INFO 2950: Intro to Data Science

Lecture 17
2023-10-25

Agenda

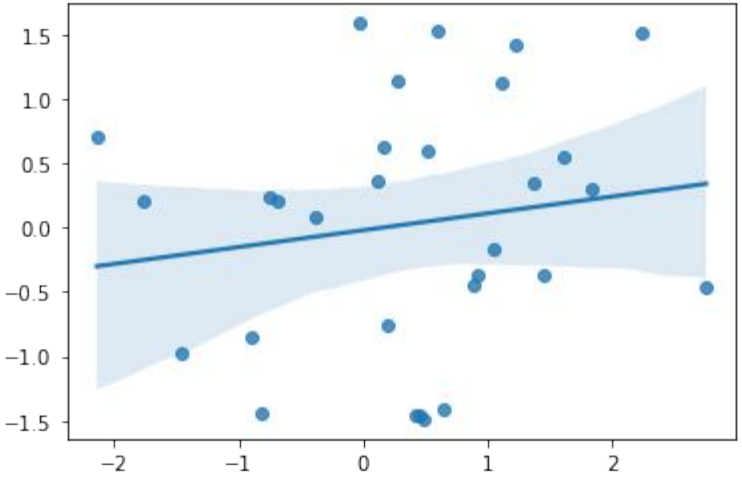
1. Bootstrap vs Permutation
2. Hypothesis testing with regressions
3. Reading regression tables for significance

Did we find something?

The first part of the class was about tools for finding patterns. But they will find patterns even in data that is actually random!

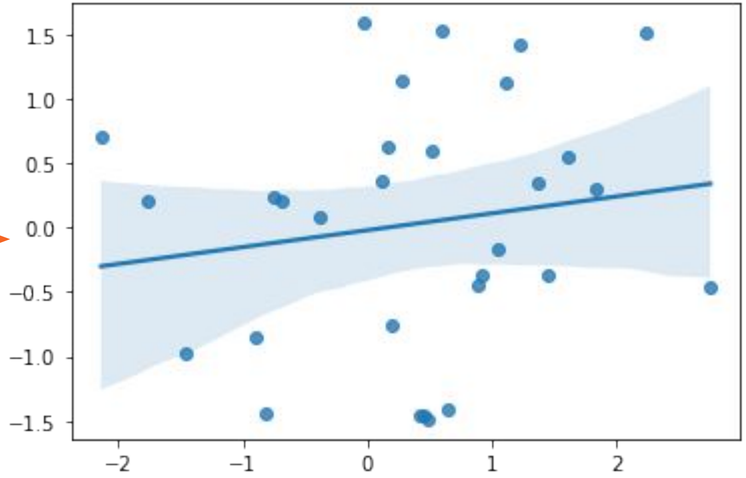
In order to say that a pattern is outside the bounds of randomness, we need to be precise about what *could* be possible through randomness.

Do you trust this regression?



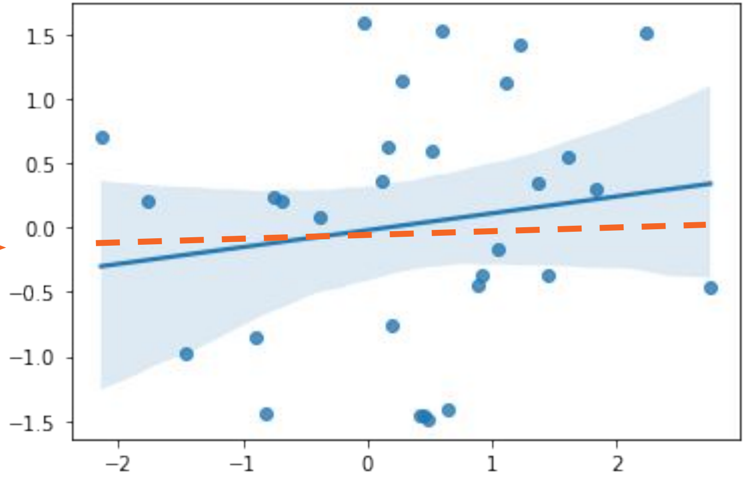
Do you trust this regression?

How confident are you that the slope β is not equal to zero?



Do you trust this regression?

The bootstrap confidence interval says that *similar* datasets could easily have slope = 0



What are we saying?

Null (boring) hypothesis:

X and Y have no relationship

Spooky (alternative) hypothesis:

There ***is*** a linear relationship between X and Y

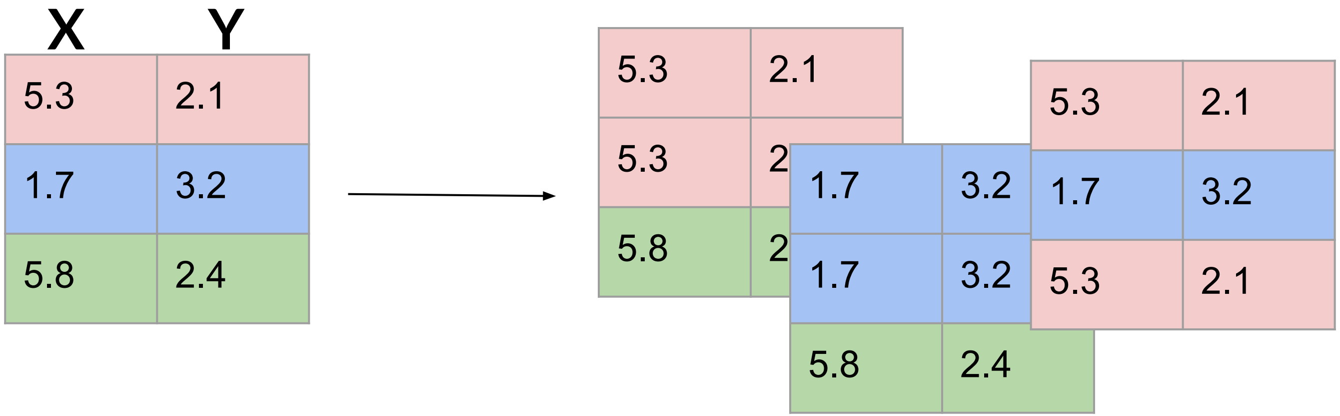
(i.e., the slope of Y ~ X is not equal to 0)

Alternative: permutation test

- If there was no connection between X and Y, what is the chance we would get a slope as large as the one we actually observed?

- Insight: we can simulate **no connection between X and Y** by shuffling the order of X values!

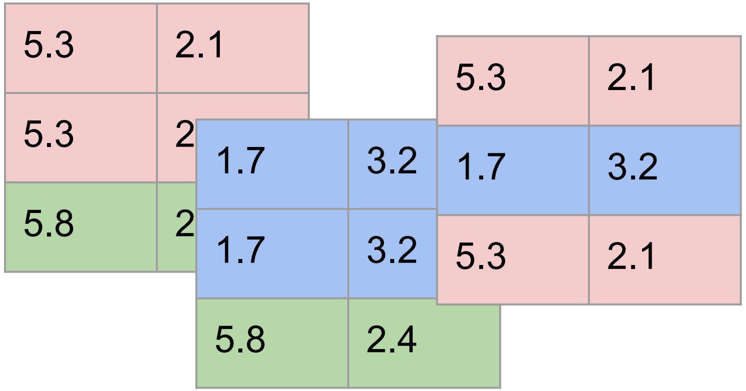
Bootstrap



Refresher: what are the two key components of bootstrapped samples?

Bootstrap

X	Y
5.3	2.1
1.7	3.2
5.8	2.4

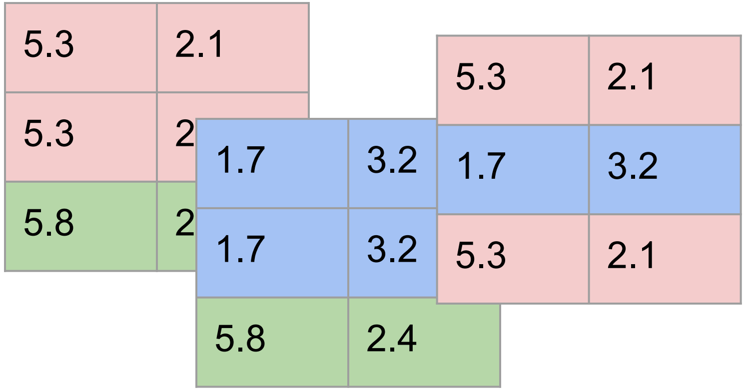


Refresher: what are the two key components of bootstrapped samples?

1. *N* stays the same (3 rows in each sample)
2. Sampling is **with replacement** (rows can be repeated)

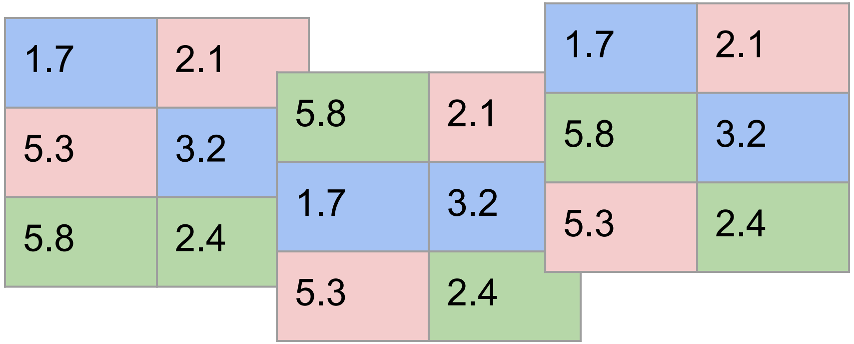
Bootstrap

X	Y
5.3	2.1
1.7	3.2
5.8	2.4



Permutation

X	Y
5.3	2.1
1.7	3.2
5.8	2.4



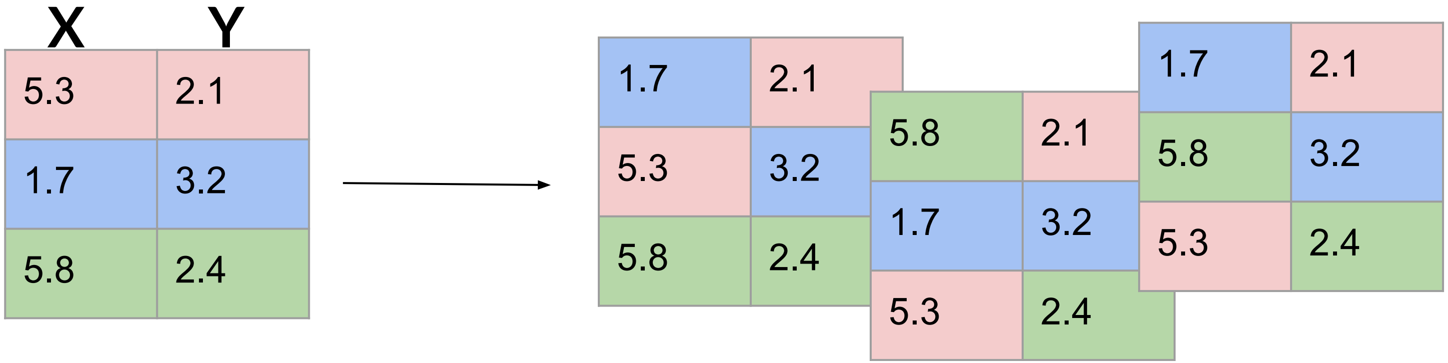
Are these 2 components true for permutations too?

One is the same, one is different:

1. In permutations, *N* is still the same (3 rows in each sample)

2. But, sampling is different: now, we randomly shuffle the order of rows *for each column, without replacement*

Permutation



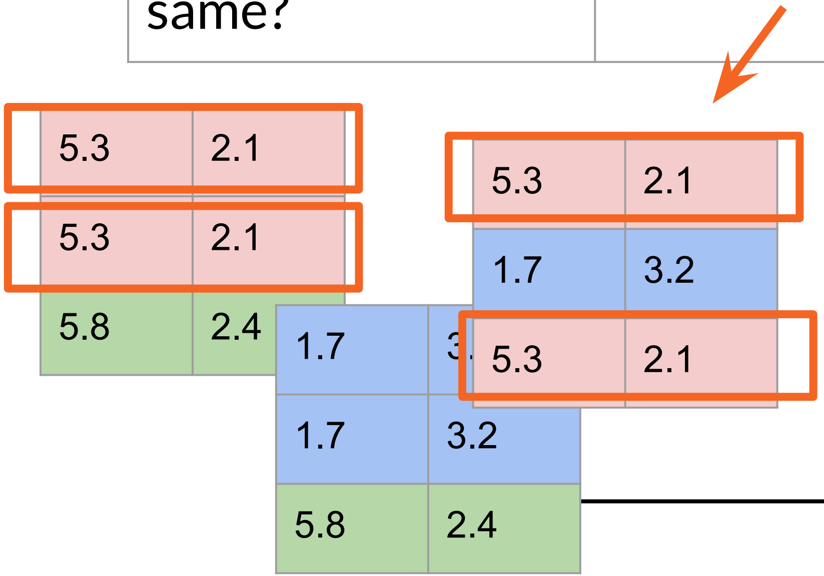
Bootstrap vs. Permutation

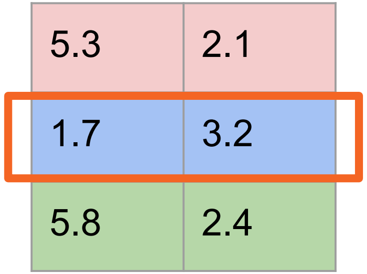
	Bootstrap	**Permutation**
What stays the same?	Rows are preserved	Column distributions are preserved

5.3	2.1
1.7	3.2
5.8	2.4

Bootstrap vs. Permutation

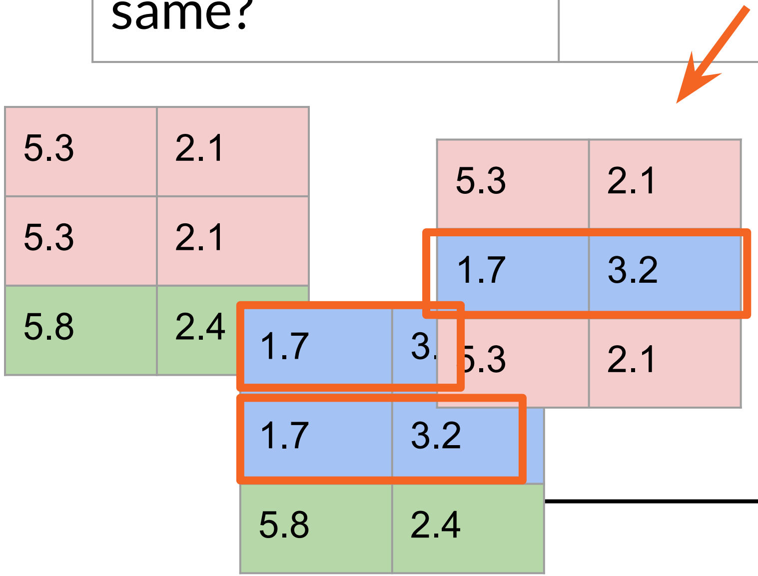
	Bootstrap	Permutation
What stays the same?	Rows are preserved	Column distributions are preserved

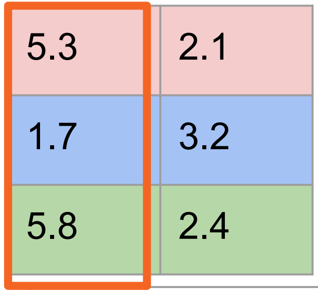




Bootstrap vs. Permutation

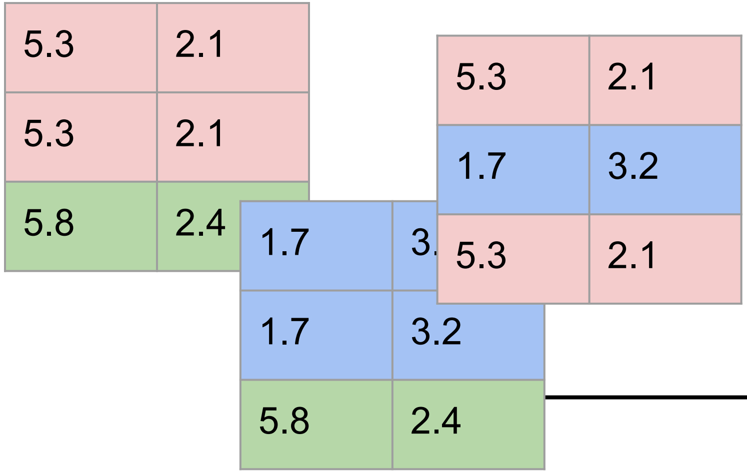
	Bootstrap	Permutation
What stays the same?	Rows are preserved	Column distributions are preserved

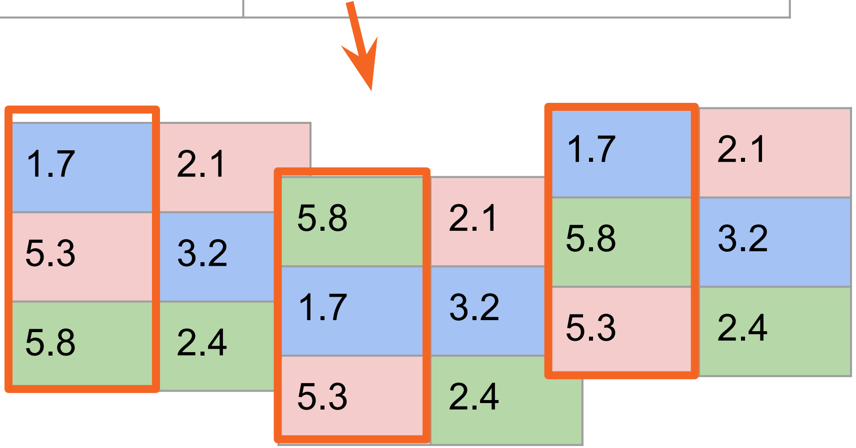


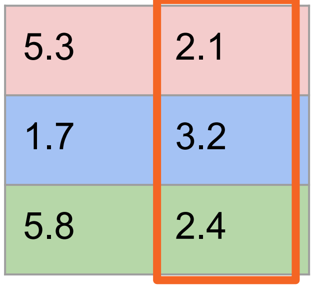


Bootstrap vs. Permutation

	Bootstrap	Permutation
What stays the same?	Rows are preserved	Column distributions are preserved

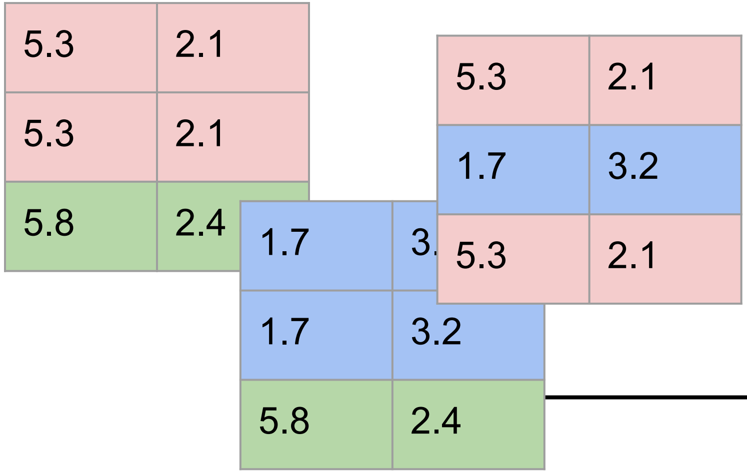


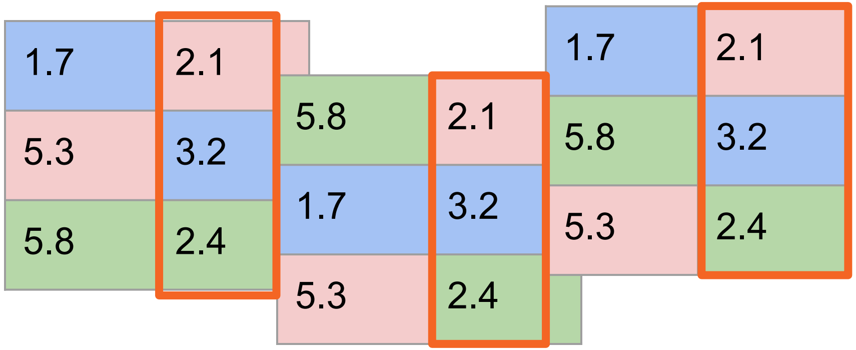




Bootstrap vs. Permutation

	Bootstrap	Permutation
What stays the same?	Rows are preserved	Column distributions are preserved





Bootstrap vs. Permutation

	Bootstrap	**Permutation**
What stays the same?	Rows are preserved	Column distributions are preserved
What is different?	**Rows may be repeated or removed**	**Order of values in columns are random**

Bootstrap vs. Permutation

	Bootstrap	Permutation
What stays the same?	Rows are preserved	Column distributions are preserved
What is different?	**Rows may be repeated or removed**	**Order of values in columns are random**

(with replacement) (without replacement)

Bootstrap vs. Permutation

	Bootstrap	**Permutation**
What stays the same?	Rows are preserved	Column distributions are preserved
What is different?	Rows may be repeated or removed	Order of values in columns are random
What question are we asking?	**What if we had a similar but not exactly identical dataset?**	**What if there were no connection between columns?**

Bootstrap vs. Permutation

Original data: there's some relationship between month and temperature

Month	Temp
Aug	90
Oct	50
Dec	25

	Bootstrap	Permutation
What question are we asking?	**What if we had a similar but not exactly identical dataset?**	**What if there were no connection between columns?**

Bootstrap vs. Permutation

Month	Temp
Aug	90
Oct	50
Dec	25

Month	Temp
Aug	50
Oct	25
Dec	90

Permuted data: there is likely no longer some relationship between month and temperature

	Bootstrap	Permutation
What question are we asking?	**What if we had a similar but not exactly identical dataset?**	**What if there were no connection between columns?**

Bootstrap vs. Permutation

	Bootstrap	**Permutation**
What stays the same?	Rows are preserved	Column distributions are preserved
What is different?	Rows may be repeated or removed	Order of values in columns are random
What question are we asking?	**What if we had a similar but not exactly identical dataset?**	**What if there were no connection between columns?**

Permutation and regression

If you have one column with 5 rows, how many permutations (distinct orderings) can you make?

Permutation and regression

If you have one column with 5 rows, how many permutations (distinct orderings) can you make?

5! = 5 * 4 * 3 * 2 * 1 = 120

Permutation and regression

X	Y
1	20
2	30
3	50
4	10
5	40

If you had 5 X values and 5 Y values, how could you permute the values within each column to create the largest regression slope?

Permutation and regression

X	Y
1	10
2	20
3	30
4	40
5	50

If you had 5 X values and 5 Y values, how could you permute the values within each column to create the largest regression slope?

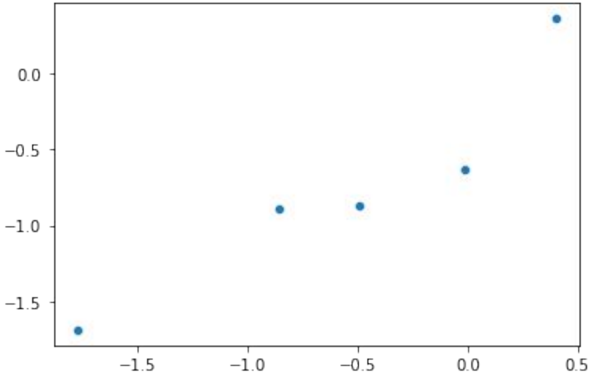
In general: sort both arrays. Pair the lowest X with the lowest Y, and so forth until you get to the highest X and the highest Y

Sample some random points

```
X = np.random.normal(0, 1, 5)
Y = np.random.normal(0, 1, 5)
```

Generate 5 normally-distributed X values (mean 0 stdev 1), and same for Y

```
seaborn.scatterplot(x=X, y=Y)
```

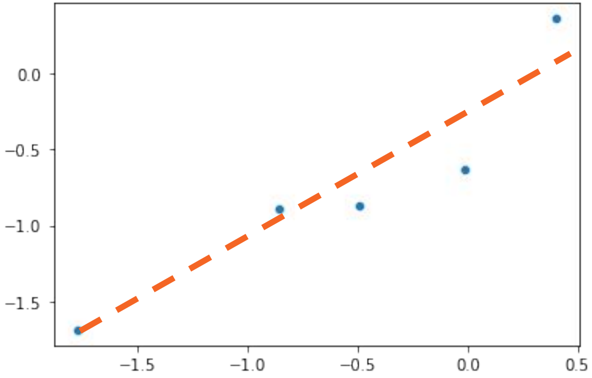


Calculate regression slope

```
df = pd.DataFrame({"X": X, "Y": Y})
model = LinearRegression().fit(df[["X"]], df["Y"])
model.coef_

array([0.82531542])
```

Calculate regression slope

```
df = pd.DataFrame({"X": X, "Y": Y})
model = LinearRegression().fit(df[["X"]], df["Y"])
model.coef_
array([0.82531542])
```



For these X and Y values, our regression slope is 0.825

Create permutations

```
x_permutations = list(itertools.permutations(X))
```

Now let's permute our X values!
(No need to permute both columns)

Create permutations

```
x_permutations = list(itertools.permutations(X))
len(x_permutations), 5 * 4 * 3 * 2 * 1
```

```
(120, 120)
```

Double check that x_permutations is the right size

Calculate regressions for all permutations

```
permutation_slopes = np.zeros(120)

for i, x_permutation in enumerate(x_permutations):
  df = pd.DataFrame({"X": x_permutation, "Y": Y})
  model = LinearRegression().fit(df[["X"]], df["Y"])
  permutation_slopes[i] = model.coef_[0]
```

For each permutation, run a regression and record the slope

How does our actual value compare?

```
permutation_slopes = sorted(permutation_slopes)
```

These are the 120 slopes we calculated from Y ~ permuted X values.

Let's sort them low to high.

How does our actual value compare?

```
permutation_slopes = sorted(permutation_slopes)
permutation_slopes[:5], permutation_slopes[-5:]
```

Now let’s print the smallest 5 and largest 5 slopes we got among our permutations...

How does our actual value compare?

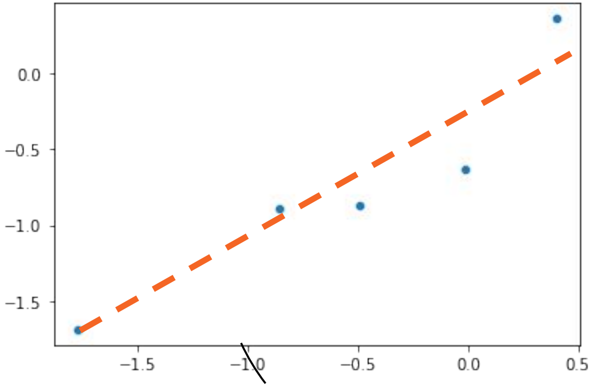
```
permutation_slopes = sorted(permutation_slopes)
permutation_slopes[:5], permutation_slopes[-5:]
```

Now let's print the smallest 5 and largest 5 slopes we got among our permutations...

```
([-0.848, -0.844, -0.819, -0.812, -0.775],
 [0.752, 0.778, 0.785, 0.822, 0.825])
```

Calculate regression slope

```
df = pd.DataFrame({"X": X, "Y": Y})
model = LinearRegression().fit(df[["X"]], df["Y"])
model.coef_
array([0.82531542])
```



For these X and Y values, our regression slope is 0.825

How does our actual value compare?

```
permutation_slopes = sorted(permutation_slopes)
permutation_slopes[:5], permutation_slopes[-5:]

([-0.848, -0.844, -0.819, -0.812, -0.775],
 [0.752, 0.778, 0.785, 0.822, 0.825])
```

We had the actual largest possible β!

How does our actual value compare?

```
permutation_slopes = sorted(permutation_slopes)
permutation_slopes[:5], permutation_slopes[-5:]
```

```
([-0.848, -0.844, -0.819, -0.812, -0.775],
 [0.752, 0.778, 0.785, 0.822, 0.825])
```

What is the probability we would have gotten 0.825 by chance?

Hint: what is len(permutation_slopes)?

How does our actual value compare?

```
permutation_slopes = sorted(permutation_slopes)
permutation_slopes[:5], permutation_slopes[-5:]

([-0.848, -0.844, -0.819, -0.812, -0.775],
 [0.752, 0.778, 0.785, 0.822, 0.825])
```

What is the probability we would have gotten 0.825 by chance?

1/120

How does our actual value compare?

```
permutation_slopes = sorted(permutation_slopes)
permutation_slopes[:5], permutation_slopes[-5:]

([-0.848, -0.844, -0.819, -0.812, -0.775],
 [0.752, 0.778, 0.785, 0.822, 0.825])
```

What is the probability that β > 0.8 purely by chance?

One-sided test

```
permutation_slopes = sorted(permutation_slopes)
permutation_slopes[:5], permutation_slopes[-5:]
```

```
([-0.848, -0.844, -0.819, -0.812, -0.775],
 [0.752, 0.778, 0.785, 0.822, 0.825])
```

What is the probability that β > 0.8 purely by chance? 2/120

How does our actual value compare?

```
permutation_slopes = sorted(permutation_slopes)
permutation_slopes[:5], permutation_slopes[-5:]
```

```
([-0.848, -0.844, -0.819, -0.812, -0.775],
 [0.752, 0.778, 0.785, 0.822, 0.825])
```

What is the probability that $|\beta| > 0.8$ purely by chance?

Two-sided test

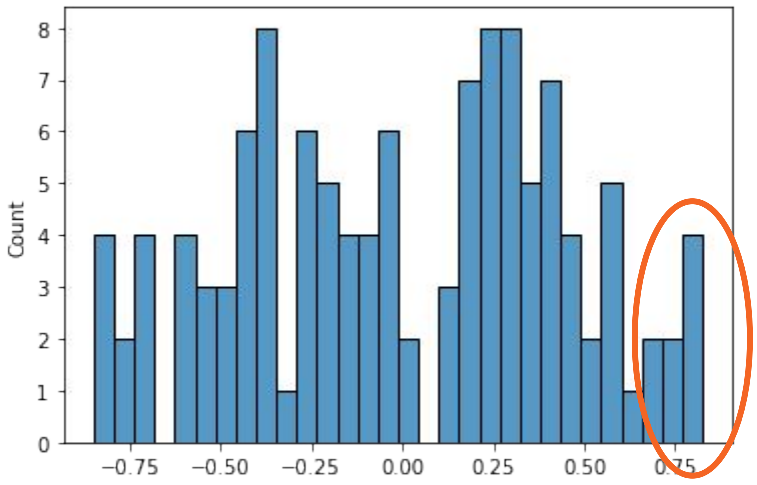
```
permutation_slopes = sorted(permutation_slopes)
permutation_slopes[:5], permutation_slopes[-5:]
```

```
([-0.848, -0.844, -0.819, -0.812, -0.775],
 [0.752, 0.778, 0.785, 0.822, 0.825])
```

What is the probability that
|β| > 0.8 purely by chance?
6/120

Display permutation slopes

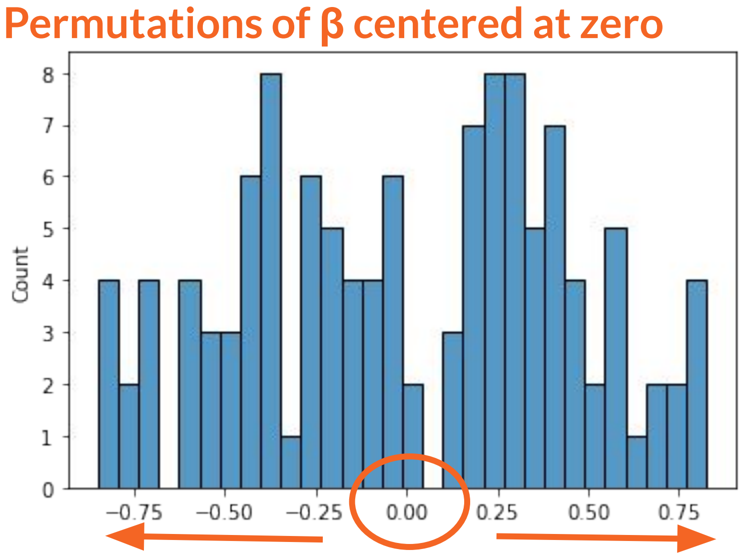
```
seaborn.histplot(permutation_slopes, bins=30)
```



Originally, we had the actual largest possible β by sheer luck!

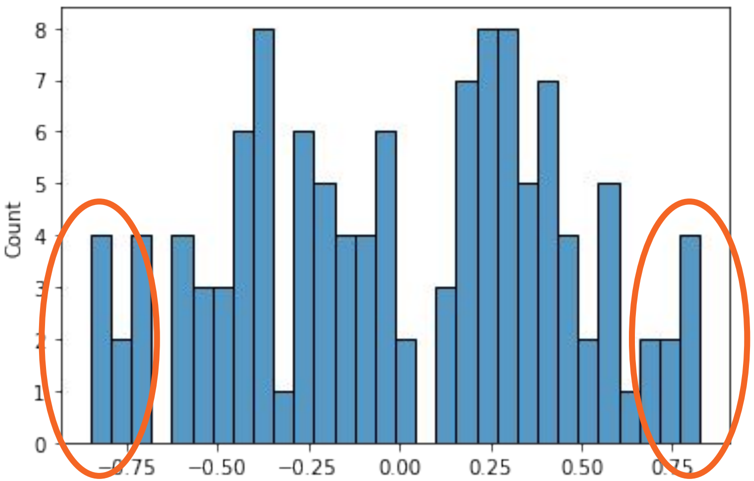
Display permutation slopes

```
seaborn.histplot(permutation_slopes, bins=30)
```



Display permutation slopes

```
seaborn.histplot(permutation_slopes, bins=30)
```



A two-sided test would look in both directions

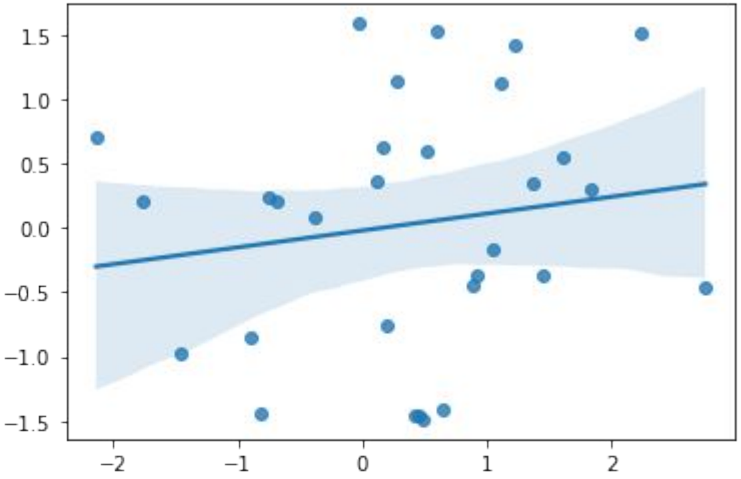
Sample some random points

```
X_30 = np.random.normal(0, 1, 30)
Y_30 = np.random.normal(0, 1, 30)
```

Let’s try again with more simulated data (30 instead of 5 points)

Sample some random points

```
X_30 = np.random.normal(0, 1, 30)
Y_30 = np.random.normal(0, 1, 30)
seaborn.regplot(x=X_30, y=Y_30)
```



regplot in seaborn allows you to automatically plot the regression line with your X's and Y's

Calculate regression slope

```
df = pd.DataFrame({"X": X_30, "Y": Y_30})
model = LinearRegression().fit(df[["X"]], df["Y"])
model.coef_
```

```
array([0.13104837])
```

Calculate regression slope

```
df = pd.DataFrame({"X": X_30, "Y": Y_30})
model = LinearRegression().fit(df[["X"]], df["Y"])
model.coef_
```

```
array([0.13104837])
```

How many permutations of 30 rows can we make? (express using the factorial sign)

Create permutations

```
factorial(30)
```

There are 30! permutations of 30 rows

```
2.652528598121911e+32
```

Create permutations

```
factorial(30)
```

```
2.652528598121911e+32
```

Nope, way too many for our computers!

Calculate regressions for ~~all~~ a lot of permutations

```
permutation_slopes = np.zeros(1000)

for i in range(1000):
  permuted_X = np.random.choice(X_30, 30, replace=False)
  df = pd.DataFrame({"X": permuted_X, "Y": Y_30})
  model = LinearRegression().fit(df[["X"]], df["Y"])
  permutation_slopes[i] = model.coef_[0]
```

Calculate regressions for ~~all~~ a lot of permutations

Shuffling without replacement = permutations

```
permutation_slopes = np.zeros(1000)

for i in range(1000):
  permuted_X = np.random.choice(X_30, 30, replace=False)
  df = pd.DataFrame({"X": permuted_X, "Y": Y_30})
  model = LinearRegression().fit(df[["X"]], df["Y"])
  permutation_slopes[i] = model.coef_[0]
```

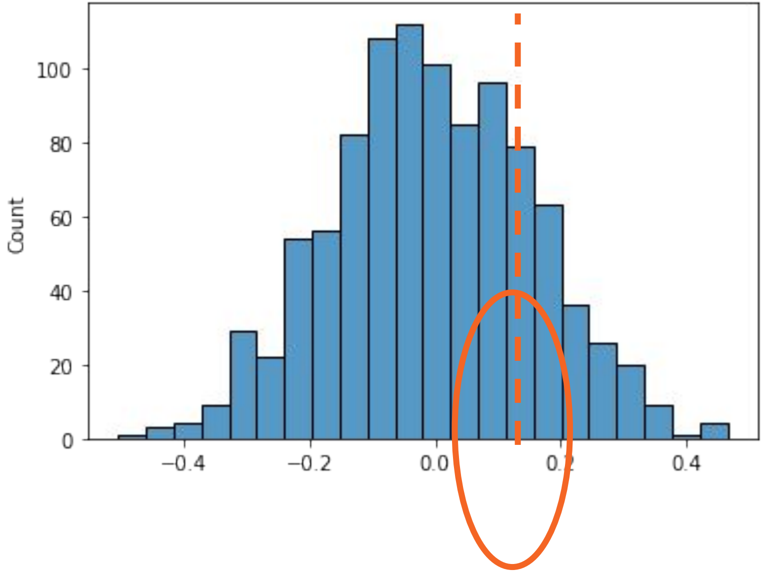
Calculate regressions for ~~all~~ a lot of permutations

```
permutation_slopes = np.zeros(1000)

for i in range(1000):
  permuted_X = np.random.choice(X_30, 30, replace=False)
  df = pd.DataFrame({"X": permuted_X, "Y": Y_30})
  model = LinearRegression().fit(df[["X"]], df["Y"])
  permutation_slopes[i] = model.coef_[0]
```

Storing each of our 1,000 permuted slopes into *permutation_slopes*

Display permutation slopes

```
seaborn.histplot(permutation_slopes, bins=30)
```

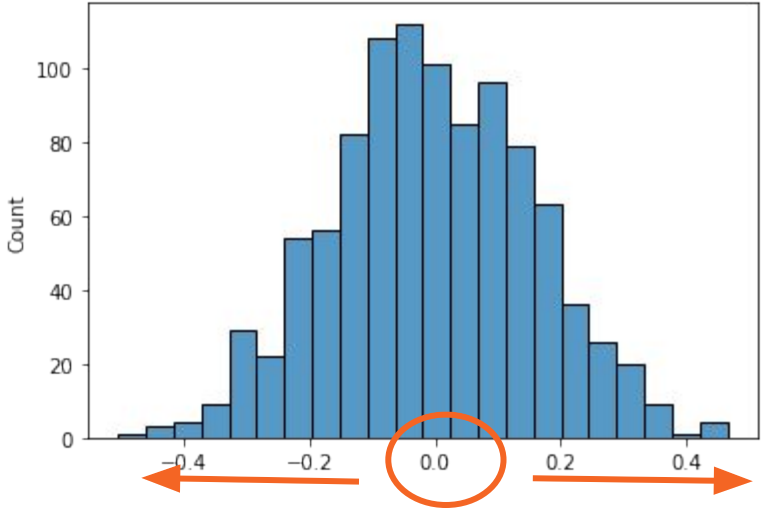


Our original β=0.13 is not at all unusual!

Display permutation slopes

```
seaborn.histplot(permutation_slopes, bins=30)
```

Permutations of β still centered at zero



Bootstrap vs. Permutation for β

	Bootstrap	Permutation
What is the average?	Same as original data	0
What question are we asking?	What is a confidence interval around the estimated value?	What is a confidence interval around the null hypothesis?
What are we looking for?	Does the confidence interval include 0?	Does the confidence interval include the original data value?

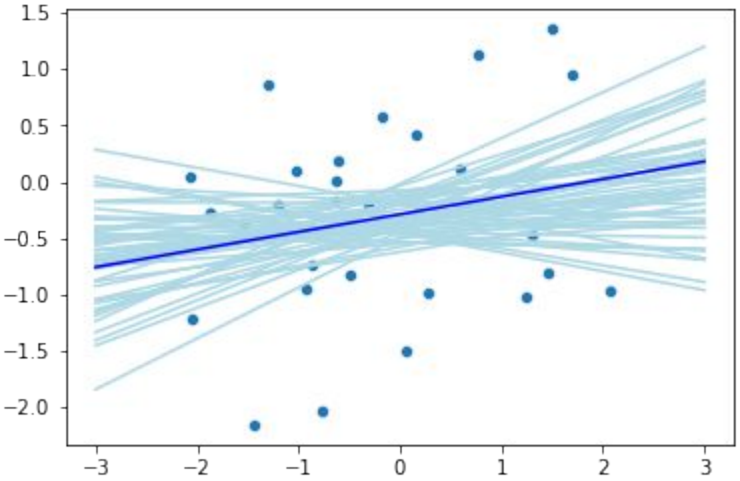
Bootstrap vs. Permutation for β

	Bootstrap	**Permutation**
Best for…	Estimating Confidence Intervals	Testing Hypotheses

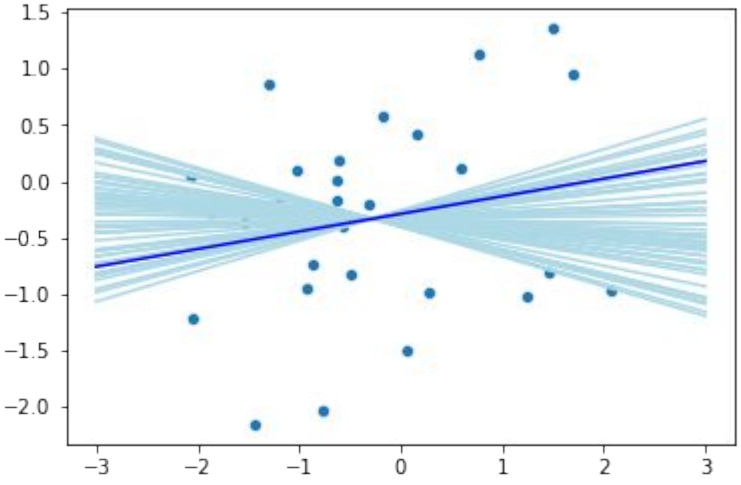
```python
seaborn.scatterplot(x=X_30, y=Y_30)

domain = np.array([-3,3])

original_df = pd.DataFrame({"X": X_30, "Y": Y_30})

for i in range(50):

  df = original_df.sample(30, replace=True)
  model = LinearRegression().fit(df[["X"]], df["Y"])
  y_pred = domain * model.coef_[0] +  model.intercept_
  seaborn.lineplot(x=domain, y=y_pred, color="lightblue")

df = pd.DataFrame({"X": X_30, "Y": Y_30})
model = LinearRegression().fit(df[["X"]], df["Y"])
y_pred = domain * model.coef_[0] +  model.intercept_
seaborn.lineplot(x=domain, y=y_pred, color="blue")
pyplot.show()
```

Bootstrap test Resampling may change (mean X, mean Y), so lines don't all pass through the same point.

Confidence region is centered around original slope



```
seaborn.scatterplot(x=X_30, y=Y_30)
domain = np.array([-3,3])

for i in range(50):
  df = pd.DataFrame({"X": np.random.choice(X_30, 30, replace=False), "Y": Y_30})
  model = LinearRegression().fit(df[["X"]], df["Y"])
  y_pred = domain * model.coef_[0] +  model.intercept_
  seaborn.lineplot(x=domain, y=y_pred, color="lightblue")

df = pd.DataFrame({"X": X_30, "Y": Y_30})
model = LinearRegression().fit(df[["X"]], df["Y"])
y_pred = domain * model.coef_[0] +  model.intercept_
seaborn.lineplot(x=domain, y=y_pred, color="blue")
pyplot.show()
```



Permutation test All regression lines go through the point at (mean X, mean Y). Permutation doesn't change these means.

Confidence region is centered around 0 slope (not Y=0!)

1 min break

https://dartbrains.org/_downloads/3e7f0b078bb38ceb305cb965867402f9/Relatability_Presentation.pdf

Hypothesis testing for a linear model

- $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$
- Have you ever heard people talk about whether their regression coefficients are "significant"?

https://vitalflux.com/linear-regression-hypothesis-testing-examples/

Hypothesis testing for a linear model

- $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$
- Have you ever heard people talk about whether their regression coefficients are “significant”?

```
Call:
lm(formula = log(medv) ~ crim + chas + rad + lstat)

Residuals:
     Min       1Q   Median       3Q      Max
-0.77765 -0.14342 -0.02525  0.10632  0.88673

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.5728575  0.0220849 161.778  < 2e-16 ***
crim        -0.0090038  0.0015482  -5.816 1.08e-08 ***
chas1        0.1766052  0.0399708   4.418 1.22e-05 ***
rad         -0.0008834  0.0015599  -0.566    0.571
lstat       -0.0402739  0.0016669 -24.161  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2274 on 501 degrees of freedom
Multiple R-squared:  0.6929,    Adjusted R-squared:  0.6904
F-statistic: 282.6 on 4 and 501 DF,  p-value: < 2.2e-16
```

significance stars... on a β?!

https://vitalflux.com/linear-regression-hypothesis-testing-examples/

Hypothesis Testing: Regression

- **Null (boring) hypothesis:** there is no relationship between output (y) and input x_1
  - **Equivalently, β_1=0**

Hypothesis Testing: Regression

- **Null (boring) hypothesis:** there is no relationship between output (y) and input x_1
  - **Equivalently, $\boldsymbol{\beta_1=0}$**
- **Alternative (spooky) hypothesis:** there is some relationship between output(y) and a specific input (x_1)
  - **Equivalently, $\boldsymbol{\beta_1\neq0}$**

Hypothesis Testing: Regression

Model only includes intercept and error

- **Null (boring) hypothesis:** there is no relationship between output (y) and input x_1
  - **Equivalently, β_1=0**

Model includes intercept, error, and the beta estimate (from Python)

- **Alternative (spooky) hypothesis:** there is some relationship between output(y) and a specific input (x_1)
  - **Equivalently, $\beta_1 \neq 0$**

Hypothesis Testing: Regression

$y_i = \alpha + \varepsilon_i$

- **Null (boring) hypothesis:** there is no relationship between output (y) and input x_1
  - **Equivalently, $\boldsymbol{\beta_1}$=0**

$y_i = \alpha + \beta_1 x_1 + \varepsilon_i$

- **Alternative (spooky) hypothesis:** there is some relationship between output(y) and a specific input (x_1)

Hypothesis Testing: Regression

$y_i = \alpha + \varepsilon_i$

For this to work, need to assume normally distributed errors centered at 0

$y_i = \alpha + \beta_1 x_1 + \varepsilon_i$

- **Null (boring) hypothesis:** there is no relationship between output (y) and input x_1
  - **Equivalently, β_1=0**
- **Alternative (spooky) hypothesis:** there is some relationship between output(y) and a specific input (x_1)

Hypothesis Testing: Regression

- A t-test will give you a t-statistic, which you can plot against a t-distribution to evaluate spookiness

- A one-sample t-test answers:
  - If the true slope is zero, how ***unlikely*** is it that we would find a sample of data that presents evidence for a non-zero slope that it as convincing as the sample we have actually observed? (*quantified with p-value*)

https://www.reed.edu/economics/course_pages.archive/red_spots/testing_hypotheses.htm

Testing for differences

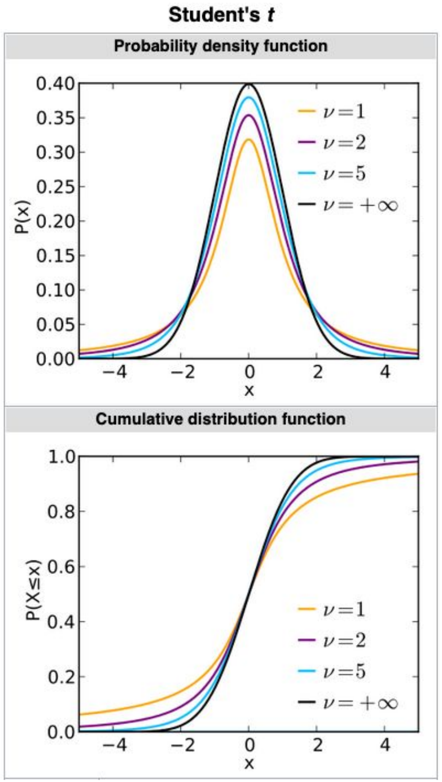
- **One Sample** t-tests: comparing a *group* to a known *value*
  - Distribution of slopes vs. your specific slope
  - WTA players' distribution vs. Serena William's # Aces
- **Two Sample** t-tests: comparing two *groups* to each other
  - WTA players' distribution vs. Men's tennis players' distribution

Hypothesis Testing: Regression

- A one-sample t-test answers:
  - If the true slope is zero, how ***unlikely*** is it that we would find a sample of data that presents evidence for a non-zero slope that it as convincing as the sample we have actually observed? (*quantified with p-value*)

This sounds like what we were doing with permutations earlier!

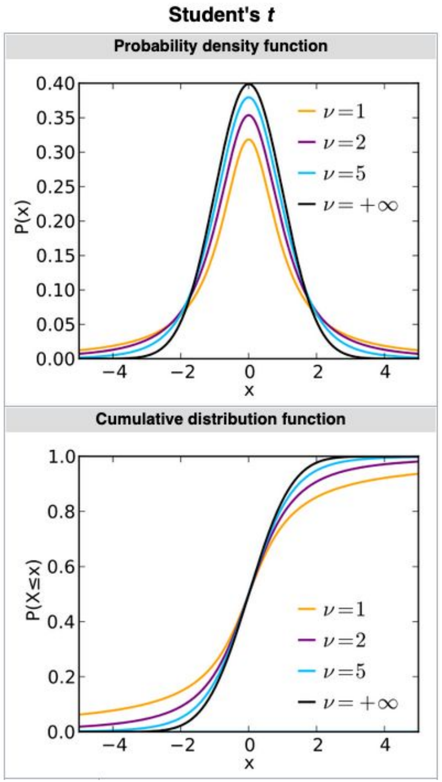
https://www.reed.edu/economics/course_pages.archive/red_spots/testing_hypotheses.htm



Permutation vs. t-test

- If we can sample permutations, why bother with a t-test?

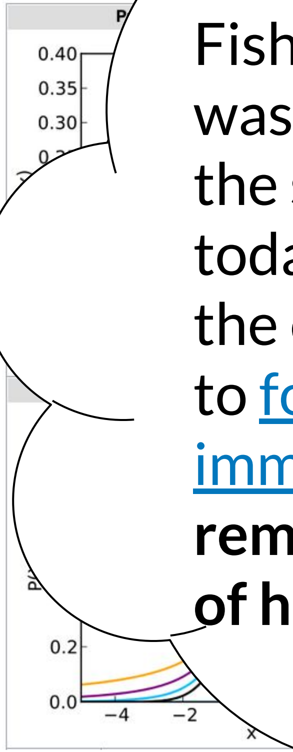
https://en.wikipedia.org/wiki/Student%27s_t-distribution



Permutation vs. t-test

- If we can sample permutations, why bother with a t-test?
- We couldn't run 1,000 permutations in 1938
- The t-test is an approximation to permutation tests
- They usually give you the same answer, and you will see t-tests frequently

https://en.wikipedia.org/wiki/Student%27s_t-distribution



Permutation is also known as Fisher's Exact Test. Ronald Fisher was brilliant and created many of the statistical tools we still use today, and was *also* a founder of the eugenics movement that led to [forced sterilizations]() and [racist immigration laws](). **We want you to remember *both* of these aspects of his legacy.**

ons, why bother

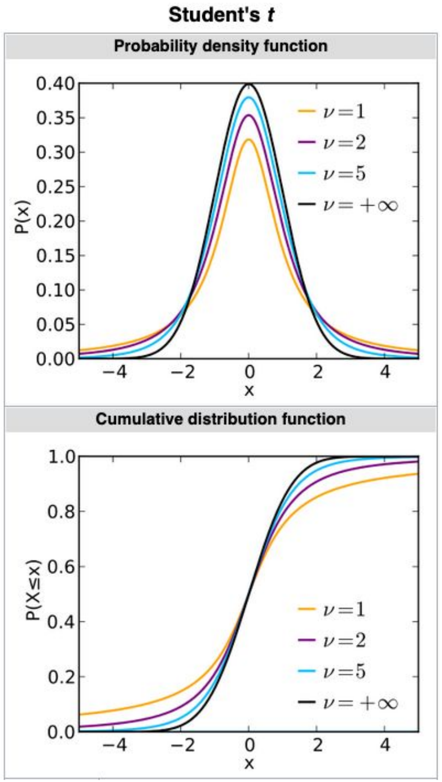
1938

ion to

u the same answer, and

sts frequently

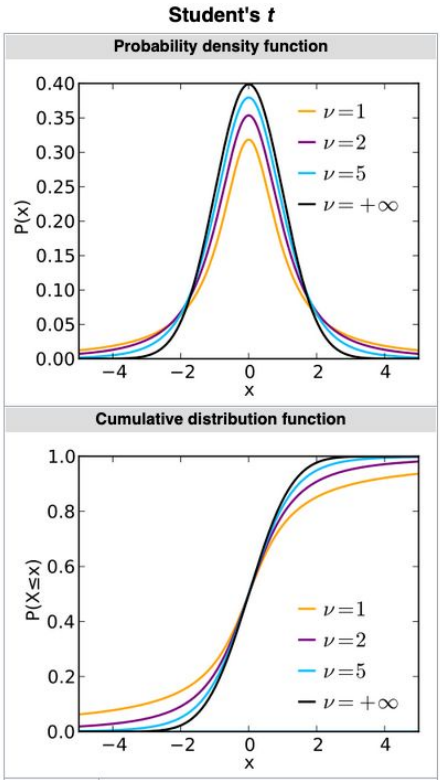
https://en.wikipedia.org/wiki/Student%27s_t-distribution



Hypothesis Testing: Regression

- t-statistic: $t = (b - \beta) / SE_b$
- Compare to t-distribution
- Decide if t spooky enough to reject null

https://en.wikipedia.org/wiki/Student%27s_t-distribution

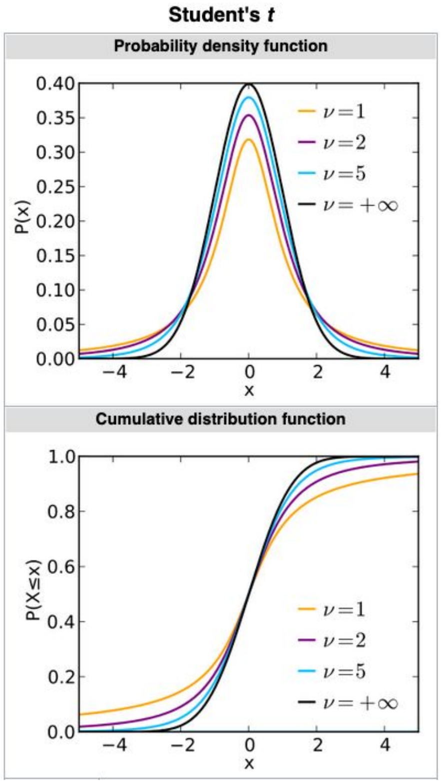


Hypothesis Testing: Regression

- t-statistic: $\boxed{t = (b - \beta) / SE_b}$
- Compare to t-distribution
- Decide if t spooky enough to reject null

Python gives you everything you need to calculate this!

https://en.wikipedia.org/wiki/Student%27s_t-distribution

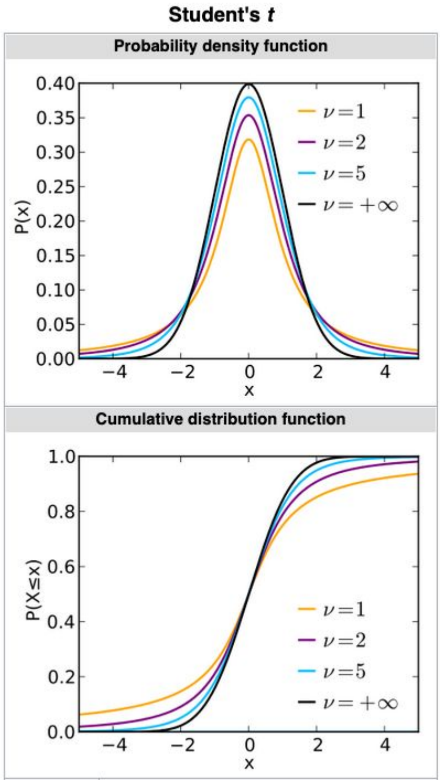


Hypothesis Testing: Regression

- t-statistic: **$t = (b - \beta) / SE_b$**
  - b = the estimated coefficient
  - SE_b = the "standard error" of the estimated coefficient
  - β = the coefficient under the null

https://en.wikipedia.org/wiki/Student%27s_t-distribution

Hypothesis Testing: Regression

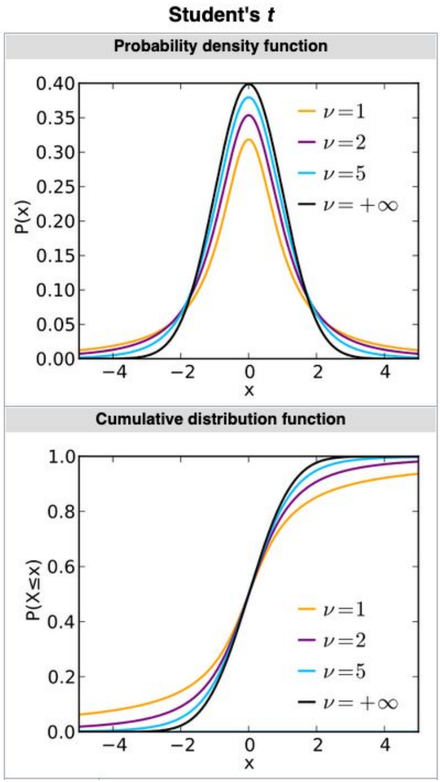


- t-statistic: **t = (b - β) / SE$_b$**
  - b = the estimated coefficient
  - SE$_b$ = the "standard error" of the estimated coefficient
  - β = the coefficient under the null

We've also called this β-hat

https://en.wikipedia.org/wiki/Student%27s_t-distribution

Hypothesis Testing: Regression

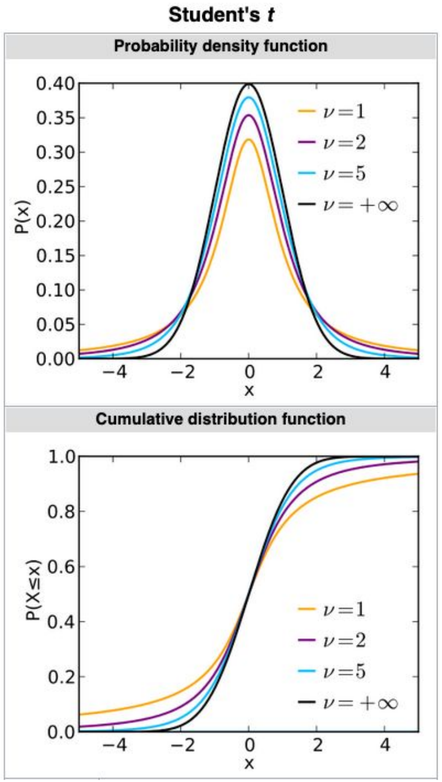


- t-statistic: **$t = (b - \beta) / SE_b$**
  - b = the estimated coefficient
  - SE_b = the "standard error" of the estimated coefficient
  - β = the coefficient under the null

We've also called this β-hat

How do we get this in Python? model.______

https://en.wikipedia.org/wiki/Student%27s_t-distribution



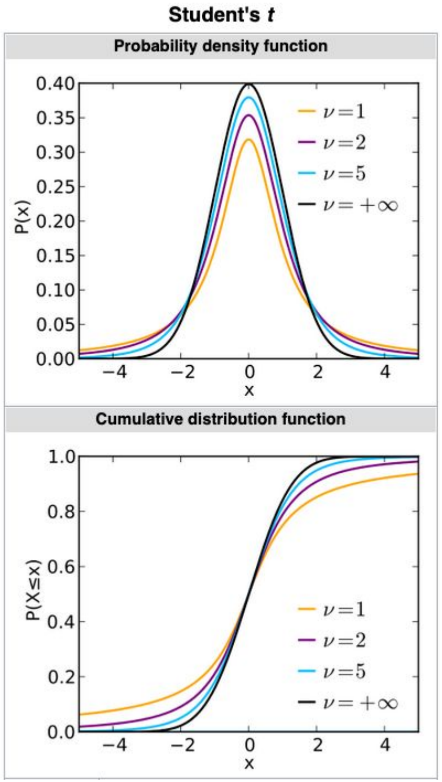
Hypothesis Testing: Regression

- t-statistic: **$t = (b - \beta) / SE_b$**
  - b = the estimated coefficient
  - SE_b = the "standard error" of the estimated coefficient
  - β = the coefficient under the null

We've also called this β-hat

How do we get this in Python? `model.coef_`

https://en.wikipedia.org/wiki/Student%27s_t-distribution

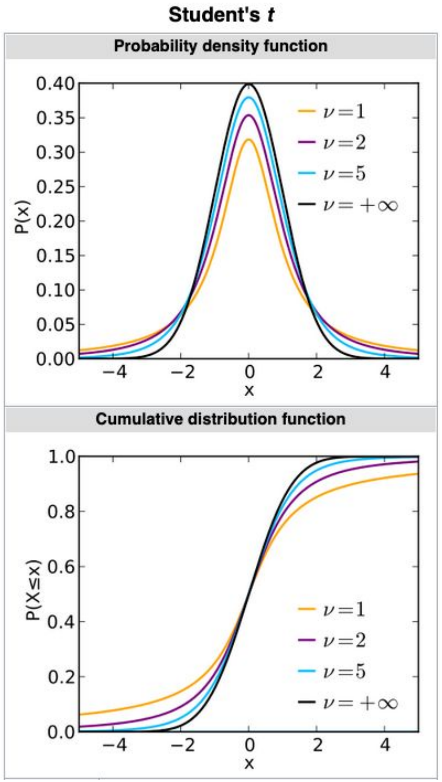


Hypothesis Testing: Regression

- t-statistic: **t = (b - β) / SE$_b$**
  - b = the estimated coefficient
  - SE$_b$ = the “standard error” of the estimated coefficient
  - β = the coefficient under the null

Standard Error is the standard deviation of the *estimates* (remember the margin of error calculations when doing opinion polls?)

https://en.wikipedia.org/wiki/Student%27s_t-distribution

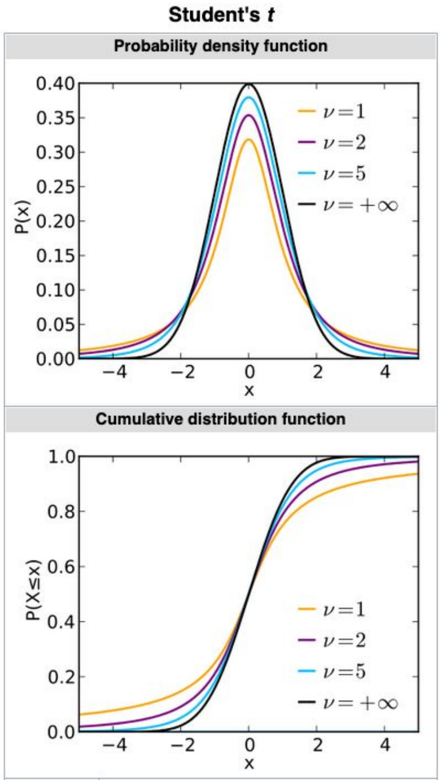


Hypothesis Testing: Regression

- t-statistic: **t = (b - β) / SE$_b$**
  - b = the estimated coefficient
  - $\boxed{\text{SE}_b \text{ = the "standard error" of the estimated coefficient}}$
  - β = the coefficient under the null

Smaller Standard Error → more precise estimate of that coefficient

https://en.wikipedia.org/wiki/Student%27s_t-distribution



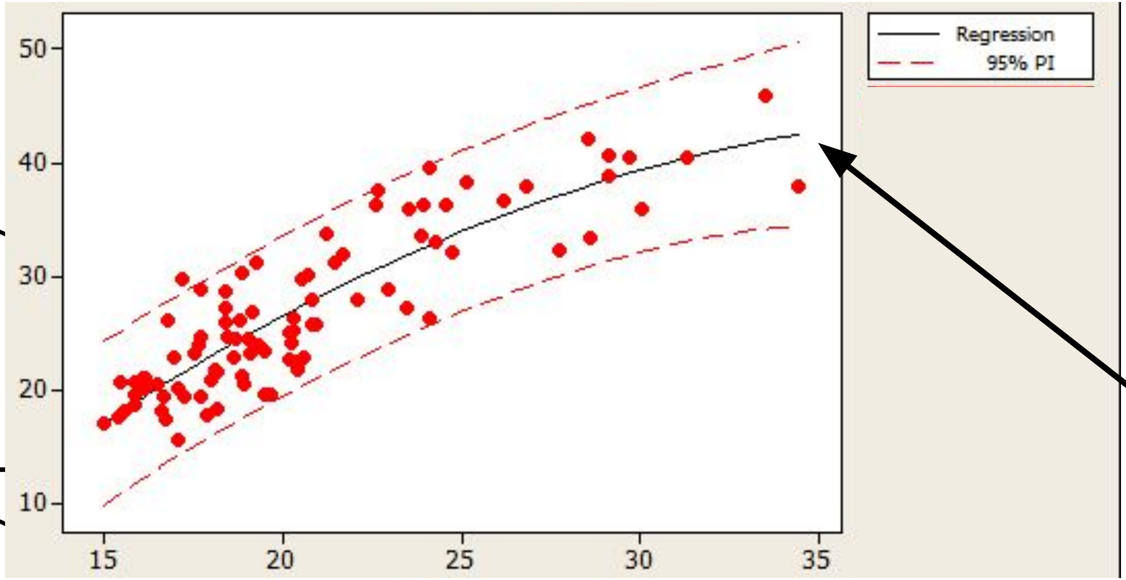
Hypothesis Testing: Regression

- t-statistic: **$t = (b - \beta) / SE_b$**
  - b = the estimated coefficient
  - SE_b = the "standard error" of the estimated coefficient
  - β = the coefficient under the null

SE is calculated for each coefficient x. Intuition: it multiplies the *SE of the regression* times a factor that's specific to that coefficient

https://en.wikipedia.org/wiki/Student%27s_t-distribution

Standard Error of the Regression



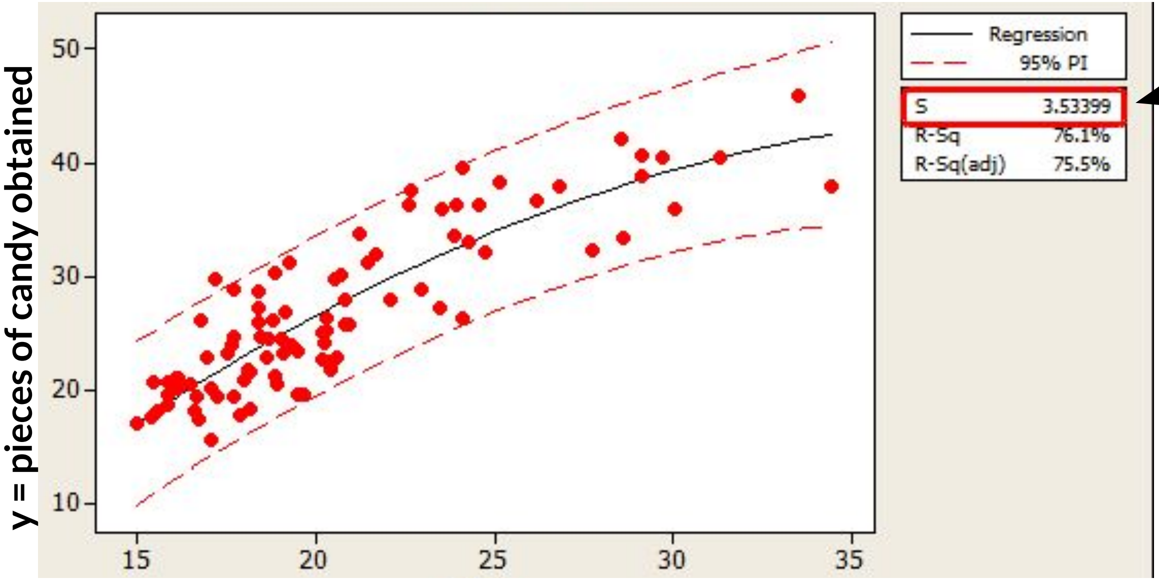
y = pieces of candy obtained

x = number of minutes trick-or-treating

Our black regression line is curvy because our model is $y = \alpha + \beta_1 x + \beta_2 x^2$

https://blog.minitab.com/en/adventures-in-statistics-2/regression-analysis-how-to-interpret-s-the-standard-error-of-the-regression

Standard Error



x = number of minutes trick-or-treating

The “S” here is also used to denote Standard Error (SE)

Interpretation: the average distance of data points from the fitted line is 3.53 pieces of candy

https://blog.minitab.com/en/adventures-in-statistics-2/regression-analysis-how-to-interpret-s-the-standard-error-of-the-regression

Standard Error

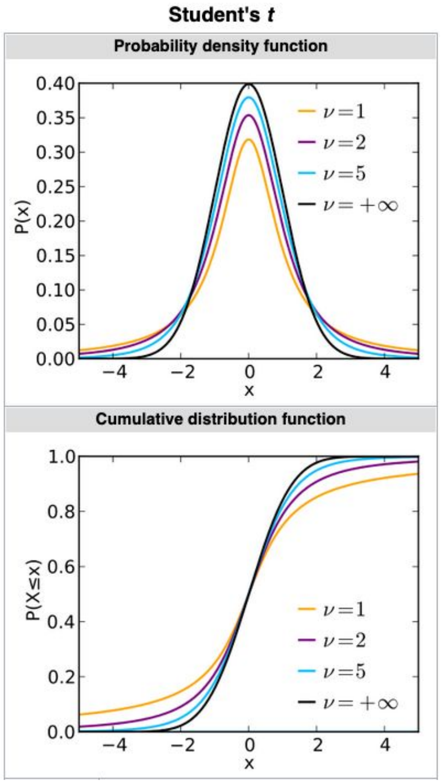
- SE represents the **average distance** between observed values and the regression line
- x = # minutes of trick-or-treating
- y = pieces of candy obtained

- **What units does SE take?**

Standard Error

- SE represents the **average distance** between observed values and the regression line
- x = # minutes of trick-or-treating
- y = pieces of candy obtained

- **What units does SE take? The same units as the y-axis value (pieces of candy obtained)**



Hypothesis Testing: Regression

- t-statistic: $\mathbf{t = (b - \beta) / SE_b}$
  - b = the estimated coefficient
  - SE_b = the "standard error" of the estimated coefficient
  - β = the coefficient under the null

How do we get SE_b in Python?

https://en.wikipedia.org/wiki/Student%27s_t-distribution

Another package for regressions

Using statsmodels instead of scikit-learn

```
>>> import statsmodels.api as sm
>>> import numpy as np
>>> duncan_prestige = sm.datasets.get_rdataset("Duncan", "carData")
>>> Y = duncan_prestige.data['income']
>>> X = duncan_prestige.data['education']
>>> X = sm.add_constant(X)
>>> model = sm.OLS(Y,X)
>>> results = model.fit()
```

https://www.statsmodels.org/dev/user-guide.html#regression-and-linear-models

Another package for regressions

```
>>> import statsmodels.api as sm
>>> import numpy as np
>>> duncan_prestige = sm.datasets.get_rdataset("Duncan", "carData")
>>> Y = duncan_prestige.data['income']
>>> X = duncan_prestige.data['education']
>>> X = sm.add_constant(X)
>>> model = sm.OLS(Y,X)
>>> results = model.fit()
```

This example is from Lec 7 slides: fitting $Y = \alpha + \beta x$

https://www.statsmodels.org/dev/user-guide.html#regression-and-linear-models

Hypothesis Testing: Regression

Printing .summary() of our model gives us a nice regression table!

```
>>> print(results.summary())
OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.862
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     15.56
Date:                Thu, 12 May 2022   Prob (F-statistic):            0.00713
Time:                        14:15:07   Log-Likelihood:                -24.316
No. Observations:                   8   AIC:                             54.63
Df Residuals:                       5   BIC:                             54.87
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5226      4.431      1.246      0.268      -5.867      16.912
x1             0.4471      0.285      1.567      0.178      -0.286       1.180
x2             0.2550      0.453      0.563      0.598      -0.910       1.420
```

https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

```
>>> print(results.summary())
OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.862
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     15.56
Date:                Thu, 12 May 2022   Prob (F-statistic):            0.00713
Time:                        14:15:07   Log-Likelihood:                -24.316
No. Observations:                   8   AIC:                             54.63
Df Residuals:                       5   BIC:                             54.87
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5226      4.431      1.246      0.268      -5.867      16.912
x1             0.4471      0.285      1.567      0.178      -0.286       1.180
x2             0.2550      0.453      0.563      0.598      -0.910       1.420
```

https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

```
>>> print(results.summary())
OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.862
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     15.56
Date:                Thu, 12 May 2022   Prob (F-statistic):            0.00713
Time:                        14:15:07   Log-Likelihood:                -24.316
No. Observations:                   8   AIC:                             54.63
Df Residuals:                       5   BIC:                             54.87
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5226      4.431      1.246      0.268      -5.867      16.912
x1             0.4471      0.285      1.567      0.178      -0.286       1.180
x2             0.2550      0.453      0.563      0.598      -0.910       1.420
```

Fill in the blank with our estimated model:

y = ________________

https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

```
>>> print(results.summary())
OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.862
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     15.56
Date:                Thu, 12 May 2022   Prob (F-statistic):            0.00713
Time:                        14:15:07   Log-Likelihood:                -24.316
No. Observations:                   8   AIC:                             54.63
Df Residuals:                       5   BIC:                             54.87
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5226      4.431      1.246      0.268      -5.867      16.912
x1             0.4471      0.285      1.567      0.178      -0.286       1.180
x2             0.2550      0.453      0.563      0.598      -0.910       1.420
```

$y = 5.52 + 0.45x_1 + 0.26x_2$

https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

```
>>> print(results.summary())
OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.862
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     15.56
Date:                Thu, 12 May 2022   Prob (F-statistic):            0.00713
Time:                        14:15:07   Log-Likelihood:                -24.316
No. Observations:                   8   AIC:                             54.63
Df Residuals:                       5   BIC:                             54.87
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5226      4.431      1.246      0.268      -5.867      16.912
x1             0.4471      0.285      1.567      0.178      -0.286       1.180
x2             0.2550      0.453      0.563      0.598      -0.910       1.420
```

Python gives us the SE too! (and the t-statistic, and the p-value!)

https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

(this is why we like statsmodels for regression tables – you need lin alg to calculate these if using sklearn)

```
>>> print(results.summary())
OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.862
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     15.56
Date:                Thu, 12 May 2022   Prob (F-statistic):            0.00713
Time:                        14:15:07   Log-Likelihood:                -24.316
No. Observations:                   8   AIC:                             54.63
Df Residuals:                       5   BIC:                             54.87
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5226      4.431      1.246      0.268      -5.867      16.912
x1             0.4471      0.285      1.567      0.178      -0.286       1.180
x2             0.2550      0.453      0.563      0.598      -0.910       1.420
```

https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

```
>>> print(results.summary())
OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.862
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     15.56
Date:                Thu, 12 May 2022   Prob (F-statistic):            0.00713
Time:                        14:15:07   Log-Likelihood:                -24.316
No. Observations:                   8   AIC:                             54.63
Df Residuals:                       5   BIC:                             54.87
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5226      4.431      1.246      0.268      -5.867      16.912
x1             0.4471      0.285      1.567      0.178      -0.286       1.180
x2             0.2550      0.453      0.563      0.598      -0.910       1.420
```

Which coefficient is most precisely estimated, based on SE?

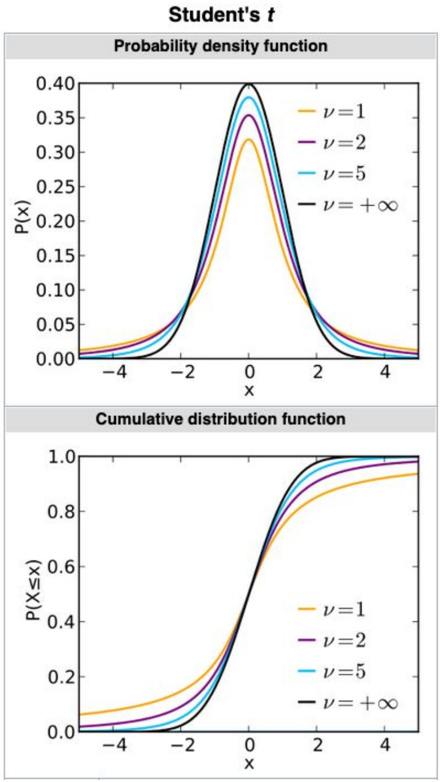
https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

```
>>> print(results.summary())
OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.862
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     15.56
Date:                Thu, 12 May 2022   Prob (F-statistic):            0.00713
Time:                        14:15:07   Log-Likelihood:                -24.316
No. Observations:                   8   AIC:                             54.63
Df Residuals:                       5   BIC:                             54.87
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5226      4.431      1.246      0.268      -5.867      16.912
x1             0.4471      0.285      1.567      0.178      -0.286       1.180
x2             0.2550      0.453      0.563      0.598      -0.910       1.420
```

x1 (has the lowest SE)

https://realpython.com/linear-regression-in-python/

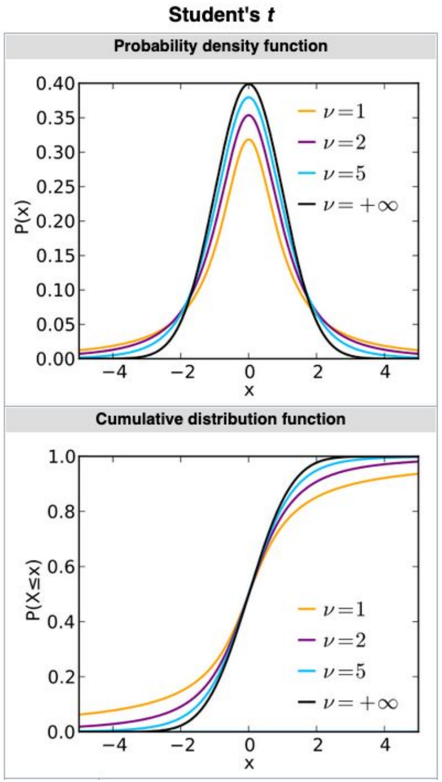


Hypothesis Testing: Regression

Python gives us both of these

- t-statistic: **$t = (b - \beta) / SE_b$**
  - b = the estimated coefficient
  - SE_b = the "standard error" of the estimated coefficient
  - β = the coefficient under the null

https://en.wikipedia.org/wiki/Student%27s_t-distribution

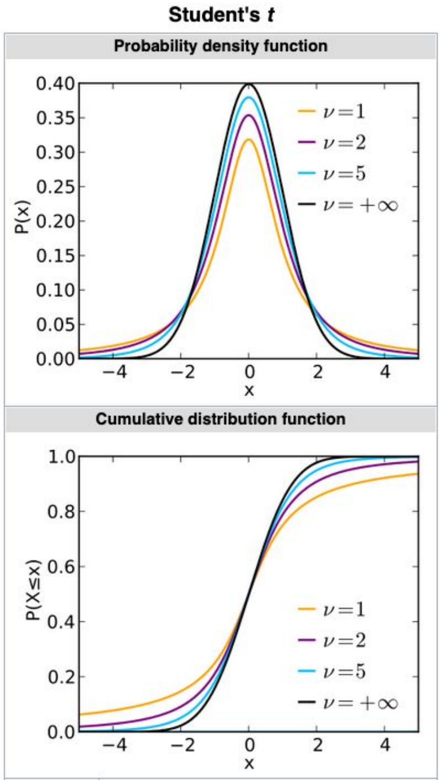


Hypothesis Testing: Regression

- t-statistic: **$t = (b - \beta) / SE_b$**
  - b = the estimated coefficient
  - SE_b = the “standard error” of the estimated coefficient
  - β = the coefficient under the null

How do we get this?

https://en.wikipedia.org/wiki/Student%27s_t-distribution



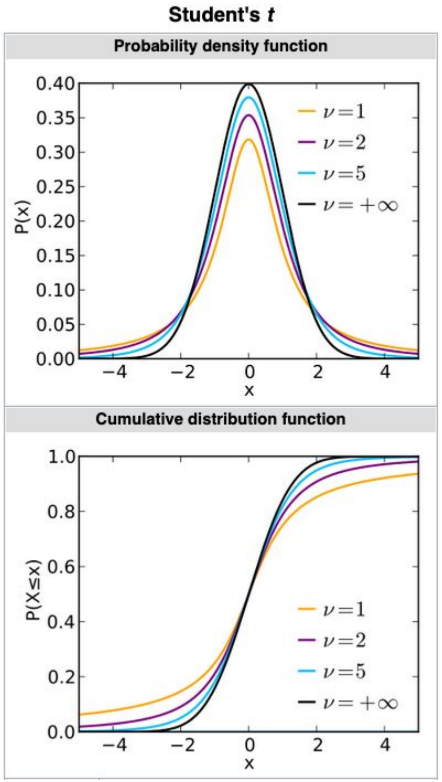
Hypothesis Testing: Regression

- t-statistic: **$t = (b - \beta) / SE_b$**
  - b = the estimated coefficient
  - SE_b = the "standard error" of the estimated coefficient
  - β = the coefficient under the null

Null (boring) hypothesis: there is no relationship between output (y) and input x_1

- **Equivalently, β_1=0**

https://en.wikipedia.org/wiki/Student%27s_t-distribution



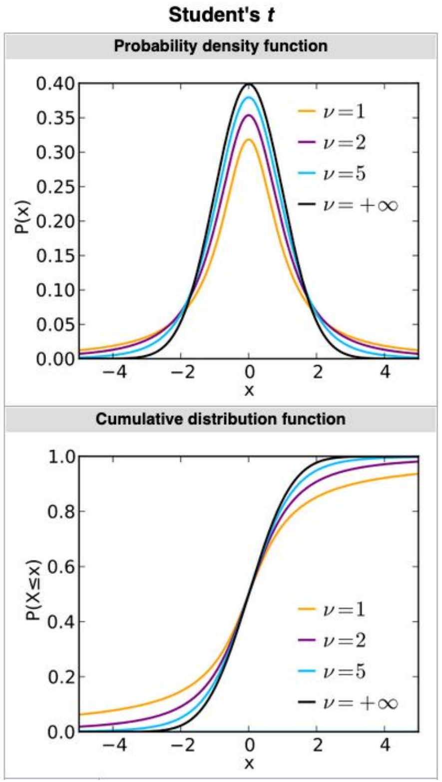
Hypothesis Testing: Regression

- t-statistic: $\mathbf{t = (b - \beta) / SE_b}$
  - b = the estimated coefficient
  - SE_b = the "standard error" of the estimated coefficient
  - β = the coefficient under the null

Just set β = 0

https://en.wikipedia.org/wiki/Student%27s_t-distribution

Hypothesis Testing: Regression



- t-statistic: **t = b / SE$_b$**
  - b = the estimated coefficient
  - SE$_b$ = the "standard error" of the estimated coefficient

https://en.wikipedia.org/wiki/Student%27s_t-distribution

Hypothesis Testing: Regression

Predictor	Coeff
Constant	1.352
Weight	0.9207

Let's practice with an example:

y = price of skincare product

x_1 = weight of product

Least-Squares Regression Line: $\hat{y} = a + bx \quad \rightarrow$

https://calcworkshop.com/linear-regression/t-test/

Hypothesis Testing: Regression

Predictor	Coeff
Constant	1.352
Weight	0.9207

Least-Squares Regression Line: $\hat{y} = a + bx \rightarrow \hat{y} = 1.352 + 0.9207x$

https://calcworkshop.com/linear-regression/t-test/

Hypothesis Testing: Regression

a

SE_b

Predictor	Coeff	SE	T	P
Constant	1.352	2.501	0.46	0.315
Weight	0.9207	0.8104	1.136	0.1375
R - Sq = 82.0%	R - $Sq(adj)$ = 81.1%			

P
(two-sided t-test)

b

t

Least-Squares Regression Line: $\hat{y} = a + bx \rightarrow \hat{y} = 1.352 + 0.9207x$

https://calcworkshop.com/linear-regression/t-test/

Hypothesis Testing: Regression

a SE_b

Predictor	Coeff	SE	T	P
Constant	1.352	2.501	0.46	0.315
Weight	0.9207	0.8104	1.136	0.1375
$R - Sq = 82.0\%$ $R - Sq(adj) = 81.1\%$				

P (two-sided t-test)

b t

If $H_o: \beta = 0$ *and* $H_a: \beta \neq 0$, then

Test Statistic: $t = \dfrac{b - \beta}{SE_b} \rightarrow t = \dfrac{b - \beta}{SE_b} \rightarrow t = \dfrac{0.9207 - 0}{0.8104} = 1.136$

https://calcworkshop.com/linear-regression/t-test/

Hypothesis Testing: Regression

a SE_b

Predictor	Coeff	SE	T	P
Constant	1.352	2.501	0.46	0.315
Weight	0.9207	0.8104	1.136	0.1375
R - Sq = 82.0% R - $Sq(adj)$ = 81.1%				

b

P
(two-sided t-test)

t

If $H_o: \beta = 0$ *and* $H_a: \beta \neq 0$, then

Test Statistic: $t = \dfrac{b-\beta}{SE_b} \rightarrow t = \dfrac{b-\beta}{SE_b} \rightarrow t = \dfrac{0.9207-0}{0.8104} = 1.136$

Hypothesis Testing: Regression

Predictor	Coeff	SE	T	P
Constant	1.352	2.501	0.46	0.315
Weight	0.9207	0.8104	1.136	0.1375
$R - Sq = 82.0\%$ $R - Sq(adj) = 81.1\%$				

a — 1.352

b — 0.9207

SE_b — 0.8104

t — 1.136

P (two-sided t-test) — 0.1375

If $H_o: \beta = 0$ *and* $H_a: \beta \neq 0$, then

Test Statistic: $t = \dfrac{b - \beta}{SE_b} \rightarrow t = \dfrac{b - \beta}{SE_b} \rightarrow t = \dfrac{0.9207 - 0}{0.8104} = 1.136$

Hypothesis Testing: Regression

a

SE_b

Predictor	Coeff	SE	T	P
Constant	1.352	2.501	0.46	0.315
Weight	0.9207	0.8104	1.136	0.1375
R - Sq = 82.0%	R - $Sq(adj)$ = 81.1%			

P
(two-sided t-test)

b

t

If $H_o: \beta = 0$ *and* $H_a: \beta \neq 0$, then

Test Statistic: $t = \dfrac{b - \beta}{SE_b} \rightarrow t = \dfrac{b - \beta}{SE_b} \rightarrow t = \dfrac{0.9207 - 0}{0.8104} = 1.136$

https://calcworkshop.com/linear-regression/t-test/

Hypothesis Testing: Regression

Predictor	Coeff	SE	T	P
Constant	1.352	2.501	0.46	0.315
Weight	0.9207	0.8104	1.136	0.1375
$R-Sq = 82.0\%$	$R-Sq(adj) = 81.1\%$			

a

b

SE_b

t

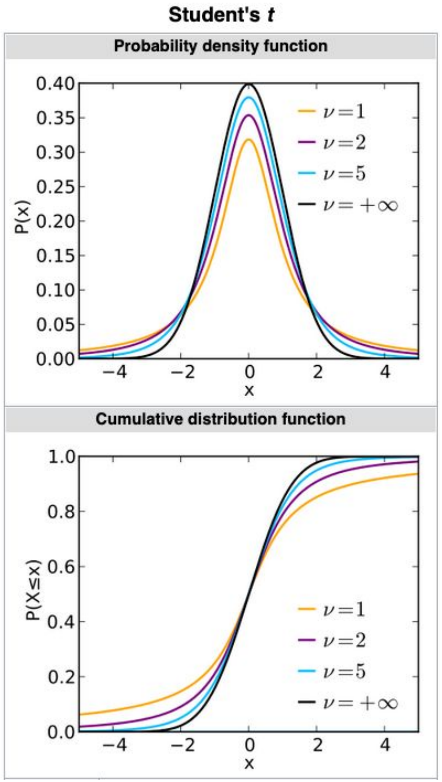
P
(two-sided t-test)

If $H_o: \beta = 0$ *and* $H_a: \beta \neq 0$, then

Test Statistic: $t = \dfrac{b-\beta}{SE_b} \rightarrow t = \dfrac{b-\beta}{SE_b} \rightarrow t = \dfrac{0.9207 - 0}{0.8104} = 1.136$

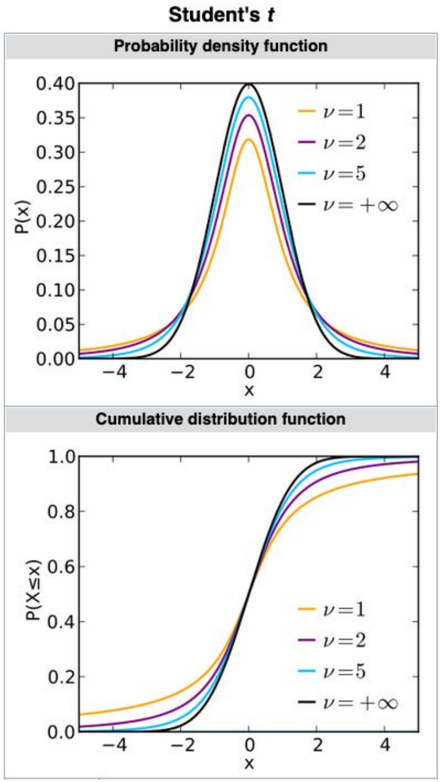
https://calcworkshop.com/linear-regression/t-test/

Hypothesis Testing: Regression



- t-statistic: **$t = (b - \beta) / SE_b$**
- Compare to t-distribution
- Decide if t spooky enough to reject null

https://en.wikipedia.org/wiki/Student%27s_t-distribution



Hypothesis Testing: Regression

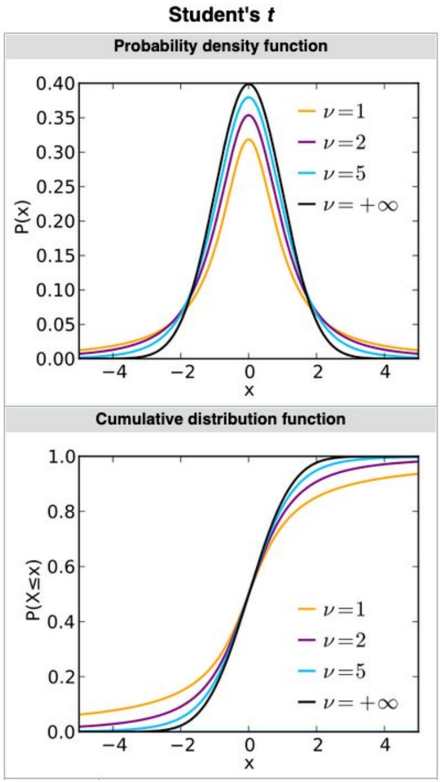
- t-statistic: **$t = (b - \beta) / SE_b$**
- Compare to t-distribution
- Decide if t spooky enough to reject null

Degrees of freedom is N-p-1, where p = # x's in the regression, due to some math involving us not knowing the error ε's variance

https://en.wikipedia.org/wiki/Student%27s_t-distribution

https://calcworkshop.com/linear-regression/t-test/

Hypothesis Testing: Regression



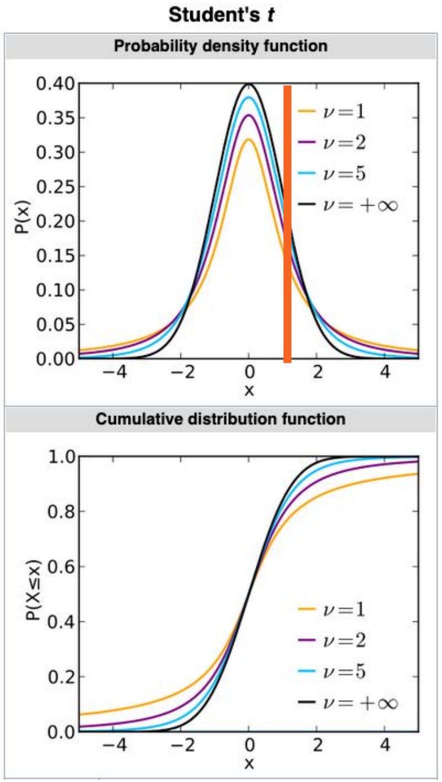
- t-statistic: **$t = (b - \beta) / SE_b$**
- Compare to t-distribution
- Decide if t spooky enough to reject null

Test Statistic: $t = \dfrac{b-\beta}{SE_b} \rightarrow t = \dfrac{b-\beta}{SE_b} \rightarrow t = \dfrac{0.9207 - 0}{0.8104} = 1.136$

https://en.wikipedia.org/wiki/Student%27s_t-distribution https://calcworkshop.com/linear-regression/t-test/

Hypothesis Testing: Regression

Not very spooky!



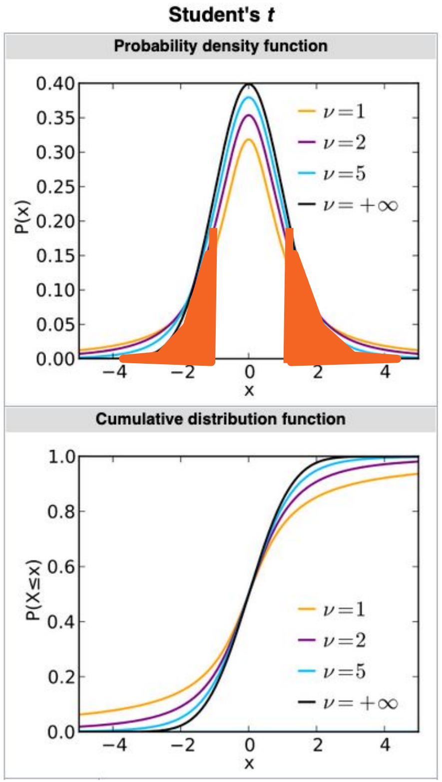
- t-statistic: **$t = (b - \beta) / SE_b$**
- Compare to t-distribution
- Decide if t spooky enough to reject null

Test Statistic: $t = \dfrac{b-\beta}{SE_b} \rightarrow t = \dfrac{b-\beta}{SE_b} \rightarrow t = \dfrac{0.9207 - 0}{0.8104} = \boxed{1.136}$

https://en.wikipedia.org/wiki/Student%27s_t-distribution

https://calcworkshop.com/linear-regression/t-test/

Not very spooky!



Hypothesis Testing: Regression

- t-statistic: **t = (b - β) / SE$_b$**
- Compare to t-distribution
- Decide if t spooky enough to reject null

Test Statistic: $t = \frac{b-\beta}{SE_b} \rightarrow t = \frac{b-\beta}{SE_b} \rightarrow t = \frac{0.9207 - 0}{0.8104} = 1.136$

P-Value: $p = 2P\left(t > |test\ statistic|\right) \rightarrow p = 2P\left(t > 1.136\right) = \boxed{0.1375}$

https://en.wikipedia.org/wiki/Student%27s_t-distribution https://calcworkshop.com/linear-regression/t-test/

Hypothesis Testing: Regression

a SE_b

Predictor	Coeff	SE	T	P
Constant	1.352	2.501	0.46	0.315
Weight	0.9207	0.8104	1.136	0.1375
$R - Sq = 82.0\%$ $R - Sq(adj) = 81.1\%$				

P (two-sided t-test)

b t

- **Summarize the relationship between weight (grams) and price ($) of skincare products based on the regression.**

Hypothesis Testing: Regression

a

SE_b

Predictor	Coeff	SE	T	P
Constant	1.352	2.501	0.46	0.315
Weight	0.9207	0.8104	1.136	0.1375
$R - Sq = 82.0\%$ $R - Sq(adj) = 81.1\%$				

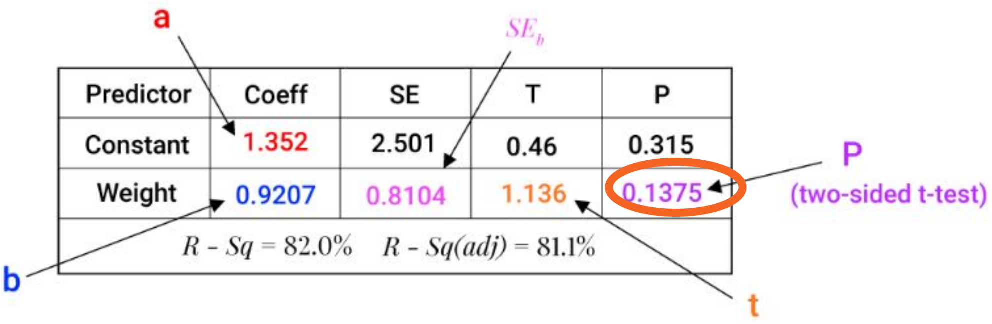
P
(two-sided t-test)

b

t

- For 1 additional gram increase in skincare product weight, our model predicts that price will increase by 92 cents.

Hypothesis Testing: Regression



Predictor	Coeff	SE	T	P
Constant	1.352	2.501	0.46	0.315
Weight	0.9207	0.8104	1.136	0.1375
$R - Sq = 82.0\%$ $\quad R - Sq(adj) = 81.1\%$				

- For 1 additional gram increase in skincare product weight, our model predicts that price will increase by 92 cents.
- But, we **cannot** reject the null hypothesis. *Null hypothesis (b=0): there is no relationship between price and weight of skincare products*
- Our weight coefficient is not significant (even at the 10% level)

https://calcworkshop.com/linear-regression/t-test/

Interpreting regression significance

Coefficients

Term	Coef	SE Coef	T	P
Constant	389.166	66.0937	5.8881	0.000
East	2.125	1.2145	1.7495	0.092
South	5.318	0.9629	5.5232	0.000
North	-24.132	1.8685	-12.9153	0.000

https://statisticsbyjim.com/regression/interpret-coefficients-p-values-regression/

Interpreting regression significance

Coefficients

Term	Coef	SE Coef	T	P
Constant	389.166	66.0937	5.8881	0.000
East	2.125	1.2145	1.7495	0.092
South	5.318	0.9629	5.5232	0.000
North	-24.132	1.8685	-12.9153	0.000

How many β coefficients are significant at the 5% level?

https://statisticsbyjim.com/regression/interpret-coefficients-p-values-regression/

Interpreting regression significance

Coefficients

Term	Coef	SE Coef	T	P
Constant	389.166	66.0937	5.8881	0.000
East	2.125	1.2145	1.7495	0.092
South	5.318	0.9629	5.5232	0.000
North	-24.132	1.8685	-12.9153	0.000

How many β coefficients are significant at the 5% level?

Two of them: only South and North inputs have significant coefficients (we're not counting Constant – the intercept)

https://statisticsbyjim.com/regression/interpret-coefficients-p-values-regression/

Interpreting regression significance

Coefficients

Term	Coef	SE Coef	T	P
Constant	389.166	66.0937	5.8881	0.000
East	2.125	1.2145	1.7495	0.092
South	5.318	0.9629	5.5232	0.000
North	-24.132	1.8685	-12.9153	0.000

We think that South and North inputs each have significant effects on the outcome

https://statisticsbyjim.com/regression/interpret-coefficients-p-values-regression/

Interpreting regression significance

Coefficients

Term	Coef	SE Coef	T	P
Constant	389.166	66.0937	5.8881	0.000
East	2.125	1.2145	1.7495	0.092
South	5.318	0.9629	5.5232	0.000
North	-24.132	1.8685	-12.9153	0.000

South has a significant *positive* effect;
North has a significant *negative* effect

https://statisticsbyjim.com/regression/interpret-coefficients-p-values-regression/

Interpreting regression significance

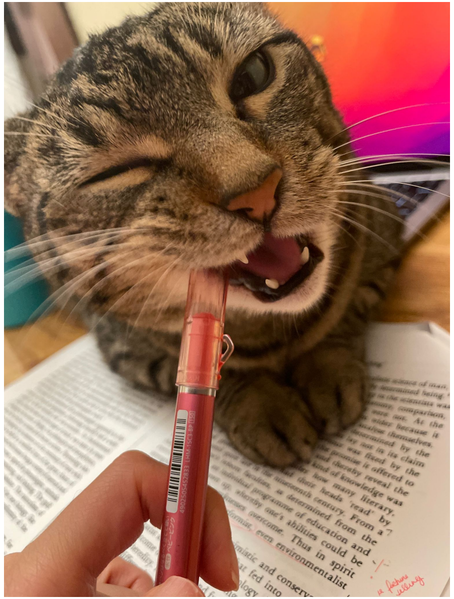
Coefficients

Term	Coef	SE Coef	T	P
Constant	389.166	66.0937	5.8881	0.000
East	2.125	1.2145	1.7495	0.092
South	5.318	0.9629	5.5232	0.000
North	-24.132	1.8685	-12.9153	0.000

Sometimes, if a variable is very not-significant and has near-0 magnitude effect on the output, it can be a good idea to drop them from the regression since they might just be adding noise to your model

https://statisticsbyjim.com/regression/interpret-coefficients-p-values-regression/

1 min break & attendance



tinyurl.com/4fkxswx9

Hypothesis Testing: Regression

```
>>> print(results.summary())
OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.862
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     15.56
Date:                Thu, 12 May 2022   Prob (F-statistic):            0.00713
Time:                        14:15:07   Log-Likelihood:                -24.316
No. Observations:                   8   AIC:                             54.63
Df Residuals:                       5   BIC:                             54.87
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5226      4.431      1.246      0.268      -5.867      16.912
x1             0.4471      0.285      1.567      0.178      -0.286       1.180
x2             0.2550      0.453      0.563      0.598      -0.910       1.420
```

We've talked about these columns already

https://realpython.com/linear-regression-in-python/

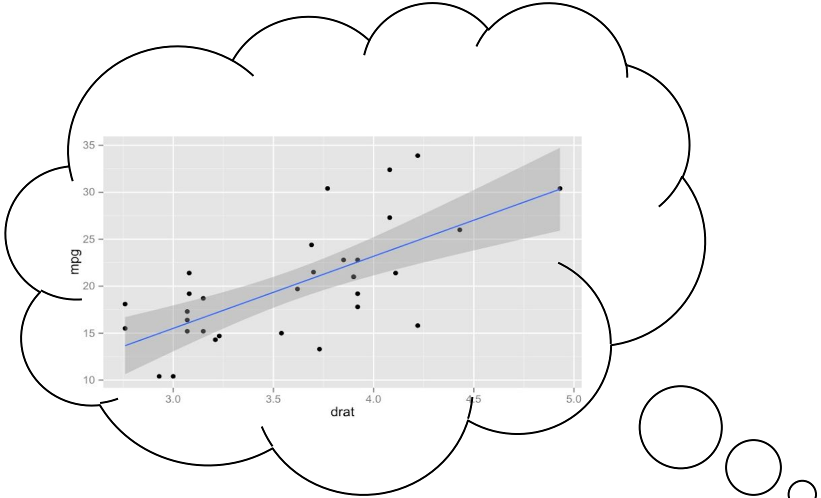
Hypothesis Testing: Regression

```
>>> print(results.summary())
OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.862
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     15.56
Date:                Thu, 12 May 2022   Prob (F-statistic):            0.00713
Time:                        14:15:07   Log-Likelihood:                -24.316
No. Observations:                   8   AIC:                             54.63
Df Residuals:                       5   BIC:                             54.87
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5226      4.431      1.246      0.268      -5.867      16.912
x1             0.4471      0.285      1.567      0.178      -0.286       1.180
x2             0.2550      0.453      0.563      0.598      -0.910       1.420
```

Python gives us the 95% CI for free for each coef, too

https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression



When we thought about Confidence Intervals before, we thought about how much we trust the regression output ŷ for any given input x

Hypothesis Testing: Regression

$100(1-\alpha)\%$ Confidence Interval for Slope

Point Estimate $\pm$ Margin of Error

$$b \pm t^{*}\left(SE_{b}\right)$$

But we can actually also think about the CI for each individual slope, too!

Hypothesis Testing: Regression

$100(1-\alpha)\%$ **Confidence Interval for Slope**

Point Estimate $\pm$ Margin of Error

$$b \pm t^{*}\left(SE_{b}\right)$$

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.5226	4.431	1.246	0.268	-5.867	16.912
x1	0.4471	0.285	1.567	0.178	-0.286	1.180
x2	0.2550	0.453	0.563	0.598	-0.910	1.420

https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

$100(1-\alpha)\%$ **Confidence Interval for Slope**

Point Estimate $\pm$ **Margin of Error**

$$b \pm \boxed{t}^{*}\left(SE_b\right)$$

95% CI → alpha of 0.05

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.5226	4.431	1.[illegible]	0.268	-5.867	16.912
x1	0.4471	0.285	1.[illegible]	0.178	-0.286	1.180
x2	0.2550	0.453	[illegible]63	0.598	-0.910	1.420

https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

n-p-1 degrees of freedom 8-2-1=5

```
>>> print(results.summary())
OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.862
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     15.56
Date:                Thu, 12 May 2022   Prob (F-statistic):            0.00713
Time:                        14:15:07   Log-Likelihood:                -24.316
No. Observations:                   8   AIC:                             54.63
Df Residuals:                       5   BIC:                             54.87
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5226      4.431      1.246      0.268      -5.867      16.912
x1             0.4471      0.285      1.567      0.178      -0.286       1.180
x2             0.2550      0.453      0.563      0.598      -0.910       1.420
```

n =

2 predictors p

https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

From t-Table:

- n-p-1 degrees of freedom
  8-2-1=5
- two-tailed 0.05 alpha
- t = 2.571

$100(1-\alpha)\%$ **Confidence Interval for Slope**

Point Estimate $\pm$ **Margin of Error**

$$b \pm \boxed{t}^{*}\left(SE_b\right)$$

95% CI → alpha of 0.05

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.5226	4.431	1.	0.268	-5.867	16.912
x1	0.4471	0.285	1.	0.178	-0.286	1.180
x2	0.2550	0.453	563	0.598	-0.910	1.420

https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

From t-Table:

- n-p-1 degrees of freedom
  8-2-1=5
- two-tailed 0.05 alpha
- t = 2.571

***t* Table**

cum. prob	$t_{.50}$	$t_{.75}$	$t_{.80}$	$t_{.85}$	$t_{.90}$	$t_{.95}$	$t_{.975}$	$t_{.99}$	$t_{.995}$	$t_{.999}$	$t_{.9995}$
one-tail	**0.50**	**0.25**	**0.20**	**0.15**	**0.10**	**0.05**	**0.025**	**0.01**	**0.005**	**0.001**	**0.0005**
two-tails	**1.00**	**0.50**	**0.40**	**0.30**	**0.20**	**0.10**	**0.05**	**0.02**	**0.01**	**0.002**	**0.001**
df											
1	0.000	1.000	1.376	1.963	3.078	6.314	12.71	31.82	63.66	318.31	636.62
2	0.000	0.816	1.061	1.386	1.886	2.920	4.303	6.965	9.925	22.327	31.599
3	0.000	0.765	0.978	1.250	1.638	2.353	3.182	4.541	5.841	10.215	12.924
4	0.000	0.741	0.941	1.190	1.533	2.132	2.776	3.747	4.604	7.173	8.610
5	0.000	0.727	0.920	1.156	1.476	2.015	$\boxed{2.571}$	3.365	4.032	5.893	6.869
6	0.000	0.718	0.906	1.134	1.440	1.943	2.447	3.143	3.707	5.208	5.959
7	0.000	0.711	0.896	1.119	1.415	1.895	2.365	2.998	3.499	4.785	5.408
8	0.000	0.706	0.889	1.108	1.397	1.860	2.306	2.896	3.355	4.501	5.041
9	0.000	0.703	0.883	1.100	1.383	1.833	2.262	2.821	3.250	4.297	4.781
10	0.000	0.700	0.879	1.093	1.372	1.812	2.228	2.764	3.169	4.144	4.587
11	0.000	0.697	0.876	1.088	1.363	1.796	2.201	2.718	3.106	4.025	4.437
12	0.000	0.695	0.873	1.083	1.356	1.782	2.179	2.681	3.055	3.930	4.318
13	0.000	0.694	0.870	1.079	1.350	1.771	2.160	2.650	3.012	3.852	4.221
14	0.000	0.692	0.868	1.076	1.345	1.761	2.145	2.624	2.977	3.787	4.140
15	0.000	0.691	0.866	1.074	1.341	1.753	2.131	2.602	2.947	3.733	4.073
16	0.000	0.690	0.865	1.071	1.337	1.746	2.120	2.583	2.921	3.686	4.015
17	0.000	0.689	0.863	1.069	1.333	1.740	2.110	2.567	2.898	3.646	3.965
18	0.000	0.688	0.862	1.067	1.330	1.734	2.101	2.552	2.878	3.610	3.922
19	0.000	0.688	0.861	1.066	1.328	1.729	2.093	2.539	2.861	3.579	3.883
20	0.000	0.687	0.860	1.064	1.325	1.725	2.086	2.528	2.845	3.552	3.850
21	0.000	0.686	0.859	1.063	1.323	1.721	2.080	2.518	2.831	3.527	3.819
22	0.000	0.686	0.858	1.061	1.321	1.717	2.074	2.508	2.819	3.505	3.792
23	0.000	0.685	0.858	1.060	1.319	1.714	2.069	2.500	2.807	3.485	3.768
24	0.000	0.685	0.857	1.059	1.318	1.711	2.064	2.492	2.797	3.467	3.745
25	0.000	0.684	0.856	1.058	1.316	1.708	2.060	2.485	2.787	3.450	3.725
26	0.000	0.684	0.856	1.058	1.315	1.706	2.056	2.479	2.779	3.435	3.707
27	0.000	0.684	0.855	1.057	1.314	1.703	2.052	2.473	2.771	3.421	3.690
28	0.000	0.683	0.855	1.056	1.313	1.701	2.048	2.467	2.763	3.408	3.674
29	0.000	0.683	0.854	1.055	1.311	1.699	2.045	2.462	2.756	3.396	3.659
30	0.000	0.683	0.854	1.055	1.310	1.697	2.042	2.457	2.750	3.385	3.646
40	0.000	0.681	0.851	1.050	1.303	1.684	2.021	2.423	2.704	3.307	3.551
60	0.000	0.679	0.848	1.045	1.296	1.671	2.000	2.390	2.660	3.232	3.460
80	0.000	0.678	0.846	1.043	1.292	1.664	1.990	2.374	2.639	3.195	3.416
100	0.000	0.677	0.845	1.042	1.290	1.660	1.984	2.364	2.626	3.174	3.390
1000	0.000	0.675	0.842	1.037	1.282	1.646	1.962	2.330	2.581	3.098	3.300
z	0.000	0.674	0.842	1.036	1.282	1.645	1.960	2.326	2.576	3.090	3.291
	0%	50%	60%	70%	80%	90%	95%	98%	99%	99.8%	99.9%
	Confidence Level										

https://www.sjsu.edu/faculty/gerstman/StatPrimer/t-table.pdf https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

$100(1-\alpha)\%$ **Confidence Interval for Slope**

Point Estimate $\pm$ Margin of Error

$$\underbrace{b \pm t^{*}\left(SE_{b}\right)}$$

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.5226	4.431	1.246	0.268	-5.867	16.912
x1	0.4471	0.285	1.567	0.178	-0.286	1.180
x2	0.2550	0.453	0.563	0.598	-0.910	1.420

https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

$100(1-\alpha)\%$ Confidence Interval for Slope

Point Estimate $\pm$ Margin of Error

$$b \pm t^{*}\left(SE_{b}\right)$$

The coefficient estimate will be exactly in the middle of the confidence interval

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.5226	4.431	1.246	0.268	-5.867	16.912
x1	0.4471	0.285	1.567	0.178	-0.286	1.180
x2	0.2550	0.453	0.563	0.598	-0.910	1.420

https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

$100(1-\alpha)\%$ **Confidence Interval for Slope**

Point Estimate $\pm$ **Margin of Error**

$$b \pm t^{*}\left(SE_{b}\right)$$

Remember when we said Margin of Error was about 2*SE for a 95% CI?

This is because t = 1.96 (check a t-table)!

https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

```
>>> print(results.summary())
OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.862
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     15.56
Date:                Thu, 12 May 2022   Prob (F-statistic):            0.00713
Time:                        14:15:07   Log-Likelihood:                -24.316
No. Observations:                   8   AIC:                             54.63
Df Residuals:                       5   BIC:                             54.87
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5226      4.431      1.246      0.268      -5.867      16.912
x1             0.4471      0.285      1.567      0.178      -0.286       1.180
x2             0.2550      0.453      0.563      0.598      -0.910       1.420
```

What does it mean if 0 is in the CI?

https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

It means that if we take 100 samples of the coefficient and compute a 95% CI for each sample, about 95/100 samples will contain the value β = 0

What does β = 0 mean?

```
>>> print(results.summary())
OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.862
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     15.56
Date:                Thu, 12 May 2022   Prob (F-statistic):            0.00713
Time:                        14:15:07   Log-Likelihood:                -24.316
No. Observations:                   8   AIC:                             54.63
Df Residuals:                       5   BIC:                             54.87
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5226      4.431      1.246      0.268      -5.867      16.912
x1             0.4471      0.285      1.567      0.178      -0.286       1.180
x2             0.2550      0.453      0.563      0.598      -0.910       1.420
```

https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

β = 0 means that input has no effect on the outcome

```
>>> print(results.summary())
OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.862
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     15.56
Date:                Thu, 12 May 2022   Prob (F-statistic):            0.00713
Time:                        14:15:07   Log-Likelihood:                -24.316
No. Observations:                   8   AIC:                             54.63
Df Residuals:                       5   BIC:                             54.87
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5226      4.431      1.246      0.268      -5.867      16.912
x1             0.4471      0.285      1.567      0.178      -0.286       1.180
x2             0.2550      0.453      0.563      0.598      -0.910       1.420
```

https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

β = 0 means that input has no effect on the outcome

Summarizing: a 1 unit increase in x_1 yields a ____ increase in y.

```
>>> print(results.summary())
OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.862
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     15.56
Date:                Thu, 12 May 2022   Prob (F-statistic):            0.00713
Time:                        14:15:07   Log-Likelihood:                -24.316
No. Observations:                   8   AIC:                             54.63
Df Residuals:                       5   BIC:                             54.87
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5226      4.431      1.246      0.268      -5.867      16.912
x1             0.4471      0.285      1.567      0.178      -0.286       1.180
x2             0.2550      0.453      0.563      0.598      -0.910       1.420
```

https://realpython.com/linear-regression-in-python/

Hypothesis Testing: Regression

β = 0 means that input has no effect on the outcome

Summarizing: a 1 unit increase in x_1 yields a 0 unit increase in y.

Predicting: no matter what value we put in x_1, y will not be affected

```
>>> print(results.summary())
OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.862
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     15.56
Date:                Thu, 12 May 2022   Prob (F-statistic):            0.00713
Time:                        14:15:07   Log-Likelihood:                -24.316
No. Observations:                   8   AIC:                             54.63
Df Residuals:                       5   BIC:                             54.87
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5226      4.431      1.246      0.268      -5.867      16.912
x1             0.4471      0.285      1.567      0.178      -0.286       1.180
x2             0.2550      0.453      0.563      0.598      -0.910       1.420
```

https://realpython.com/linear-regression-in-python/

False discoveries

Assume we do an experiment and find $p = 0.05$

What is the chance that there isn't actually an effect, and the result we got is purely due to random chance?

False discoveries

Assume we do an experiment and find p = 0.05

What is the chance that there isn't actually an effect, and the result we got is purely due to random chance?

5%, or about 1 in 20

False discoveries

Assume we do 100 experiments

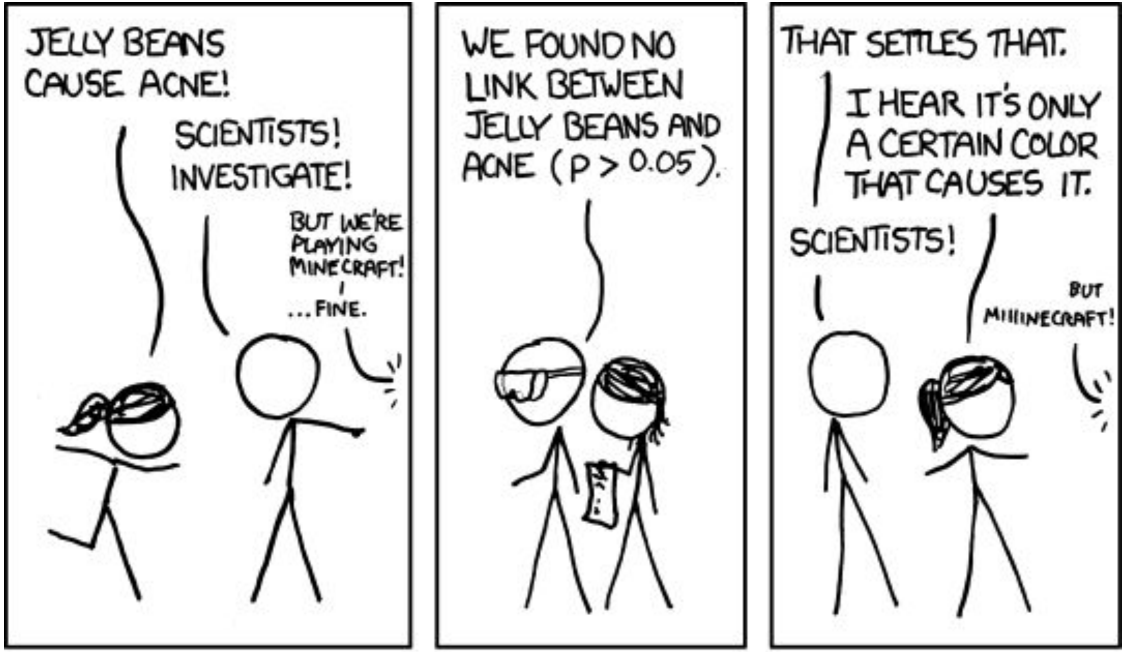
How many results do we expect to be "significant" at $p < 0.05$ purely by random chance?

False discoveries

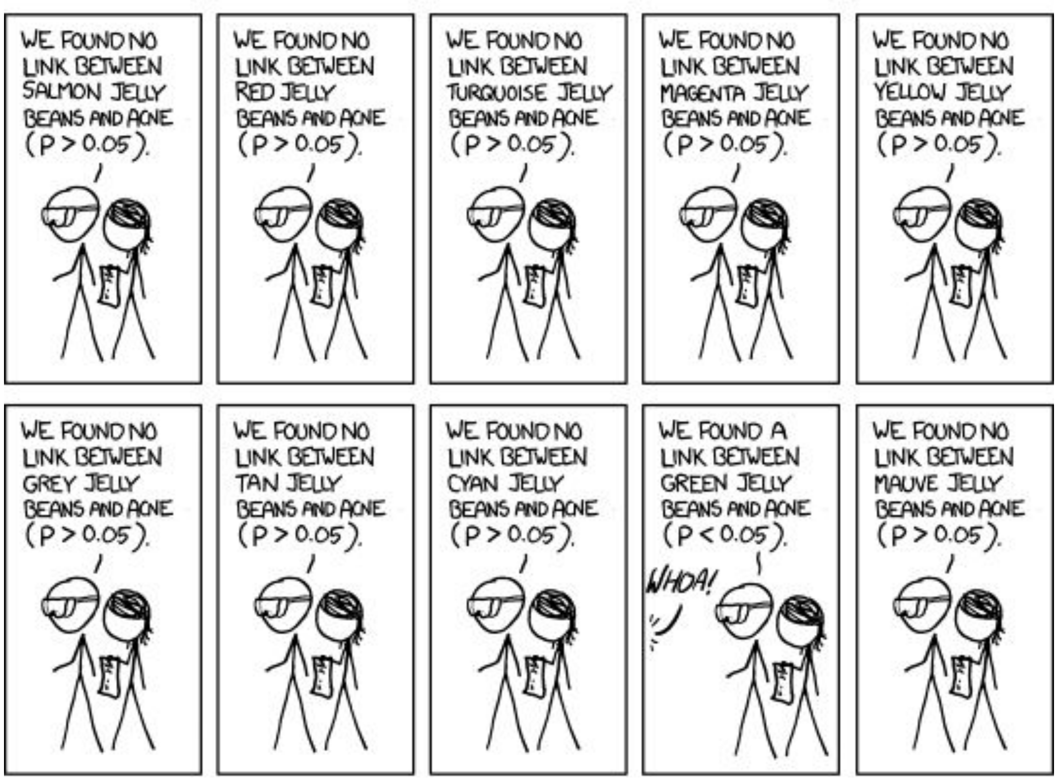
Assume we do 100 experiments

How many results do we expect to be "significant" at $p < 0.05$ purely by random chance?

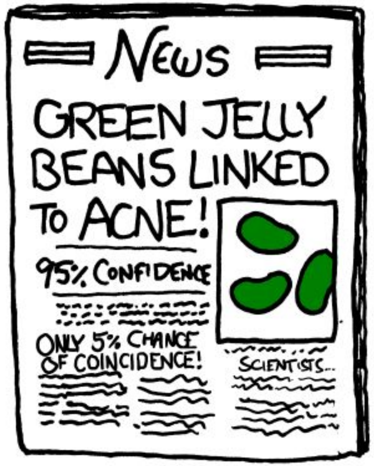
On average: 5, the expected value of Binomial($p=0.05$, $N=100$)



https://xkcd.com/882/



https://xkcd.com/882/

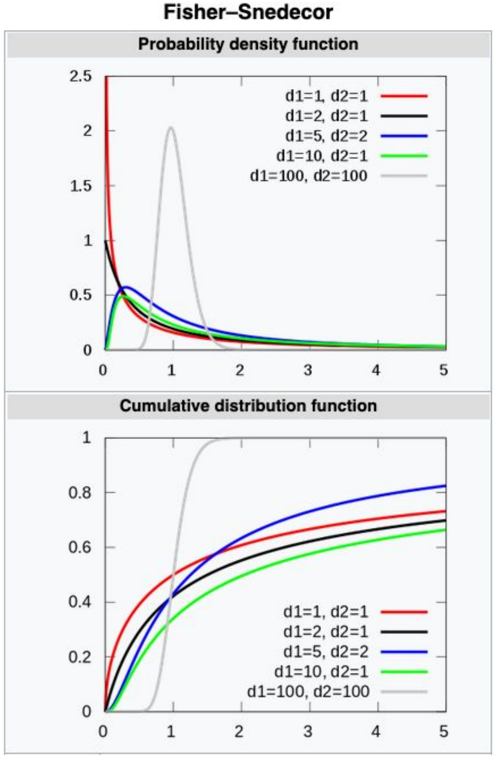


https://xkcd.com/882/

Hypothesis Testing: Regression

- If you have n input coefficients $x_1, x_2, \ldots, x_m$, you'd have *m* hypothesis tests to do (one for each coefficient) using the t-test
- Or, we can do them all at once using an f-test!

Regression significance: F-tests

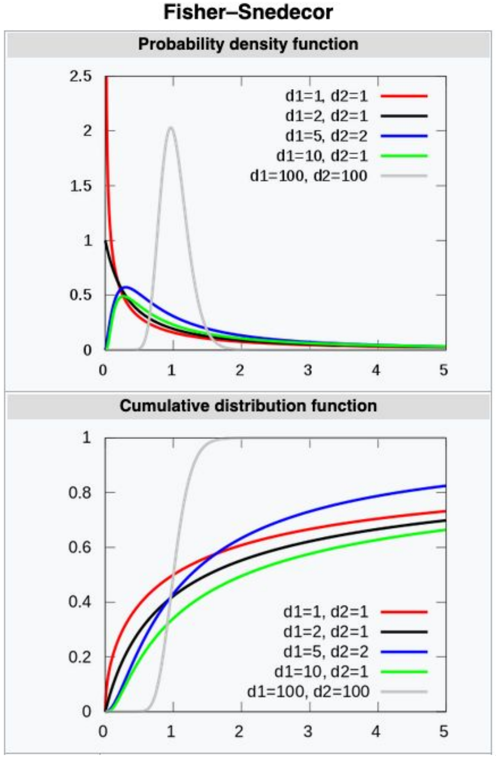


<u>t-tests</u>: compare small-sample, unknown sd population's **means**

<u>f-tests</u>: compare normal population's **variances**

https://en.wikipedia.org/wiki/F-distribution

Regression significance: F-tests



- Use f-tests to assess **multiple** regression coefficients simultaneously
- null hypothesis = your model's fit is **the same as** that of the intercept-only model
- alternative hypothesis: your model's fit is **better than** that of the intercept-only model

https://en.wikipedia.org/wiki/F-distribution

Hypothesis Testing: Regression

```
>>> print(results.summary())
OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.862
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     15.56
Date:                Thu, 12 May 2022   Prob (F-statistic):            0.00713
Time:                        14:15:07   Log-Likelihood:                -24.316
No. Observations:                   8   AIC:                             54.63
Df Residuals:                       5   BIC:                             54.87
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5226      4.431      1.246      0.268      -5.867      16.912
x1             0.4471      0.285      1.567      0.178      -0.286       1.180
x2             0.2550      0.453      0.563      0.598      -0.910       1.420
```

https://realpython.com/linear-regression-in-python/

Regression significance: F-tests

Hypotheses 1

Is the regression model containing at least one predictor useful in predicting the size of the infarct?

- H_0: $\beta_1 = \beta_2 = \beta_3 = 0$
- H_A: At least one $\beta_j \neq 0$ (for j = 1, 2, 3)

Hypotheses 2

Is the size of the infarct significantly (linearly) related to the area of the region at risk?

- H_0: $\beta_1 = 0$
- H_A: $\beta_1 \neq 0$

Hypotheses 3

(Primary research question) Is the size of the infarct area significantly (linearly) related to the type of treatment upon controlling for the size of the region at risk for infarction?

- H_0: $\beta_2 = \beta_3 = 0$
- H_A: At least one $\beta_j \neq 0$ (for j = 2, 3)

Let's test each of the hypotheses now using the general linear *F*-statistic:

$$F^* = \left(\frac{SSE(R) - SSE(F)}{df_R - df_F} \right) \div \left(\frac{SSE(F)}{df_F} \right)$$

https://online.stat.psu.edu/stat501/lesson/6/6.4

https://www.reddit.com/r/statisticsmemes/comments/10dgwcc/multiple_linear_regression_significance_testing/

Multiple hypothesis testing (generally)

- We can test multiple hypotheses at "once"!
- If we have a null hypothesis H_0 and *m* different hypotheses H_1, H_2, …H_m:
  - Test H_0 and H_1; reject the null if "significant"
  - Test H_0 and H_2; reject the null if "significant"
  - …etc.
- Can summarize results over them in a table

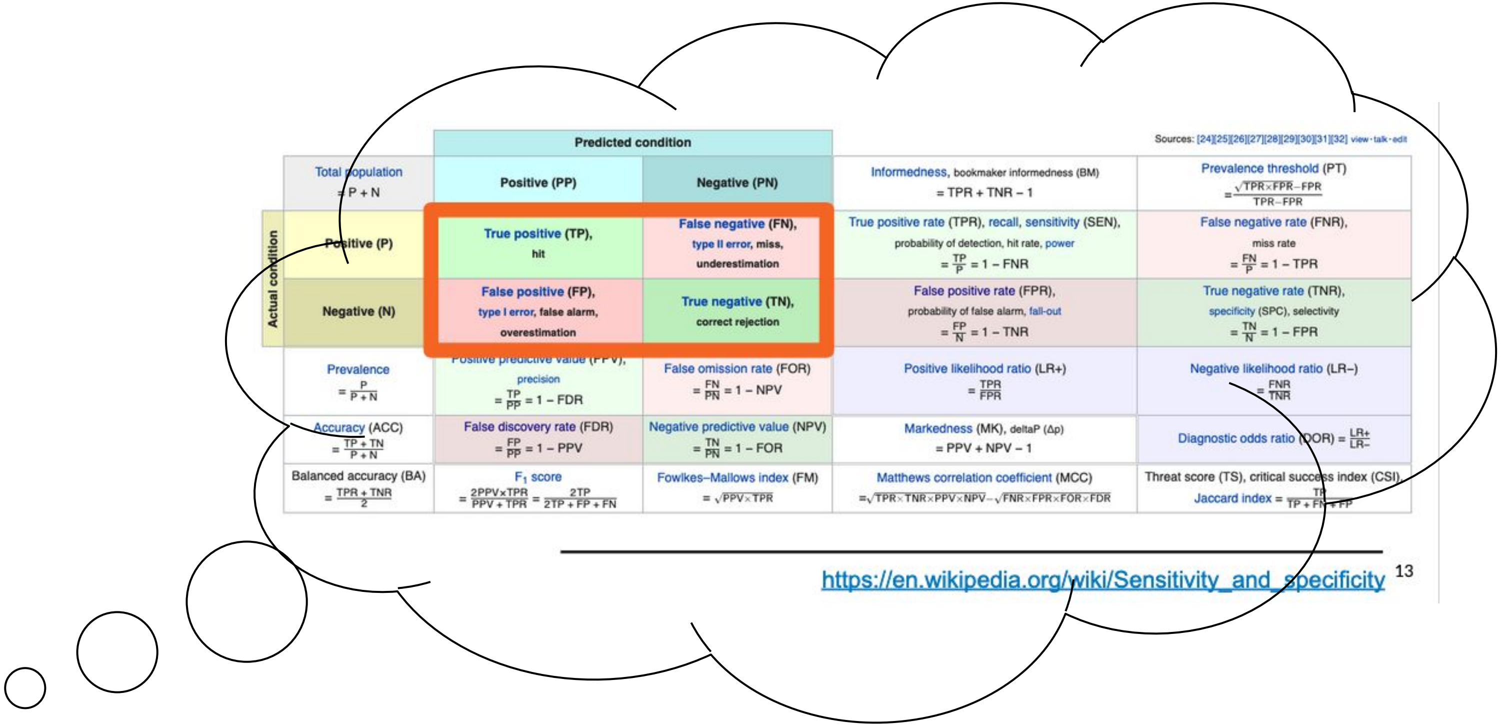
Multiple hypothesis testing

	Null hypothesis is true (H_0)	Alternative hypothesis is true (H_A)	Total
Test is declared significant	V	S	R
Test is declared non-significant	U	T	$m - R$
Total	m_0	$m - m_0$	m

- m is the total number hypotheses tested
- m_0 is the number of true null hypotheses, an unknown parameter
- $m - m_0$ is the number of true alternative hypotheses
- V is the number of false positives (Type I error) (also called "false discoveries")
- S is the number of true positives (also called "true discoveries")
- T is the number of false negatives (Type II error)
- U is the number of true negatives
- $R = V + S$ is the number of rejected null hypotheses (also called "discoveries", either true or false)

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

Reminds us of...?



Sources: [24][25][26][27][28][29][30][31][32] view · talk · edit

		Predicted condition			
	Total population $= P + N$	Positive (PP)	Negative (PN)	Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection	False positive rate (FPR), probability of false alarm, fall-out $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN} = 1 - FOR$	Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2 PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

https://en.wikipedia.org/wiki/Sensitivity_and_specificity 13

Controlling your errors

- We talk about rejecting the null at some level α
- If we do *m* hypothesis tests, there's an issue: generally, as the number of comparisons *m* increases, the number of false positives (V) increases too!

Controlling your errors

- We talk about rejecting the null at some level α
- If we do *m* hypothesis tests, there's an issue: generally, as the number of comparisons *m* increases, the number of false positives (V) increases too!
- This means we artificially get a bunch of comparisons telling us to reject the null when we really shouldn't
- We can fix this by using different ***correction*** methods

Correction methods

- Not correcting for this is p-hacking!
- Keywords to know: *Bonferroni correction* and *Benjamini-Hochberg procedure*
  - These methods correct for the issue of uncontrolled false positives
- We'll show more about these in next Friday (Nov 3rd) discussion section!

https://multithreaded.stitchfix.com/blog/2015/10/15/multiple-hypothesis-testing/

Recap on hypothesis testing

- We can do hypothesis testing on specific coefficients of a regression, which helps us interpret a model
- We can do **multiple** hypothesis testing to compare a bunch of different alternative hypotheses, but we need to be careful with correcting for bias
- All of these hypothesis tests give us p-values… be wary of how much to trust them!

https://www.themarysue.com/statistical-distribution-plushies/