calculate these on train / val / test sets

What a regression model outputs

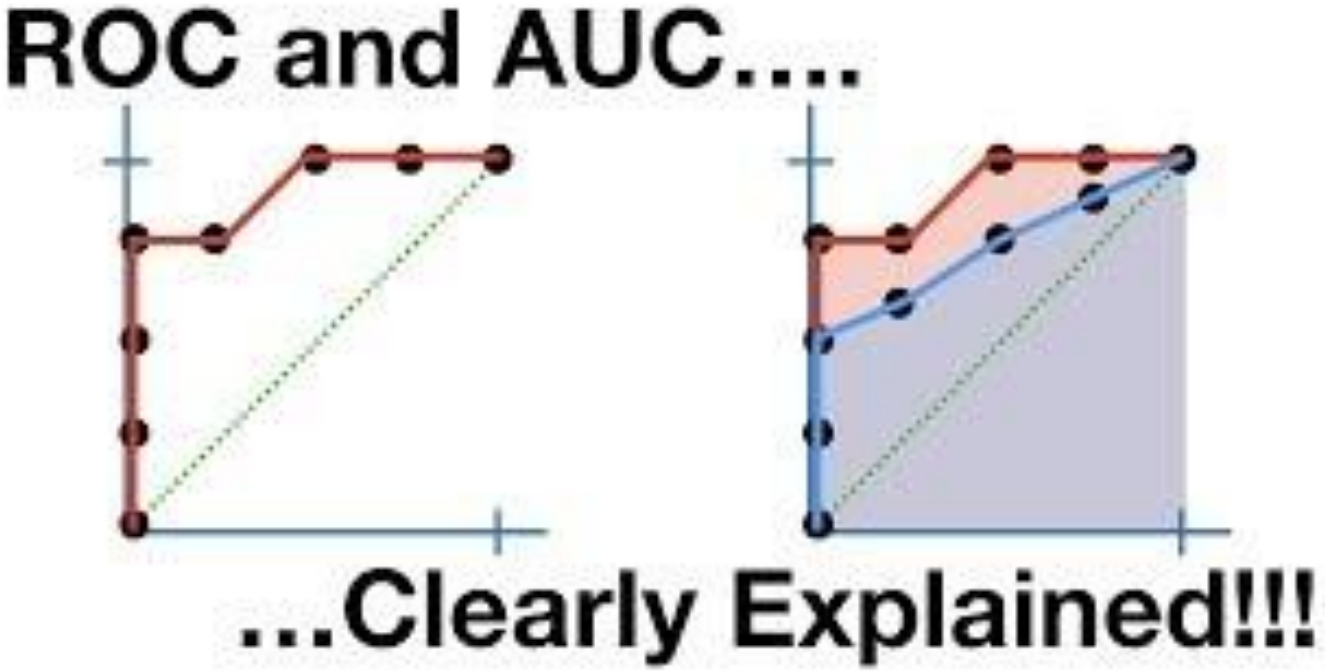
- α
- β's
- ŷ
- Evaluation metrics
  - Numeric (MAE, RMSE, etc.)
  - Binary (precision, recall, etc.)

Can get variance of metrics across folds of CV

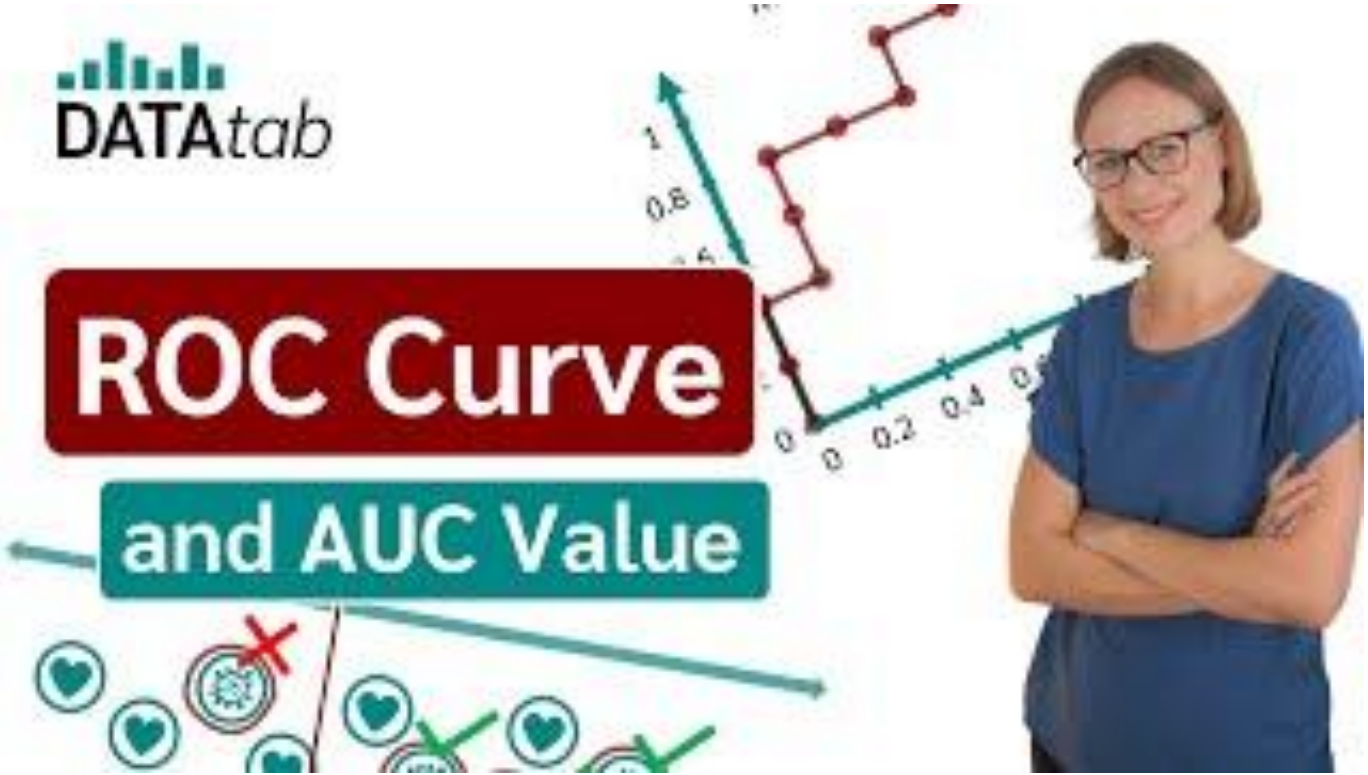
What a regression model outputs

- α
- β's
- ŷ
- Evaluation metrics
  - Numeric (MAE, RMSE, etc.)
  - Binary (precision, recall, etc.)

(See upcoming Canvas announcement for supplementary videos on ROC-AUC)



https://www.youtube.com/watch?v=4jRBRDbJemM



https://www.youtube.com/watch?v=QBVzZBsif20

What a regression model outputs

- α
- β's
- ŷ
- Evaluation metrics
  - Numeric (MAE, RMSE, etc.)
  - Binary (precision, recall, etc.)

Why do we care about variance? Want to know how **confident** we are in our evaluations!

https://medium.com/@ethanquesne/confidence-is-key-4654a06f182

What a regression model outputs

What if I care about the variance of these instead?

- α
- β's
- $\hat{y}$
- Evaluation metrics
  - Numeric (MAE, RMSE, etc.)
  - Binary (precision, recall, etc.)

What a regression model outputs

Variance here would tell me how confident I can be about the predictions of my regression

- α
- β's
- $\hat{y}$
- Evaluation metrics
  - Numeric (MAE, RMSE, etc.)
  - Binary (precision, recall, etc.)

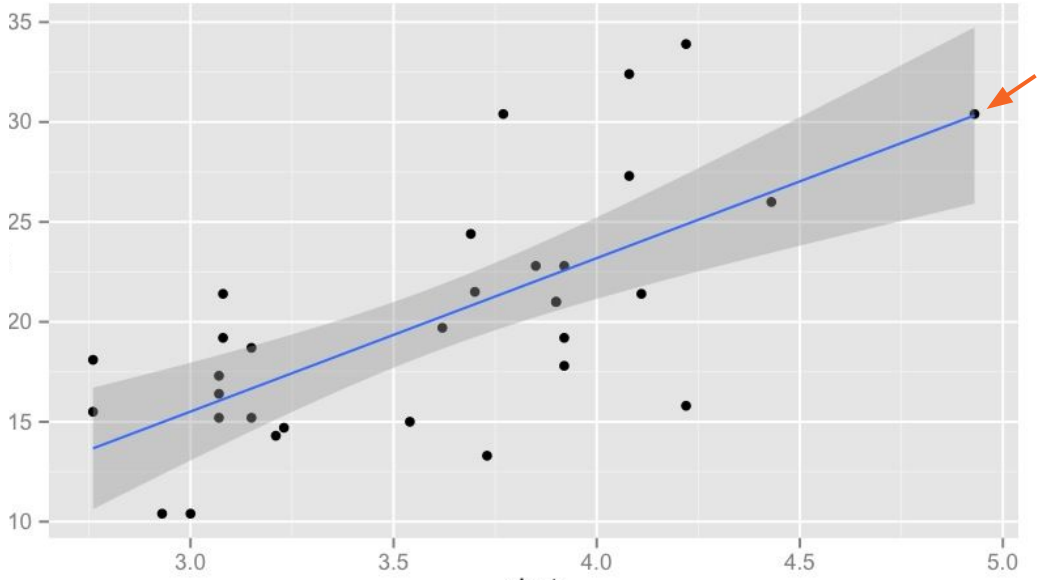
What a regression model outputs

Variance here would tell me how confident I can be about the predictions of my regression

(As opposed to confidence in generalizability of prediction *"accuracy"*)

- α
- β's
- ŷ
- Evaluation metrics
  - Numeric (MAE, RMSE, etc.)
  - Binary (precision, recall, etc.)

Have you seen this gray band?

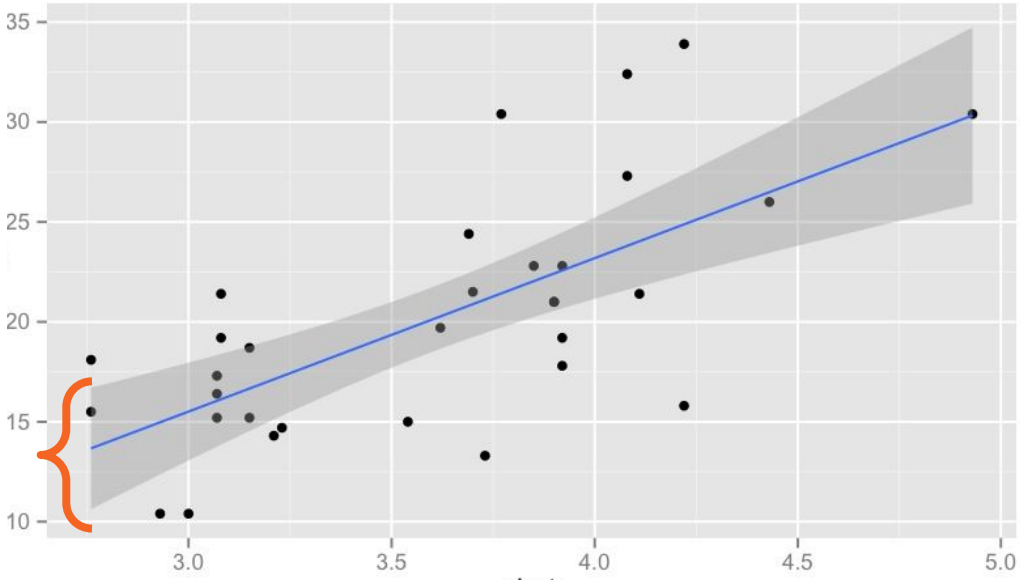


Regression line

https://stats.stackexchange.com/questions/85560/shape-of-confidence-interval-for-predicted-values-in-linear-regression

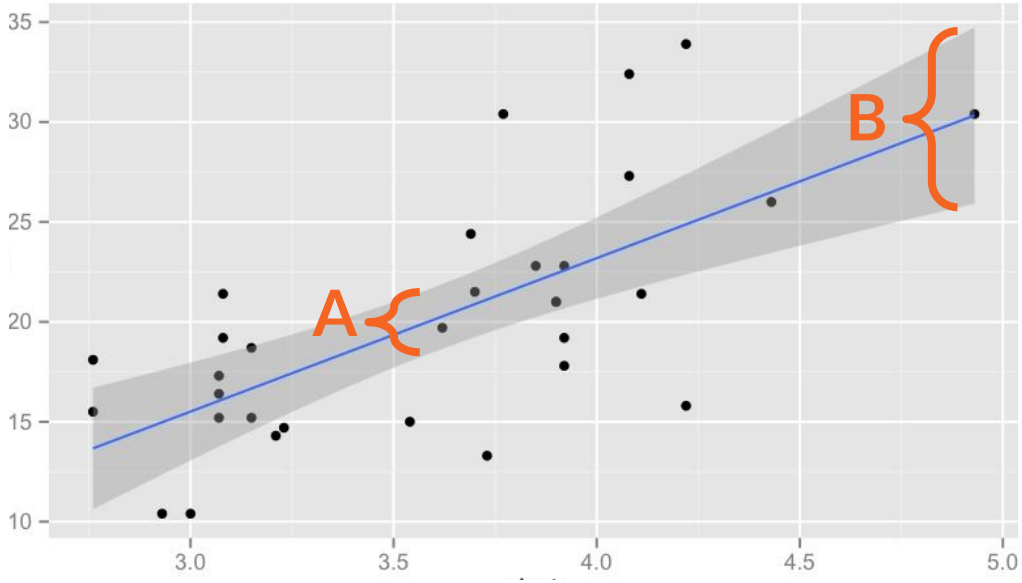
Have you seen this gray band?

Confidence Interval (CI)



https://stats.stackexchange.com/questions/85560/shape-of-confidence-interval-for-predicted-values-in-linear-regression

Where are you more confident in your model's prediction?

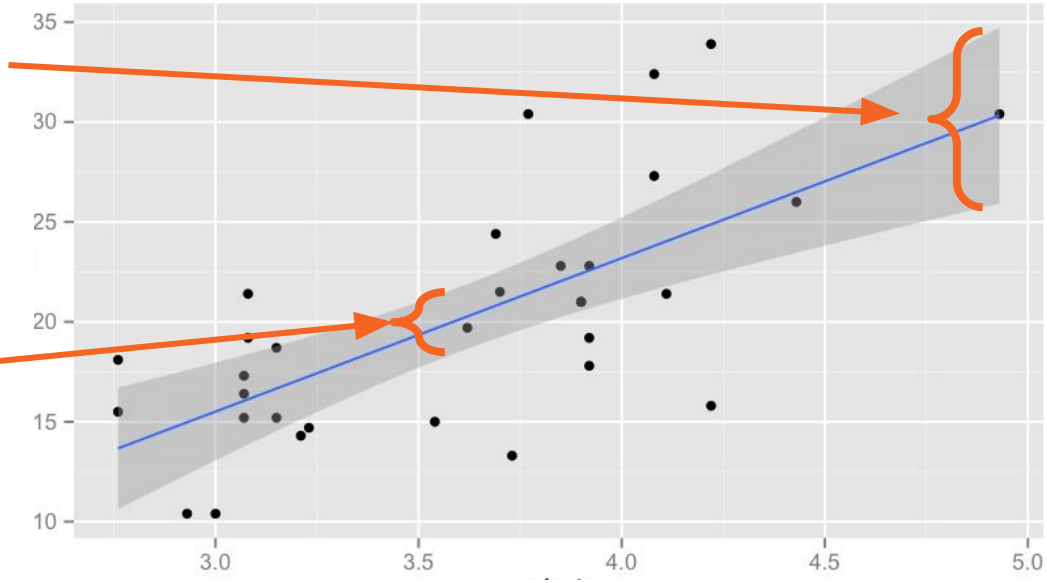


https://stats.stackexchange.com/questions/85560/shape-of-confidence-interval-for-predicted-values-in-linear-regression

Where are you more confident in your model's prediction? A

Wide CI (high var at this input) → less confident in our prediction here

Narrow CI (low var at this input) → more confident in our prediction here



https://stats.stackexchange.com/questions/85560/shape-of-confidence-interval-for-predicted-values-in-linear-regression

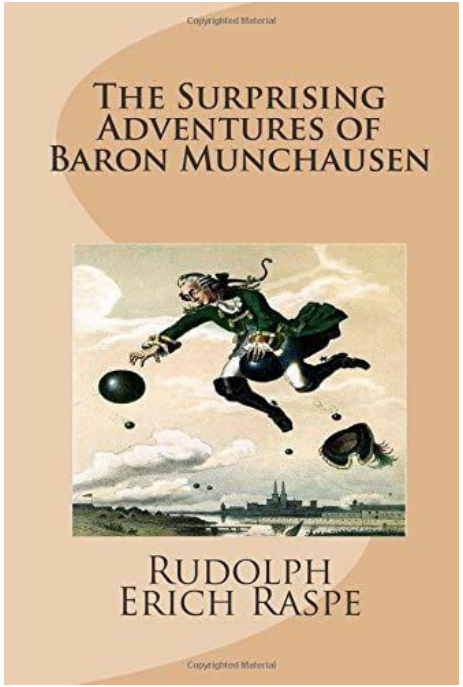
But how can we build confidence?

- All we have is a regression that we generated on a train set, gave us one set of estimated parameters [α and β's], and now predicts one ŷ for each input
- How can we get variance if we only get one ŷ per input?

Pull yourself up by the bootstraps!

https://uselessetymology.com/2019/11/07/the-origins-of-the-phrase-pull-yourself-up-by-your-bootstraps/

Pull yourself up by the bootstraps!



- "The Baron had fallen to the bottom of a deep lake. Just when it looked like all was lost, he thought to pick himself up by his own bootstraps"

  - From *The Surprising Adventures of Baron Munchausen* by Rudolph Erich Raspe (1785)

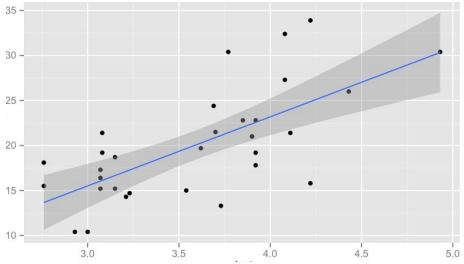
http://www3.wabash.edu/econometrics/EconometricsBook/chap23.htm | https://www.amazon.com/Surprising-Adventures-Baron-Munchausen/dp/1482557029

(Statistical) bootstraps!

- **Question**: How can we get variance if we only get one ŷ per input?

- **Answer**: "Bootstrapping"
  - "the use of the term bootstrap derives from the phrase *to pull oneself up by one's own bootstrap* . . ." [Efron and Tibshirani (1993), p. 5].

http://www3.wabash.edu/econometrics/EconometricsBook/chap23.htm



The confidence interval is a "bootstrap"

seaborn Installing Gallery Tutorial boot 0/14

- seaborn.violinplot
- seaborn.boxenplot
- seaborn.pointplot
- seaborn.barplot
- seaborn.countplot
- seaborn.lmplot
- **seaborn.regplot**
- seaborn.residplot
- seaborn.heatmap
- seaborn.clustermap
- seaborn.FacetGrid
- seaborn.pairplot
- seaborn.PairGrid
- seaborn.jointplot
- seaborn.JointGrid
- seaborn.set_theme
- seaborn.axes_style
- seaborn.set_style
- seaborn.plotting_context
- seaborn.set_context
- seaborn.set_color_codes

seaborn.regplot

```
seaborn.regplot(data=None, *, x=None, y=None, x_estimator=None, x_bins=None,
x_ci='ci', scatter=True, fit_reg=True, ci=95, n_boot=1000, units=None, seed=None,
order=1, logistic=False, lowess=False, robust=False, logx=False, x_partial=None,
y_partial=None, truncate=True, dropna=True, x_jitter=None, y_jitter=None,
label=None, color=None, marker='o', scatter_kws=None, line_kws=None, ax=None)
```

Plot data and a linear regression model fit.

There are a number of mutually exclusive options for estimating the regression model. See the tutorial for more information.

Parameters: **x, y: string, series, or vector array**

Input variables. If strings, these should correspond with column names in `data`. When pandas objects are used, axes will be labeled with the series name.

data : *DataFrame*

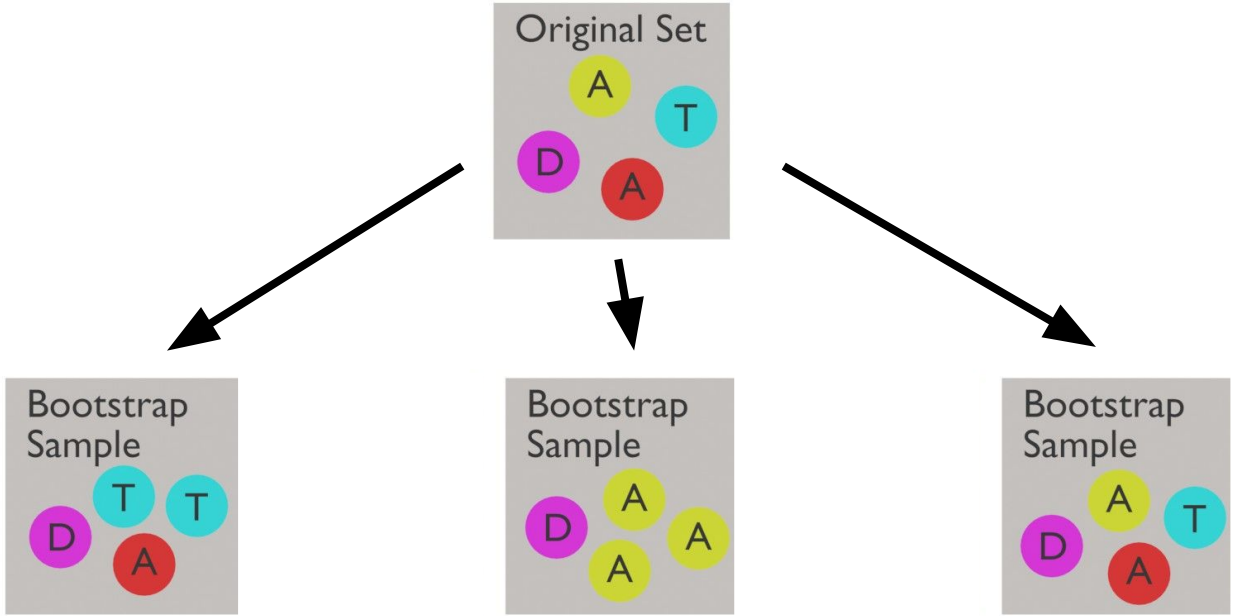
Tidy ("long-form") dataframe where each column is a variable and each row is an observation.

x_estimator : *callable that maps vector -> scalar, optional*

Apply this function to each unique value of `x` and plot the resulting estimate. This is useful when `x` is a discrete variable. If `x_ci` is given, this estimate will be bootstrapped and a confidence interval will be drawn.

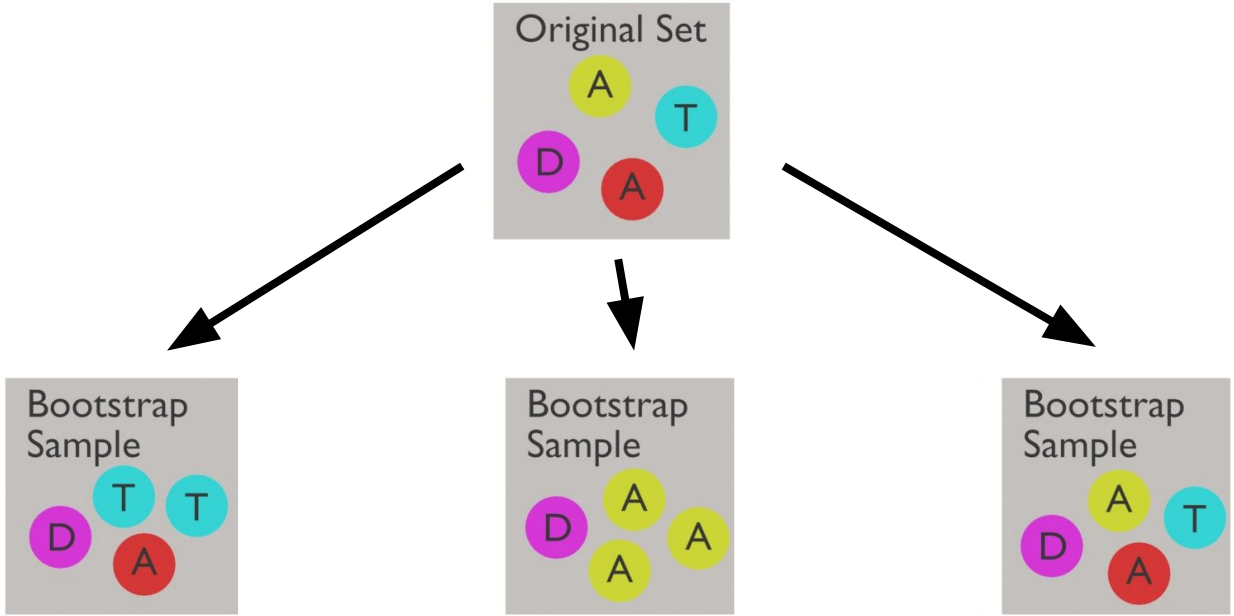
x_bins : *int or vector, optional*

Introducing: the Bootstrap sample



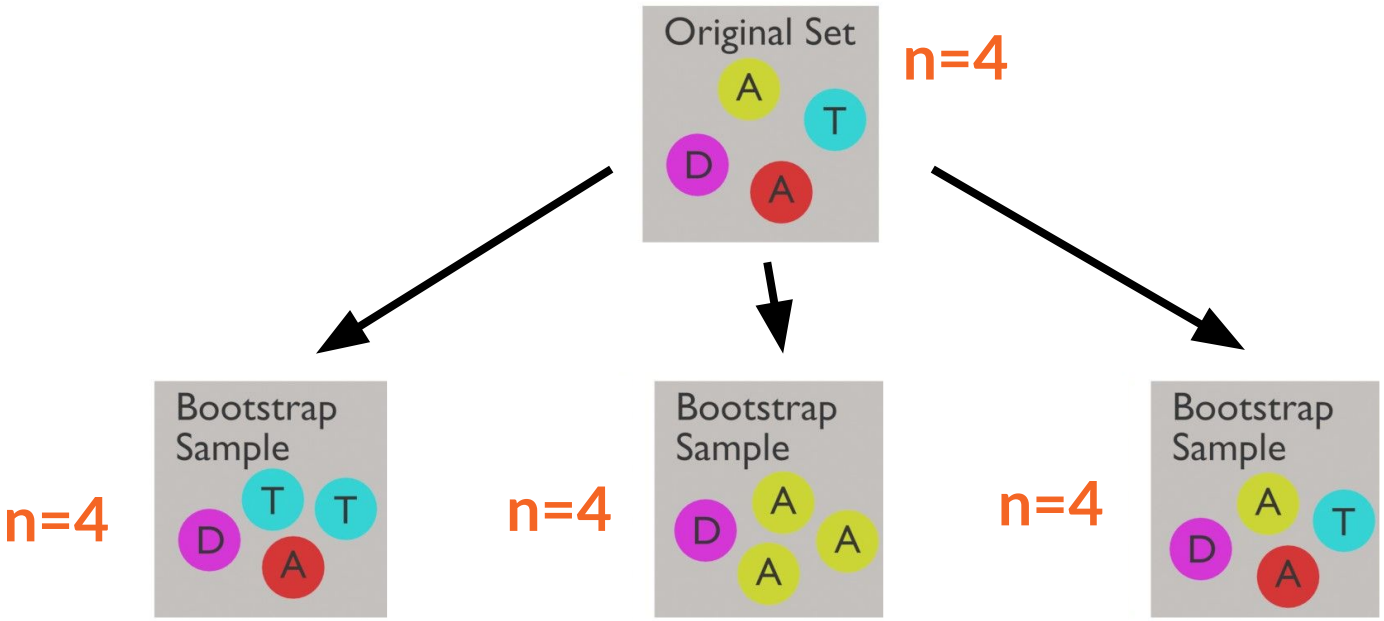
https://commons.wikimedia.org/wiki/File:Sampling_with_replacement_and_out-of-bag_dataset.jpg

Think, Pair, Share: What are 2 characteristics of bootstrap samples?



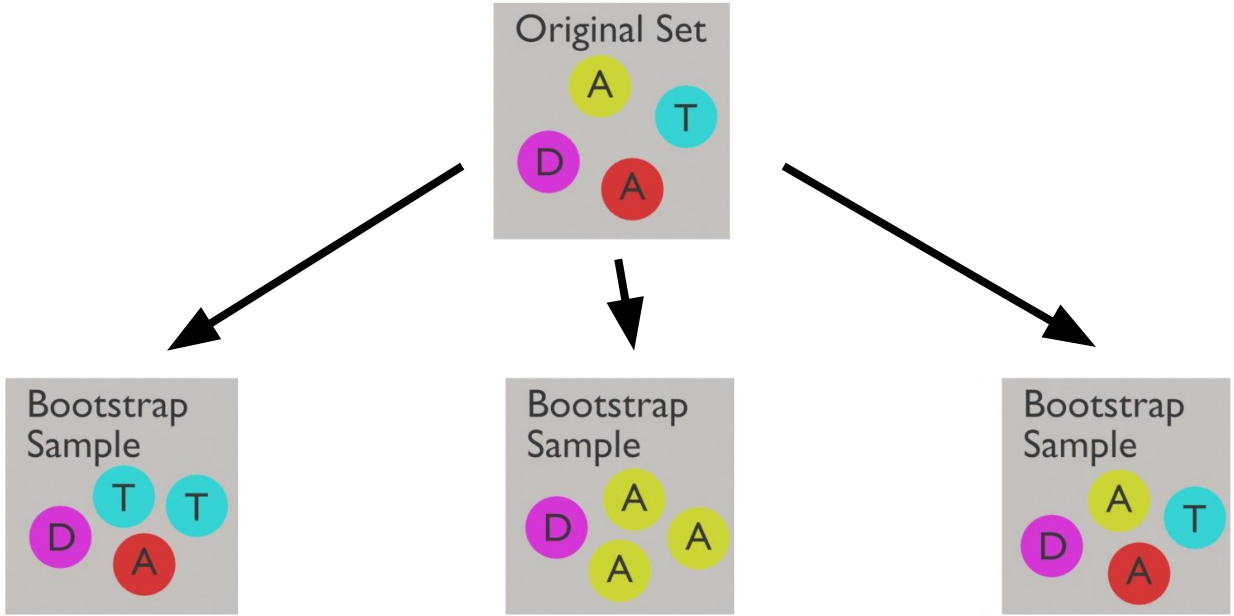
https://commons.wikimedia.org/wiki/File:Sampling_with_replacement_and_out-of-bag_dataset.jpg

Bootstrap sample: same size



https://commons.wikimedia.org/wiki/File:Sampling_with_replacement_and_out-of-bag_dataset.jpg

Bootstrap sample: *with* replacement

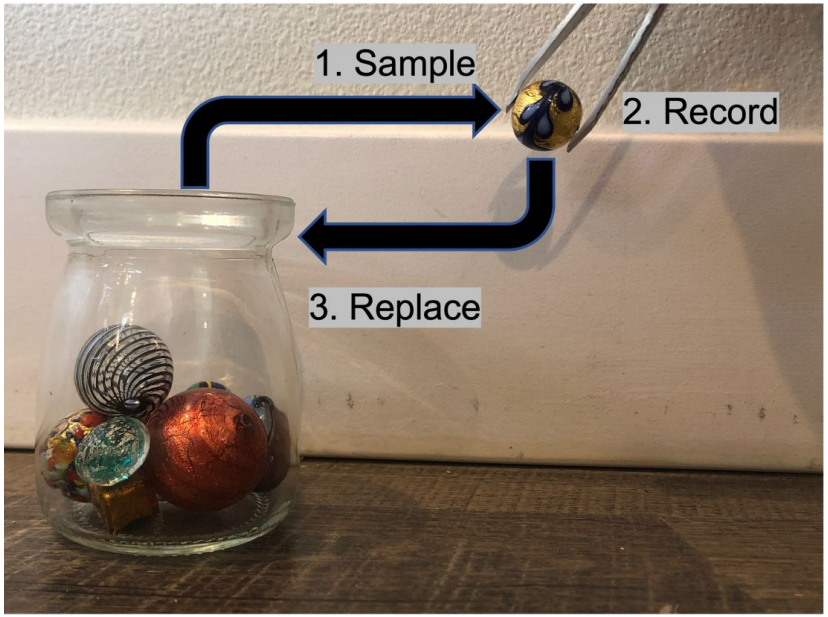


Multiple T's

No red A's or T's

https://commons.wikimedia.org/wiki/File:Sampling_with_replacement_and_out-of-bag_dataset.jpg

“Sampling with replacement”



- Sampling **with** replacement means that any two sample values are independent (whether we selected A before does not affect whether we select D in the future)

https://towardsdatascience.com/understanding-sampling-with-and-without-replacement-python-7aff8f47ebe4

Bootstrapping

- Calculate a function (e.g. mean) on multiple Bootstrapped samples of data
  - *n* sets of data → get mean *n* times
  - Now you get *n* estimated means $\bar{x}$
  - And you get *n* estimated values of $\bar{x}$ → you can calculate the **variance** of the **mean**!

Bootstrapped Means

3, 4, 5

Bootstrapped Means

3, 4, 5

3, 3, 4 4, 3, 5 3, 5, 3

Bootstrapped Means

3, 4, 5

3, 3, 4	4, 3, 5	3, 5, 3
3.33	*4*	*3.67*

Bootstrapped Means

3, 4, 5

Avg: 4
(of the mean)

3, 3, 4 | 4, 3, 5 | 3, 5, 3

3.33 | 4 | 3.67

Avg:
(of the means)
3.67

Bootstrapped Means

3, 4, 5

Stdev: ?!
(of the mean) – this statement is nonsensical

3, 3, 4 | 4, 3, 5 | 3, 5, 3

3.33 | *4* | *3.67*

Stdev:
(of the means)
0.34

Bootstrapped Means: 1000 times

```
x = np.array([3, 4, 5])
bs_means = np.zeros(1000)
for i in range(1000):
    bs_means[i] = np.random.choice(x, 3).mean()
bs_means.mean(), bs_means.std()
```

numpy.random.choice

`random.choice(a, size=None, replace=True, p=None)`

Generates a random sample from a given 1-D array

```
4.00 0.46
```

Bootstrapped Means: 1000 times

```
x = np.array([3, 4, 5])
bs_means = np.zeros(1000)
for i in range(1000):
    bs_means[i] = np.random.choice(x, 3).mean()
bs_means.mean(), bs_means.std()
```

numpy.random.choice

`random.choice(a, size=None, replace=True, p=None)`

Generates a random sample from a given 1-D array

If you bootstrap a lot (1000 times), you get close to the "true" value!

```
4.00 0.46
```

Bootstrapped Means

3, 4, 50

Bootstrapped Means

3, 4, 50

3, 3, 4 4, 3, 50 3, 50, 3

Bootstrapped Means

3, 4, 50

3, 3, 4 4, 3, 50 3, 50, 3

3.33 *19* *18.66*

Bootstrapped Means

3, 4, 50 } Avg: ?

3, 3, 4 4, 3, 50 3, 50, 3

3.33 19 18.66 } Avg: (of the means) 13.6

Bootstrapped Means

3, 4, 50

Avg: 19
(of the mean)

3, 3, 4 | 4, 3, 50 | 3, 50, 3

3.33 | 19 | 18.66

Avg: (of the means)
13.6

Bootstrapped Means

3, 4, 50

Stdev: ?!

(of the mean) – this statement is nonsensical

3, 3, 4 | 4, 3, 50 | 3, 50, 3

3.33 | *19* | *18.66*

Stdev:

(of the means)

7.31

Bootstrapped Means: 1000 times

```
x = np.array([3, 4, 50])
bs_means = np.zeros(1000)
for i in range(1000):
    bs_means[i] = np.random.choice(x, 3).mean()
bs_means.mean(), bs_means.std()
```

```
19.0 13.66
```

Bootstrapped Means: 1000 times

```
x = np.array([3, 4, 50])
bs_means = np.zeros(1000)
for i in range(1000):
    bs_means[i] = np.random.choice(x, 3).mean()
bs_means.mean(), bs_means.std()
```

Again, we recover
(3+4+50)/3 = 19

```
19.0 13.66
```

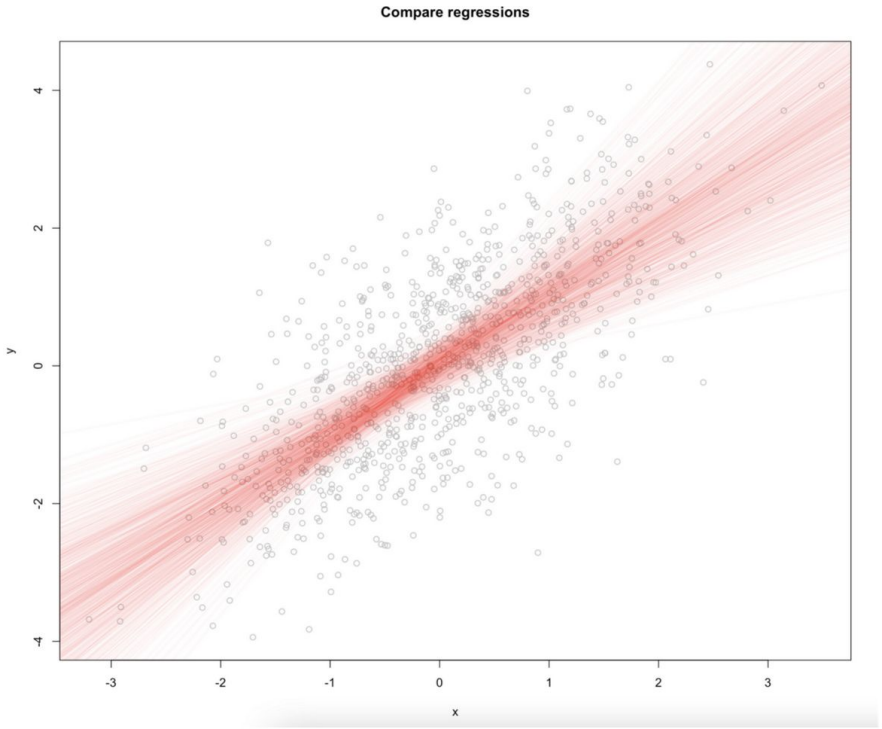
Bootstrapping with regressions!

- Fit your regression model on multiple Bootstrapped samples of data
  - *n* sets of data → fitting your model *n* times
  - Now you get *n* sets of estimated parameters [α and β's]
  - And you get *n* estimated values of $\hat{y}$ → you can calculate the variance across these $\hat{y}$'s!

Bootstrapping

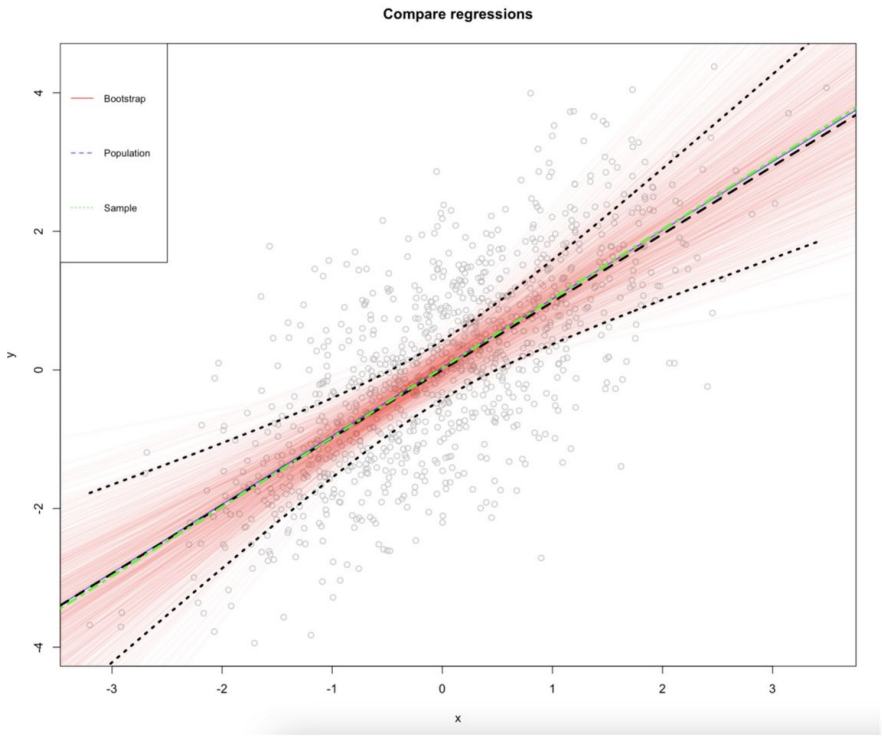
1,000 red regression lines are plotted!

Each line was trained on a *bootstrapped sample* of the *population*



https://towardsdatascience.com/bootstrap-regression-in-r-98bfe4ff5007

Bootstrapping

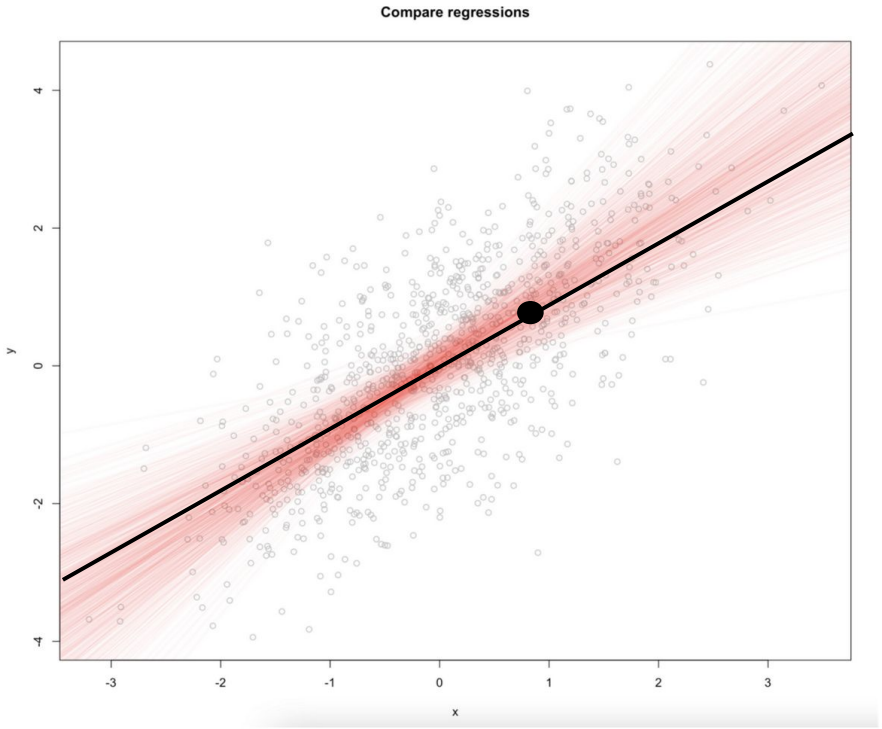


Same chart but adding some dotted lines…

Look familiar? Seems like the “gray band” CI shape!

https://towardsdatascience.com/bootstrap-regression-in-r-98bfe4ff5007

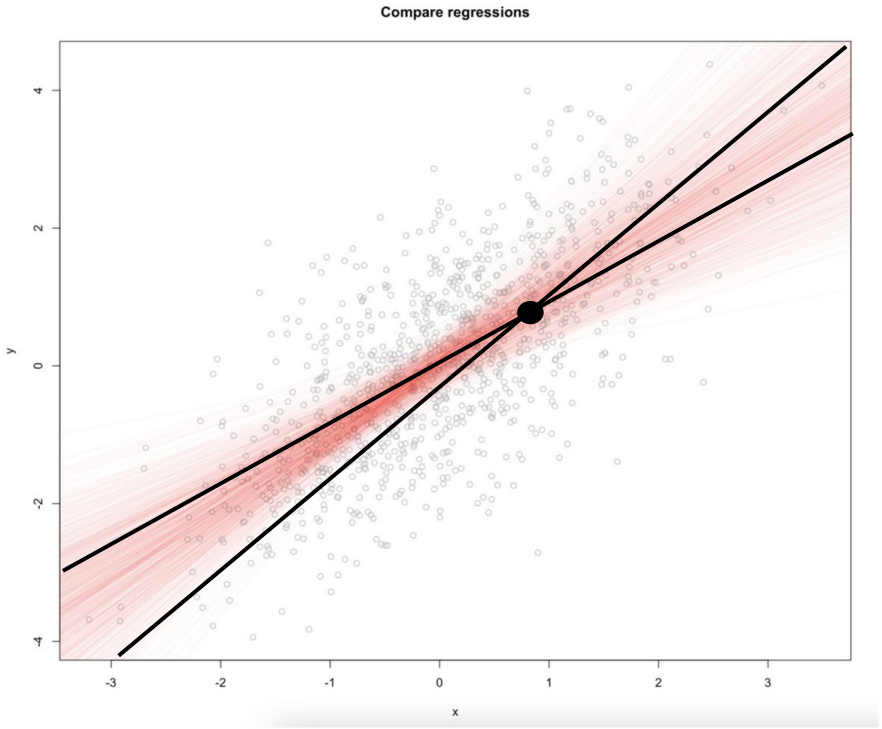
What's up with the fan shape?



Intuition: a linear regression necessarily passes through the point ($\bar{x}$, $\bar{y}$). When sampling, the slope is going to change much more than the mean point, creating a "fan"

https://towardsdatascience.com/bootstrap-regression-in-r-98bfe4ff5007

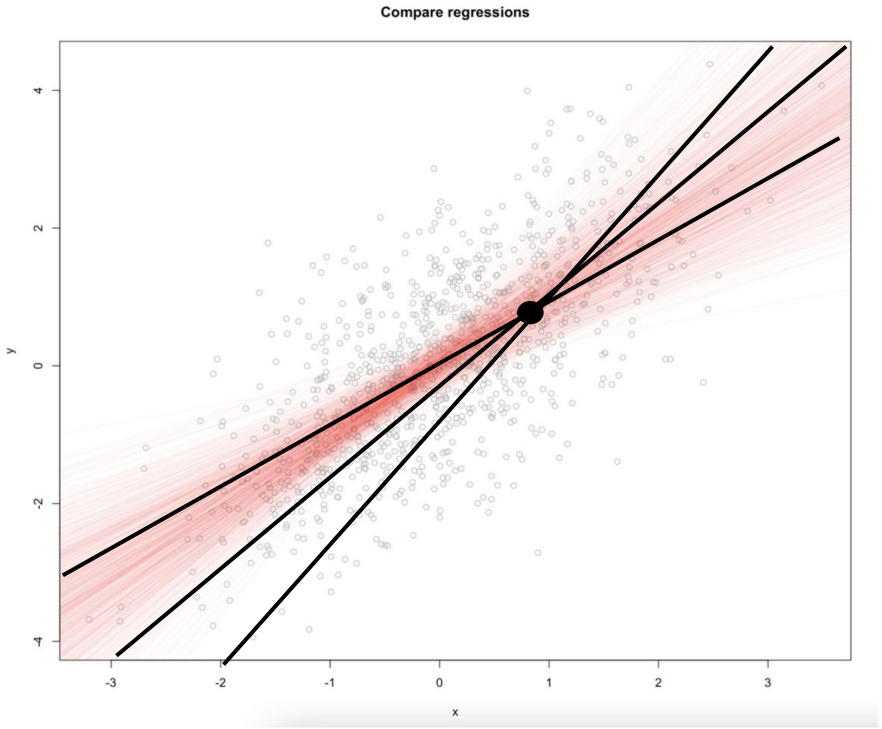
What's up with the fan shape?



Intuition: a linear regression necessarily passes through the point ($\bar{x}$, $\bar{y}$). When sampling, the slope is going to change much more than the mean point, creating a "fan"

https://towardsdatascience.com/bootstrap-regression-in-r-98bfe4ff5007

What's up with the fan shape?



Intuition: a linear regression necessarily passes through the point ($\bar{x}$, $\bar{y}$). When sampling, the slope is going to change much more than the mean point, creating a "fan"

https://towardsdatascience.com/bootstrap-regression-in-r-98bfe4ff5007

Bootstrapping

- Fit your regression model on multiple bootstrapped samples of data
  - *n* sets of data → fitting your model *n* times
  - Now you get *n* sets of estimated parameters [α and β's]
  - And you get *n* estimated values of $\hat{y}$ → you can calculate the variance across these $\hat{y}$'s!

Recall: Bootstrap samples are selected *with replacement*

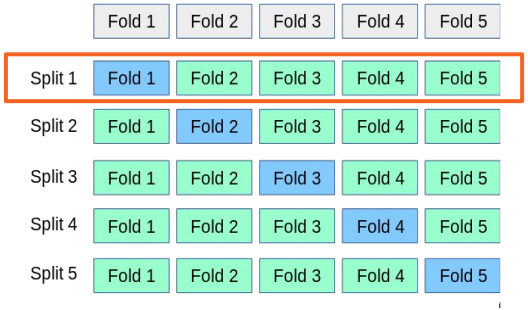
Resampling methods

Without Replacement	With Replacement
Cross Validation	Bootstrap

Resampling methods

Without Replacement	With Replacement
Cross Validation	Bootstrap

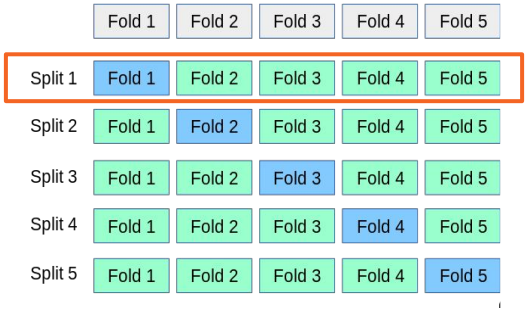
Fold 1 contents not in Folds 2-5

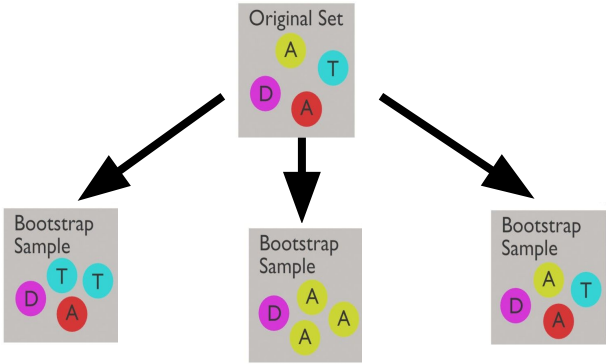


Resampling methods

Without Replacement	With Replacement
Cross Validation	Bootstrap

Fold 1 contents not in Folds 2-5





Original set values can appear multiple times

Resampling methods

Without Replacement	With Replacement
Cross Validation	Bootstrap
Leave-one-out	Jackknife

Resampling methods

Without Replacement	With Replacement
Cross Validation	Bootstrap
Leave-one-out	Jackknife

Calculate statistic (evaluation metric; mean and variance of multiple evaluation metrics) based on the left-out fold (using a model trained on the left-in samples)

Resampling methods

Without Replacement	With Replacement
Cross Validation	Bootstrap
Leave-one-out	Jackknife

Calculate statistic (predicted values; mean and variance of multiple predicted values) based on the left-in samples

Bootstrapping

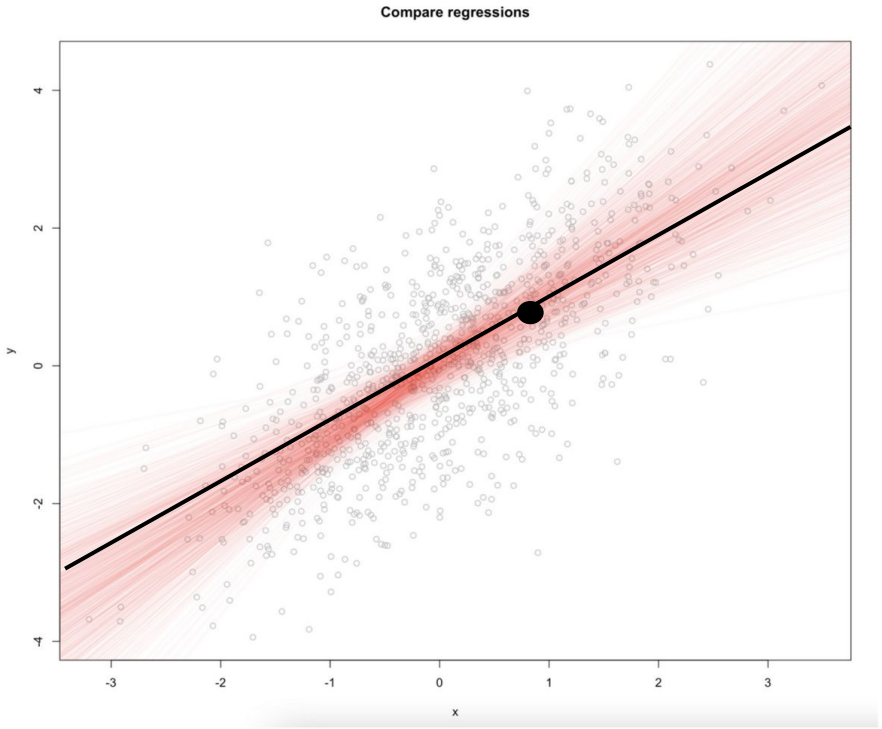
- Fit your regression model on multiple (sub)sets of data
  - *n* sets of data → fitting your model *n* times
  - Now you get *n* sets of estimated parameters [α and β's]
  - And you get *n* estimated values of $\hat{y}$ → you can calculate the variance across these $\hat{y}$'s!

Bootstrapping

- Fit your regression model on multiple (sub)sets of data
  - *n* sets of data → fitting your model *n* times
  - Now you get *n* sets of estimated parameters [α and β's]
  - And you get *n* estimated values of ŷ → you can calculate the variance across these ŷ's!

Not just the variance!

Bootstrapping



Each red regression line was run on 1 bootstrapped sample

Each line gives one set of estimated parameters [α and β's]

Each line outputs one sample $\hat{y}$ per input

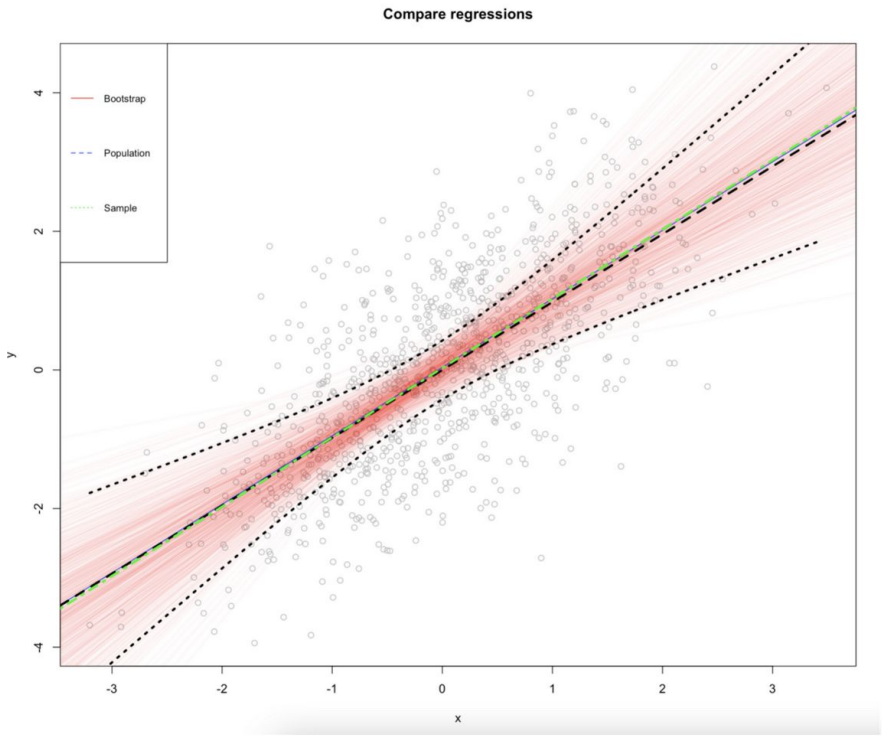
https://towardsdatascience.com/bootstrap-regression-in-r-98bfe4ff5007

Bootstrapping

You can plot distributions of, calculate averages across, and calculate variance of:

- sample ŷ's
- α estimates
- β estimates

Doing so can help you determine how confidently you claim a regression prediction or interpretation!



https://towardsdatascience.com/bootstrap-regression-in-r-98bfe4ff5007

Admin & 1 min break

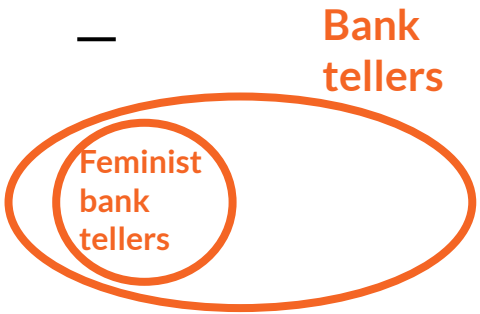
- Phase 2 due this Thursday; you will be sharing your Phase 2 during Friday discussion for peer review and feedback
- HW4 hint on D1: make sure you use pd.merge instead of df.merge in C3. If you're getting different outputs than us, try swapping your join (*a onto b* instead of *b onto a*) so your index order is the same as ours.

Which is more likely to be true?

Linda is 31 years old, single, outspoken, and very bright. She majored in philosophy. As a student, she was deeply concerned with issues of discrimination and social justice, and also participated in anti-nuclear demonstrations.

1) Linda is a bank teller.
2) Linda is a bank teller and is active in the feminist movement.

“Thinking Fast and Slow”, Kahneman and Tversky

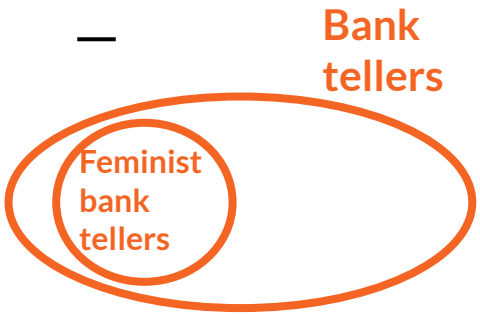


Which is more likely to be true?

Linda is 31 years old, single, outspoken, and very bright. She majored in philosophy. As a student, she was deeply concerned with issues of discrimination and social justice, and also participated in anti-nuclear demonstrations.

1) Linda is a bank teller.
2) Linda is a bank teller and is active in the feminist movement.

"Thinking Fast and Slow", Kahneman and Tversky



Which is more likely to be true?

Linda is 31 years old, single, outspoken, and very bright. She majored in philosophy. As a student, she was deeply concerned with issues of discrimination and social justice, and also participated in anti-nuclear demonstrations.

1) Linda is a bank teller.
2) Linda is a bank teller and is active in the feminist movement.

Also, probabilities can't be bigger than one. If we're multiplying multiple probabilities (assuming bank teller and feminist are i.i.d.), the probability for option 2 must less than option 1!

Pr(bank teller) = 0.2
Pr(feminist) = 0.5
Pr(bank teller & feminist) = 0.2*0.5 = 0.1 < 0.2

"Thinking Fast and Slow", Kahneman and Tversky

Probability distributions

A probability distribution has the following parts:

1. An **event space** defining the set of possible outcomes
2. **Parameters** (numbers) that define properties of the distribution
3. A **probability function** that maps each element of the event space to a probability between 0 and 1

Example: binary variable

- **Event space**: 0/1
- **Parameters**: p = probability of 1
- **Probability function**: p for 1, $(1-p)$ for 0

https://en.wikipedia.org/wiki/Penny_(United_States_coin)

Six-sided die

Set of possible outcomes

Variables that define distribution properties

Maps each element of event space to a probability

- **Event space**: ?
- **Parameters**: ?
- **Probability function**: ?

https://en.wikipedia.org/wiki/Dice

Six-sided die

Set of possible outcomes

Variables that define distribution properties

Maps each element of event space to a probability

- **Event space**: 1, 2, 3, 4, 5, 6
- **Parameters**: None necessary!
- **Probability function**: Always output **1/6**

https://en.wikipedia.org/wiki/Dice

Example: six-sided die

Set of possible outcomes

- **Event space**: 1, 2, 3, 4, 5, 6

Variables that define distribution properties

- **Parameters**: None necessary!

Maps each element of event space to a probability

- **Probability function**: Always output **1/6**

Have to make some assumptions, e.g. these are fair die!

https://en.wikipedia.org/wiki/Dice

What is an opinion poll?

- Some underlying population value p
  - **Example**: proportion of people who prefer cats to dogs
- Sample N individuals and ask them the question

What animal is your favorite?

Dogs 🐶

Cats 🐱

Something else 🦎

What is an opinion poll?

- Some underlying population value *p*
  - **Example**: proportion of people who prefer cats to dogs
- Sample *N* individuals and ask them the question
- If we sample 100 of you and 28 responded preferring cats, what do we estimate as the proportion of people who prefer cats?

What is an opinion poll?

- Some underlying population value *p*
  - **Example**: proportion of people who prefer cats to dogs
- Sample *N* individuals and ask them the question
- If we sample 100 of you and 28 responded preferring cats, what do we estimate as *p*?
  - **28/100 = 28%, but *why*? Sample != Population**

What is an opinion poll?

- Some underlying population value *p*
  - **Example**: proportion of people who prefer cats to dogs
- Sample *N* individuals and ask them the question
- Let X = number of respondents preferring cats. The true proportion *p* is unknown, but we think X/N is a good **estimate** of *p*

Example: opinion poll, single yes/no question

- **Event space**: 0 ... N yeses
- **Parameters**: N, p
- **Probability function**: ?

Example: opinion poll, single yes/no question

- **Event space**: 0 ... N yeses
- **Parameters**: N, p
- **Probability function**: from the **Binomial distribution** $\Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$

The Binomial Distribution

Counting the number of **positive events X out of total events N** where each event has **probability *p*** to be positive is the pattern for a binomial distribution

We'll start by defining the **probability** of a sequence, then show how to relate this to a count

Given *p*, what is the sequence probability?

Event = 🐱 or 🐶

p = probability of 🐱 = 0.2

Write the probability of each sequence occurring:

1. 🐱
2. 🐶
3. 🐱 then 🐶

Given *p*, what is the sequence probability?

Event = 🐱 or 🐶

p = probability of 🐱 = 0.2

Did you write…

1. **0.2**
2. **0.8**
3. **0.16**

Caution! What did you assume?

- We can **multiply** individual probabilities to get the probability of a sequence **if the data are i.i.d.**
- i.i.d. means the respondents' preferences are:
  - **Independent**, and
  - **Identically distributed**

Caution! What did you assume?

- We can **multiply** individual probabilities to get the probability of a sequence **if the data are i.i.d.**
- i.i.d. means the respondents' preferences are:
  - **Independent**, and
  - **Identically distributed**
- We'll assume this is true for our dog/cat example
  - No respondents are being coerced by other respondents to answer one way or the other
  - No reason to expect more or less than average cat appreciation

Instead of a number, let's use the variable *p*

Event = 🐱 or 🐶, i.i.d.

p = probability of 🐱

Write the probability of these sequences in terms of *p*:

1. 🐱
2. 🐶
3. 🐱 then 🐱 then 🐶

What if we knew what *p* is?

Event = 🐱 or 🐶, **i.i.d.**

p = probability of 🐱

Write the probability of these sequences in terms of *p*:

1. 🐱 = **p**
2. 🐶 = **(1-p)**
3. 🐱 then 🐱 then 🐶 = **p * p * (1-p)**

What are the sequence probabilities?

p = probability of 🐱 = 0.5

Write the i.i.d. probability of: (products are ok)

🐱 then 🐱 then 🐶

🐶 then 🐱 then 🐱

What are the sequence probabilities?

p = probability of 🐱 = 0.5

Write the i.i.d. probability of: (products are ok)

🐱 then 🐱 then 🐶 = 0.5 * 0.5 * 0.5 = 0.5^3

🐶 then 🐱 then 🐱 = 0.5 * 0.5 * 0.5 = 0.5^3

What are the sequence probabilities?

p = probability of 🐱 = 0.2

Write the i.i.d. probability of: (products are ok)

1. 🐱 then 🐱 then 🐶
2. 🐶 then 🐱 then 🐱
3. 🐱 then 🐶 then 🐶

What are the sequence probabilities?

p = probability of 🐱 = 0.2

Write the i.i.d. probability of: (products are ok)

1. 🐱 then 🐱 then 🐶 = 0.2 * 0.2 * 0.8 = $0.2^2 * 0.8$
2. 🐶 then 🐱 then 🐱 = 0.8 * 0.2 * 0.2 = $0.2^2 * 0.8$
3. 🐱 then 🐶 then 🐶 = 0.2 * 0.8 * 0.8 = $0.2 * 0.8^2$

What are the sequence probabilities?

p = probability of 🐱 = 0.2

Write the i.i.d. probability of: (products are ok)

1. 🐱 then 🐱 then 🐶 = 0.2 * 0.2 * 0.8 = 0.2^2 * 0.8 = 0.032
2. 🐶 then 🐱 then 🐱 = 0.8 * 0.2 * 0.2 = 0.2^2 * 0.8 = 0.032
3. 🐱 then 🐶 then 🐶 = 0.2 * 0.8 * 0.8 = 0.2 * 0.8^2 = 0.128

Same sequence in different order still gives you the same probability!

What are the sequence probabilities?

p = probability of 🐱 = 0.2

Write the i.i.d. probability of: (products are ok)

1. 🐱 then 🐱 then 🐶 = $0.2 * 0.2 * 0.8 = 0.2^2 * 0.8 = 0.032$
2. 🐶 then 🐱 then 🐱 = $0.8 * 0.2 * 0.2 = 0.2^2 * 0.8 = 0.032$
3. 🐱 then 🐶 then 🐶 = $0.2 * 0.8 * 0.8 = 0.2 * 0.8^2 = 0.128$

⅔ dogs has a higher probability than ⅔ cats since probability of dog (0.8) > probability of cat (0.2)

Sequences vs. Counts

- The difficulty is transitioning from a **sequence** of events to a single event that represents a **count**

Sequences vs. Counts

- The difficulty is transitioning from a **sequence** of events to a single event that represents a **count**
- We already know: the probability of a sequence of i.i.d. events is the product of the probabilities
- If we only know the total count of events, not the original sequence, we **need to account for how many sequences** result in a count

What if we only know counts?

How many sequences result in:

3 🐱

2 🐱 and 1 🐶

1 🐱 and 2 🐶

3 🐶

What if we only know counts?

How many sequences result in:

3 🐱: 🐱🐱🐱

2 🐱 and 1 🐶: 🐶🐱🐱, 🐱🐶🐱, 🐱🐱🐶

1 🐱 and 2 🐶: 🐱🐶🐶, 🐶🐱🐶, 🐶🐶🐱

3 🐶: 🐶🐶🐶

https://www.flickr.com/photos/bods/6119906063

Combination

From Wikipedia, the free encyclopedia

This article is about the mathematics of selecting part of a collection. For other uses, see Combination (disambiguation).

"COMBIN" and "nCr" redirect here. For other uses, see Combin (disambiguation) and NCR (disambiguation).

In mathematics, a **combination** is a selection of items from a set that has distinct members, such that the order of selection does not matter (unlike permutations). For example, given three fruits, say an apple, an orange and a pear, there are three combinations of two that can be drawn from this set: an apple and a pear; an apple and an orange; or a pear and an orange. More formally, a *k*-**combination** of a set *S* is a subset of *k* distinct elements of *S*. So, two combinations are identical if and only if each combination has the same members. (The arrangement of the members in each set does not matter.) If the set has *n* elements, the number of *k*-combinations, denoted as C_k^n, is equal to the binomial coefficient

$$\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{k(k-1)\cdots 1},$$

which can be written using factorials as $\frac{n!}{k!(n-k)!}$ whenever $k \leq n$, and which is zero when $k > n$. This

Counting i.i.d. events

The number of sequences of *N* events in {0, 1}, with *X* 1s:

$$\frac{N!}{X!(N-X)!}$$

"binomial" coefficient

Pascal's Triangle

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

https://en.wikipedia.org/wiki/Pascal%27s_triangle

Pascal's Triangle is binomial coefficients!!!

$$
\begin{array}{ccccccccccccccc}
 & & & & & & & 1 & & & & & & & \\
 & & & & & & 1 & & 1 & & & & & & \\
 & & & & & 1 & & 2 & & 1 & & & & & \\
 & & & & 1 & & 3 & & 3 & & 1 & & & & \\
 & & & 1 & & 4 & & 6 & & 4 & & 1 & & & \\
 & & 1 & & 5 & & 10 & & 10 & & 5 & & 1 & & \\
 & 1 & & 6 & & 15 & & 20 & & 15 & & 6 & & 1 & \\
1 & & 7 & & 21 & & 35 & & 35 & & 21 & & 7 & & 1
\end{array}
$$

https://en.wikipedia.org/wiki/Pascal%27s_triangle

Find the probabilities

Hint: count the sequences

p = probability of 🐱 = 0.5

Write the probability of each sequence occurring (in any order): (products are ok)

1. 2 🐱 and 1 🐶
2. 1 🐱 and 2 🐶
3. 3 🐶

Find the probabilities

p = probability of 🐱 = 0.5

Write the probability of: (products are ok)

1. 2 🐱 and 1 🐶 = **3** * 0.5^3
2. 1 🐱 and 2 🐶 = **3** * 0.5^3
3. 3 🐶 = **1** * 0.5^3

Different *p*, same question

p = probability of 🐱 = 0.1

Write the probability of: (products are ok)

1. 2 🐱 and 1 🐶
2. 1 🐱 and 2 🐶
3. 3 🐶

Different *p*, same question

p = probability of 🐱 = 0.1

Write the probability of: (products are ok)

1. 2 🐱 and 1 🐶 = **3** * 0.1^2 * 0.9
2. 1 🐱 and 2 🐶 = **3** * 0.1 * 0.9^2
3. 3 🐶 = **1** * 0.9^3

Different *p*, same question

p = probability of 🐱 = 0.1

Write the probability of: (products are ok)

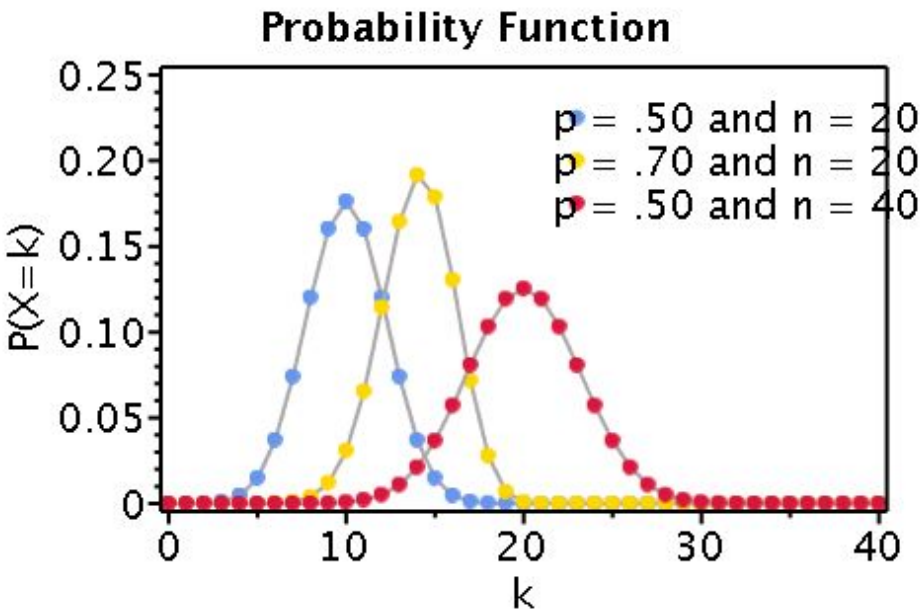
1. 2 🐱 and 1 🐶 = **3** * 0.1^2 * 0.9 = 0.027
2. 1 🐱 and 2 🐶 = **3** * 0.1 * 0.9^2 = 0.243
3. 3 🐶 = **1** * 0.9^3 = 0.729

Counting i.i.d. events

If the event is 0 or 1, and we know p=P(1), N = the total number of trials, and X = the number of 1's:

$$\underbrace{\frac{N!}{X!(N-X)!}}_{\text{How many sequences? (Binomial coefficient)}} \quad \underbrace{p^X(1-p)^{N-X}}_{\text{Probability of one sequence}}$$

Counting i.i.d. events is the basis for binomial distribution



https://de.maplesoft.com/support/help/maple/view.aspx?path=MathApps/NormalApproximationOfBinomialDistribution

Probability distributions

- For each possible value of a variable X, the probability function P(X) assigns a probability value
- All probabilities P(X) are between 0 and 1
- Sum of probabilities for all values of X is 1.0

Binomial distribution μ and σ

$$\mathbb{E}[X] = N\ p$$

$$Var[X] = N\ p(1-p)$$

$$Std[X] = \sqrt{N\ p(1-p)}$$

What is an opinion poll?

- Some underlying population value *p*
  - **Example**: proportion of people who prefer cats to dogs
- Sample *N* individuals and ask them the question
- Let X = number of respondents preferring cats. The true proportion *p* is unknown, but we think X/N is a good **estimate** of *p*

https://medium.com/@ethanquesne/confidence-is-key-4654a06f182

Is this a *good* opinion poll?

- Want to know: how **good of an estimate** is our sample's *X/N* for the true population's *p*?
    - What is a common fatal flaw in surveys that lead to poor estimates?

Sample variance is the average squared distance to the sample mean

(using N to keep things simple)

$$\frac{\Sigma_i (X_i - \bar{X})^2}{N}$$

Is this a *good* opinion poll?

- Want to know: how **good of an estimate** is our sample's X/N for the true population's p?
  - What is a common fatal flaw in surveys that lead to poor estimates? **Small sample sizes (N)!**

Is this a *good* opinion poll?

- Want to know: how **good of an estimate** is our sample's X/N for the true population's p?
  - What is a common fatal flaw in surveys that lead to poor estimates? **Small sample sizes (N)!**
  - **Small $N \rightarrow$ high variance** (Look at the denominator)
  - If we know the variance of X/N, we can quantify our (un)certainty in our estimate!

Is this a *good* opinion poll?

- Want to know: how **good of an estimate** is our sample's *X/N* for the true population's *p*?
    - Aside from small sample sizes, what else might lead to high variance?

Is this a *good* opinion poll?

- Want to know: how **good of an estimate** is our sample's *X/N* for the true population's *p*?
  - Aside from small sample sizes, what else might lead to high variance?
  - **High $(x_i - \bar{x}) \rightarrow$ high variance** (in the numerator)
  - If people's responses are very different from each other, you might not be very confident in your *X/N* estimate

Is this a *good* opinion poll?

- Want to know: how **good of an estimate** is our sample's *X/N* for the true population's *p*?
  - Calculate the **variance** of your estimate to quantify your confidence in that estimate
    - *"I estimate p = 0.28, but it might be a little higher, might be a little lower"*
  - How might we estimate variance if given 100 responses to our poll?

Tying it back to Python…

numpy.random.binomial

random.binomial(*n, p, size=None*)

Draw samples from a binomial distribution.

Samples are drawn from a binomial distribution with specified parameters, n trials and p probability of success where n an integer >= 0 and p is in the interval [0,1]. (n may be input as a float, but it is truncated to an integer in use)

```
>>> n, p = 10, .5  # number of trials, probability of each trial
>>> s = np.random.binomial(n, p, 1000)
# result of flipping a coin 10 times, tested 1000 times.
```

https://numpy.org/doc/stable/reference/random/generated/numpy.random.binomial.html

Counting individual # Cat “wins” when 100 people are surveyed

```
poll = np.random.binomial(1, 0.28, size=(1,100))
print(poll)
```

```
[[0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0
  1 1 0 0 0 0 0 0 1 1 1 0 0 1 1 0 1 0 0 0 1 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0
  1 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0]]
```

Counting individual # Cat “wins” when 100 people are surveyed

```
poll = np.random.binomial(1, 0.28, size=(1,100))
print(poll)
```

```
[[0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0
  1 1 0 0 0 0 0 0 1 1 1 0 0 1 1 0 1 0 0 0 1 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0
  1 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0]]
```

```
poll.sum()
```

```
27
```

Summing # Cat “wins” directly when 100 people are surveyed

```
np.random.binomial(100, 0.28, 1)
```

```
array([31])
```

Summing # Cat “wins” directly when 100 people are surveyed

```
np.random.binomial(100, 0.28, 1)
```

```
array([31])
```

But if I hit ‘run’ again, I can get a different number! E.g., `array([29])` or `array([33])`

To bootstrap, we could do a for loop like before...

```
poll = np.random.binomial(1, 0.28, 100)

bs_polls = np.zeros(1000)
for i in range(1000):
    bs_polls[i] = np.random.choice(poll, 100).sum()
```

(This mirrors our code in slide 38)

Bootstrapping the # Cat “wins”...

```
np.random.binomial(1, 0.28, size=(1000,100))
```

We can re-run our poll on 100 people… 1,000 times!

Bootstrapping the # Cat "wins"...

```
np.random.binomial(1, 0.28, size=(1000,100))
array([[1, 0, 0, ..., 1, 1, 1],
       [0, 0, 0, ..., 0, 0, 0],
       [1, 0, 1, ..., 1, 0, 0],
       ...,
       [1, 0, 0, ..., 0, 1, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 1, 0, ..., 1, 0, 1]])
```

Bootstrapped this many times

We can re-run our poll on 100 people... 1,000 times!

Bootstrapping the # Cat "win" sums

```
poll = np.random.binomial(100, 0.28, 1000)
```

```
Output exceeds the size limit. Open the full output data in a text editor

array([24, 26, 24, 35, 27, 35, 27, 26, 26, 25, 28, 26, 30, 31, 30, 34, 35,
       34, 22, 34, 26, 28, 34, 24, 19, 29, 23, 35, 25, 30, 40, 26, 25, 27,
       22, 22, 35, 29, 25, 25, 26, 35, 29, 35, 26, 31, 27, 26, 33, 21, 30,
       27, 36, 29, 24, 26, 28, 26, 25, 32, 27, 22, 33, 29, 26, 29, 26, 33,
       18, 21, 30, 22, 27, 35, 29, 22, 24, 25, 37, 24, 28, 30, 26, 29, 25,
       31, 27, 27, 27, 27, 29, 25, 24, 23, 27, 30, 30, 28, 36, 23, 25, 28,
       30, 37, 27, 30, 22, 26, 31, 28, 23, 25, 30, 31, 19, 24, 28, 27, 33,
       35, 22, 28, 22, 23, 23, 23, 29, 36, 27, 29, 29, 21, 29, 29, 33, 25,
       ...       26, 22, 27, 31, 28, 25, 30, 33, 24, 28, 32, 25, 20, 34])
```

On our first run, the # people who voted for Cats out of 100 people

Bootstrapping the # Cat "win" sums

```
poll = np.random.binomial(100, 0.28, 1000)
```

```
Output exceeds the size limit. Open the full output data in a text editor

array([24, 26, 24, 35, 27, 35, 27, 26, 26, 25, 28, 26, 30, 31, 30, 34, 35,
       34, 22, 34, 26, 28, 34, 24, 19, 29, 23, 35, 25, 30, 40, 26, 25, 27,
       22, 22, 35, 29, 25, 25, 26, 35, 29, 35, 26, 31, 27, 26, 33, 21, 30,
       27, 36, 29, 24, 26, 28, 26, 25, 32, 27, 22, 33, 29, 26, 29, 26, 33,
       18, 21, 30, 22, 27, 35, 29, 22, 24, 25, 37, 24, 28, 30, 26, 29, 25,
       31, 27, 27, 27, 27, 29, 25, 24, 23, 27, 30, 30, 28, 36, 23, 25, 28,
       30, 37, 27, 30, 22, 26, 31, 28, 23, 25, 30, 31, 19, 24, 28, 27, 33,
       35, 22, 28, 22, 23, 23, 23, 29, 36, 27, 29, 29, 21, 29, 29, 33, 25,
       ...        26, 22, 27, 31, 28, 25, 30, 33, 24, 28, 32, 25, 20, 34])
```

On our second run, the # people who voted for Cats out of 100 people

Bootstrapping the # Cat "win" sums

```
poll = np.random.binomial(100, 0.28, 1000)
```

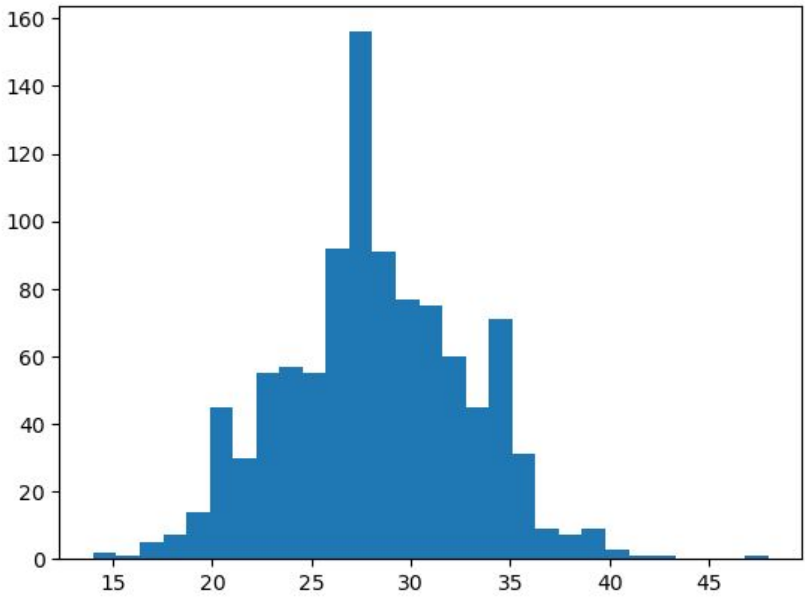
```
Output exceeds the size limit. Open the full output data in a text editor

array([24, 26, 24, 35, 27, 35, 27, 26, 26, 25, 28, 26, 30, 31, 30, 34, 35,
       34, 22, 34, 26, 28, 34, 24, 19, 29, 23, 35, 25, 30, 40, 26, 25, 27,
       22, 22, 35, 29, 25, 25, 26, 35, 29, 35, 26, 31, 27, 26, 33, 21, 30,
       27, 36, 29, 24, 26, 28, 26, 25, 32, 27, 22, 33, 29, 26, 29, 26, 33,
       18, 21, 30, 22, 27, 35, 29, 22, 24, 25, 37, 24, 28, 30, 26, 29, 25,
       31, 27, 27, 27, 27, 29, 25, 24, 23, 27, 30, 30, 28, 36, 23, 25, 28,
       30, 37, 27, 30, 22, 26, 31, 28, 23, 25, 30, 31, 19, 24, 28, 27, 33,
       35, 22, 28, 22, 23, 23, 23, 29, 36, 27, 29, 29, 21, 29, 29, 33, 25,
       ...       26, 22, 27, 31, 28, 25, 30, 33, 24, 28, 32, 25, 20, 34])
```

We do 1,000 runs

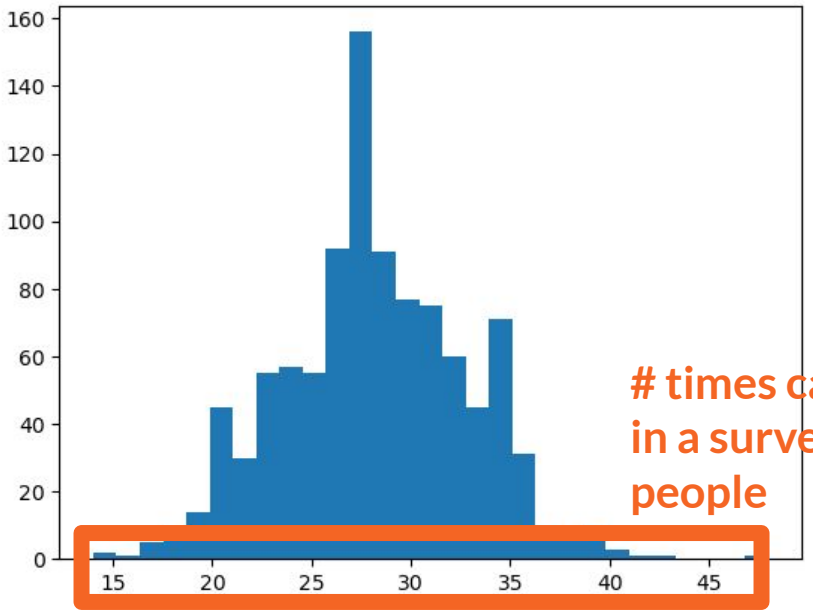
Plotting those 1,000 sums

```
plt.hist(poll,bins='auto')
plt.show()
```



Plotting those 1,000 sums

```
plt.hist(poll,bins='auto')
plt.show()
```

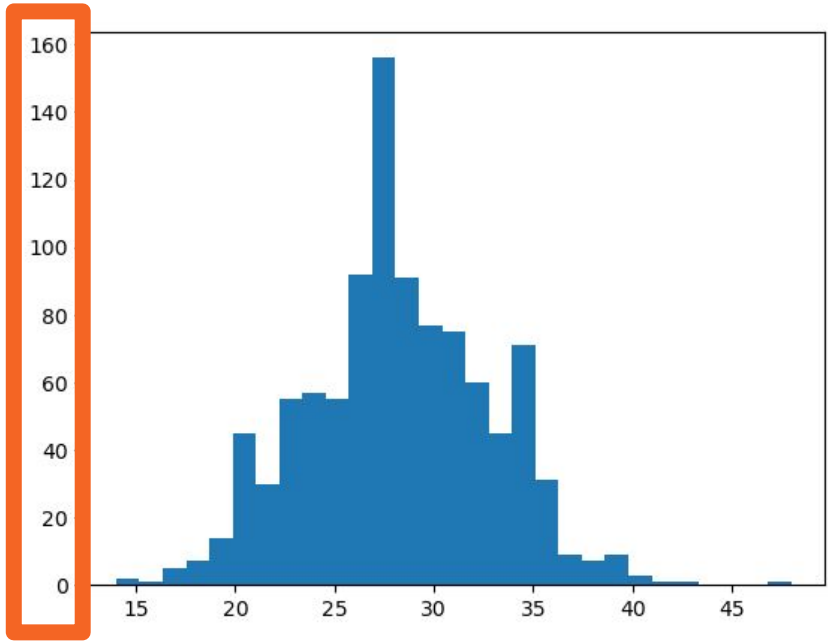


times cats “won” in a survey of 100 people

Plotting those 1,000 sums

```
plt.hist(poll,bins='auto')
plt.show()
```

times [# cats won in 100-person survey] happened within 1,000 trials



Bootstrapping the # Cat “win” sums

```
np.random.binomial(100, 0.28, 1000)
print(poll.mean())
27.97
```

Bootstrap more → closer to 'truth'

```
np.random.binomial(100, 0.28, 1000)
print(poll.mean())
27.97
```

```
np.random.binomial(100, 0.28, 1000000)
print(poll.mean())
27.99
```

(Behind the scenes math: Central Limit Theorem!)

The bootstrap confidence interval

```
poll = np.random.binomial(100, 0.28, 1000)
np.percentile(poll,[10,90])
```

find the values at the 10th and 90th percentile of sums

The bootstrap confidence interval

```
poll = np.random.binomial(100, 0.28, 1000)
np.percentile(poll,[10,90])
array([22., 34.])
```

The bootstrap confidence interval

```
poll = np.random.binomial(100, 0.28, 1000)
np.percentile(poll,[10,90])
array([22., 34.])
```

80% of the time +/- 6ish

The bootstrap confidence interval

```
poll = np.random.binomial(100, 0.28, 1000)
np.percentile(poll,[10,90])
array([22., 34.])
```

80% of the time, +/- 6ish
(relative to ~28)

The bootstrap confidence interval

```
poll = np.random.binomial(100, 0.28, 1000)
np.percentile(poll,[10,90])
array([22., 34.])
```

80% of the time, +/- 6ish

This is called the “margin of error”

The bootstrap confidence interval

```
poll = np.random.binomial(100, 0.28, 1000)
np.percentile(poll,[2.5,97.5])
array([19., 37.])
```

__% of the time, +/- __ish

The bootstrap confidence interval

```
poll = np.random.binomial(100, 0.28, 1000)
np.percentile(poll,[2.5,97.5])
array([19., 37.])
```

95% of the time, +/- 9ish

to be more confident (relative [10,90]) we need to *widen* our confidence interval

The bootstrap confidence interval

```
poll = np.random.binomial(100, 0.28, 1000)
np.percentile(poll,[2.5,97.5])
array([19., 37.])
```

95% of the time, +/- 9ish

The 95% CI for Margin of Error is 2 times the Standard Error (the standard deviation of bootstrap distribution, poll.std() is ~4.5)

The bootstrap confidence interval

```
poll = np.random.binomial(100, 0.28, 1000)
np.percentile(poll,[5,95])
array([21., 35.])
```

__% of the time, +/- __ish

The bootstrap confidence interval

```
poll = np.random.binomial(100, 0.28, 1000)
np.percentile(poll,[5,95])
array([21., 35.])
```

90% of the time, +/- 7ish

if we don't need to be that confident (relative to [2.5,97.5]) we can *narrow* our confidence interval

The bootstrap confidence interval

```
poll = np.random.binomial(100, 0.28, 1000)
np.percentile(poll,[5,95])
array([21., 35.])
```

90% of the time, +/- 7ish

Margins of error will depend on both N and p (inputs to *poll*)

Alternative: Probability distributions

We can always use bootstrap samples, but there are certain processes that occur frequently where we can define properties of that process using a **probability distribution**.

Alternative: Probability distributions

We can always use bootstrap samples, but there are certain processes that occur frequently where we can define properties of that process using a **probability distribution**.

- Why might we prefer using a probability distribution over bootstrapping? It's **faster** and easier than bootstrapping, and there's a lot of **theory** behind it to describe important data qualities.

Alternative: Probability distributions

We can always use bootstrap samples, but there are certain processes that occur frequently where we can define properties of that process using a **probability distribution**.

Recognizing a probability distribution pattern in our data lets us:

- Make predictions
- Describe important properties
- Assess whether a pattern actually holds, or if something more interesting is happening

1 min attendance break

tinyurl.com/yt9spfcf