INFO 2950:
Intro to Data Science

Lecture 7
2023-09-13

Agenda

1. **Linear regression**
   a. Interpretation review
   b. Dummy variables
   c. Math
   d. Python
2. **Phase 1 examples**
3. **Reshaping Datasets**

Homework submission snafus (do not do these!!) Part 1

- Submitting the html instead of the pdf (or submitting a funky pdf)
- Tagging instructions instead of just solutions
- Tagging the correct question #s
- Putting full names instead of netids
- Not submitting the ipynb file

Homework submission snafus (do not do these!!) Part 2

- Submitting code that goes off the PDF page (break it into multiple lines!)
- Citing ChatGPT without including your prompt and explaining how you assessed the generated code for correctness
- Not executing all of your code before submitting

Homework

- **DO** Check Ed Discussion posts for common problems!

- **DO** go to Student Hours if you have questions! We are here to help.

Last time on interpreting regressions

1. Summarize relationship between variables
2. Make predictions
3. Inspect outliers and other oddities

Regression interpretations: summarize relationship

x = millimeters of rainfall

y = umbrellas sold

$y = -19 + 0.45x$

Summarize relationship between variables:

Our model shows a positive relationship between rain and sales of umbrellas; specifically, each additional mm of rain corresponds to an extra 0.45 umbrellas we expect to be sold.

x = days (2023-01-01 to 2023-03-01)

y = hot chocolate sold

$y = -0.5x + 50$

Summarize relationship between variables:

____

____

____

Regression interpretations: summarize relationship

x = millimeters of rainfall

y = umbrellas sold

$y = -19 + 0.45x$

Summarize relationship between variables:

Our model shows a positive relationship between rain and sales of umbrellas; specifically, each additional mm of rain corresponds to an extra 0.45 umbrellas we expect to be sold.

x = days (2023-01-01 to 2023-03-01)

y = hot chocolate sold

$y = -0.5x + 50$

Summarize relationship between variables:

This model shows a negative relationship between days (between Jan 1 2023 and Mar 1 2023) and hot chocolate sales. Specifically, each additional day that goes by corresponds to half a fewer unit of hot chocolate sold.

Regression interpretations: make predictions

x = millimeters of rainfall

y = umbrellas sold

$y = -19 + 0.45x$

Make predictions:

This model indicates that at the annual Ithaca average of 110mm of rainfall, we should expect to sell 30 umbrellas.

x = days (2023-01-01 to 2023-03-01)

y = hot chocolate sold

$y = -0.5x + 50$

Make predictions:

Regression interpretations: make predictions

x = millimeters of rainfall

y = umbrellas sold

y = -19 + 0.45x

Make predictions:

This model indicates that at the annual Ithaca average of 110mm of rainfall, we should expect to sell 30 umbrellas.

x = days (2023-01-01 to 2023-03-01)

y = hot chocolate sold

y = -0.5x + 50

Make predictions:

This model predicts that...

on Dec 31, 2023, 50.5 cups of hot chocolate will be sold; on Jan 1, 2023, 50 cups of hot chocolate will be sold; on Jan 2, 2023, 49.5 cups of hot chocolate will be sold; on Jan 3, 2023, 49 cups of hot chocolate will be sold; on Mar 1, 2023, 20 cups of hot chocolate will be sold; on Mar 2, 2023, 19.5 cups of hot chocolate will be sold; on Apr 11, 2023, -0.5 cups of hot chocolate will be sold

Regression interpretations: note oddities

x = millimeters of rainfall

y = umbrellas sold

$y = -19 + 0.45x$

Inspect oddities / outliers:

We expect this model to hold for rainfall amounts between 80-170mm, but cannot extrapolate further.

x = days (2023-01-01 to 2023-03-01)

y = hot chocolate sold

$y = -0.5x + 50$

Inspect oddities / outliers:

Regression interpretations: note oddities

x = millimeters of rainfall

y = umbrellas sold

$y = -19 + 0.45x$

Inspect oddities / outliers:

We expect this model to hold for rainfall amounts between 80-170mm, but cannot extrapolate further.

x = days (2023-01-01 to 2023-03-01)

y = hot chocolate sold

$y = -0.5x + 50$

Inspect oddities / outliers:

This model is only based on a certain set of dates (Jan 1 to Mar 1, 2023). If used to predict dates outside of that range, the negative relationship between days and hot chocolate sales may not hold.

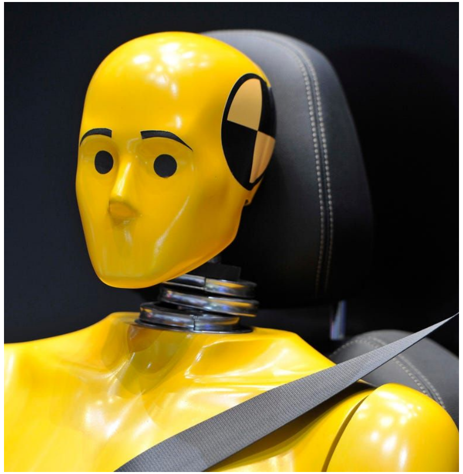
At some dates (> 100 days into 2023), the predicted # hot chocolates sold becomes negative, which is nonsensical.

How do you interpret dummies?

- A specific case of **$y = \alpha + \beta x$**

How do we interpret α and β if x is Yes or No values?

Dummies



- **Not an insult in this class**
  - https://en.wikipedia.org/wiki/Dummy_variable_(statistics)

- A **binary variable**; usually one that you make from a categorical variable since regressions can only take numerical inputs

How do you interpret dummies?

- *x* is allowed to be **categorical**, but the thing you input into a regression must be a **number**

How do you interpret dummies?

- *x* is allowed to be **categorical**, but the thing you input into a regression must be a **number**
- We do this by making sure x is converted to a **dummy**
  - If x can take two values (Yes or No), then x becomes a **binary variable (1 or 0)**

Regression interpretations: summarize relationship

x = millimeters of rainfall

y = umbrellas sold

$y = -19 + 0.45x$

Summarize relationship between variables:

Our model shows a positive relationship between rain and sales of umbrellas; specifically, each additional mm of rain corresponds to an extra 0.45 umbrellas we expect to be sold.

x = days (2023-08-12 to 2023-09-12)

y = # ice cream units sold

$y = 100 - 3x$

Summarize relationship between variables:

Our model shows a negative relationship between days and sales of ice cream; specifically, each additional day since Aug 12th 2023 corresponds to 3 fewer ice cream units we expect to be sold.

How do you interpret dummies?

- Summarize the same way: before, we talked about:
    - millimeters of rainfall
    - # days

How do you interpret dummies?

- Summarize the same way: before, we talked about:
  - millimeters of rainfall **(in relation to 0 mm)**
  - # days **(in relation to the first day)**

How do you interpret dummies?

- Summarize the same way: before, we talked about:
    - millimeters of rainfall **(in relation to 0 mm)**
    - # days **(in relation to the first day)**
    - **Yes (in relation to No)**

How do you interpret dummies?

- Summarize the same way: before, we talked about:
  - millimeters of rainfall **(in relation to 0 mm)**
  - # days **(in relation to the first day)**
  - **Yes (in relation to No)**
- **If rainfall increases by 1 mm, we expect umbrella sales to increase by β**

How do you interpret dummies?

- Summarize the same way: before, we talked about:
  - millimeters of rainfall (in relation to 0 mm)
  - # days (in relation to the first day)
  - Yes (in relation to No)
- **If X ~~rainfall~~ increases by 1 unit ~~mm~~, we expect umbrella sales to increase by β**

How do you interpret dummies?

- Summarize the same way: before, we talked about:
  - millimeters of rainfall (in relation to 0 mm)
  - # days (in relation to the first day)
  - Yes (in relation to No)

Binary X increasing by 1 unit just means going from 0 to 1!

- If X ~~rainfall~~ increases by 1 unit ~~mm~~, we expect umbrella sales to increase by β

How do you interpret dummies?

- Summarize the same way: before, we talked about:
  - millimeters of rainfall (in relation to 0 mm)
  - # days (in relation to the first day)
  - Yes (in relation to No)
- **If X goes from No to Yes, we expect umbrella sales to increase by β**

Regression interpretations: summarize relationship

x = millimeters of rainfall

y = umbrellas sold

y = -19 + 0.45x

Summarize relationship between variables:

Our model shows a positive relationship between rain and sales of umbrellas; specifically, each additional mm of rain corresponds to an extra 0.45 umbrellas we expect to be sold.

x = {0 if no rain, 1 if any rain}

y = umbrellas sold

y = 0.0 + 8x

Summarize relationship between variables:

____

____

____

Regression interpretations: summarize relationship

x = millimeters of rainfall

y = umbrellas sold

$y = -19 + 0.45x$

Summarize relationship between variables:

Our model shows a positive relationship between rain and sales of umbrellas; specifically, each additional mm of rain corresponds to an extra 0.45 umbrellas we expect to be sold.

x = {0 if no rain, 1 if any rain}

y = umbrellas sold

$y = 0.0 + 8x$

Summarize relationship between variables:

If there is any rain (as opposed to no rain), we expect the number of umbrellas sold to increase by 8.

How do you interpret dummies?

- What about the other interpretation q's?
  - You can only input two values to predict (either X=0, X=1)
  - This is a useful limitation to note for the "oddities" section

Regression interpretations: make predictions

x = millimeters of rainfall

y = umbrellas sold

$y = -19 + 0.45x$

Make predictions:

This model indicates that at the annual Ithaca average of 110mm of rainfall, we should expect to sell 30 umbrellas.

x = {0 if no rain, 1 if any rain}

y = umbrellas sold

$y = 0.0 + 8x$

Make predictions:

Regression interpretations: make predictions

x = millimeters of rainfall

y = umbrellas sold

$y = -19 + 0.45x$

Make predictions:

This model indicates that at the annual Ithaca average of 110mm of rainfall, we should expect to sell 30 umbrellas.

x = {0 if no rain, 1 if any rain}

y = umbrellas sold

$y = 0.0 + 8x$

Make predictions:

If there is no rain, the model predicts that 0 umbrellas will be sold.

If there is rain, the model predicts that 8 umbrellas will be sold.

Regression interpretations: note oddities

x = millimeters of rainfall

y = umbrellas sold

y = -19 + 0.45x

Inspect outliers and other oddities:

We expect this model to hold for rainfall amounts between 80-170mm, but cannot extrapolate further.

x = {0 if no rain, 1 if any rain}

y = umbrellas sold

y = 0.0 + 8x

Inspect outliers and other oddities:

Regression interpretations: note oddities

x = millimeters of rainfall

y = umbrellas sold

y = -19 + 0.45x

Inspect outliers and other oddities:

We expect this model to hold for rainfall amounts between 80-170mm, but cannot extrapolate further.

x = {0 if no rain, 1 if any rain}

y = umbrellas sold

y = 0.0 + 8x

Inspect outliers and other oddities:

The model only holds for binary values of x, which may not be reflective of real life (e.g., what if it only rains for a bit?).

The model will only ever predict two possible values of umbrellas sold (either 0 or 8) and will never predict anything else.

Do dummies need to be binary?

- y = umbrellas sold, x = is_raining
- y ~ x

Do dummies need to be binary?

- **y = umbrellas sold, x = is_raining**
- **y ~ x**

- x only has two values (True or False). What if we convert x from binary {0, 1} to "*binary*" {1, 2}?
    - Is raining → x=2
    - Is NOT raining → x=1

A model with one binary input

Type	x	Equation $\hat{y} = \alpha + \beta x$	Simplified
Raining	1	$\hat{y} = \alpha + \beta*1$	$\hat{y} = \alpha + \beta$
Not raining	0	$\hat{y} = \alpha + \beta*0$	$\hat{y} = \alpha$

A model with one binary input

Type	x	Equation $\hat{y} = \alpha + \beta x$	Simplified
Raining	1	$\hat{y} = \alpha + \beta*1$	$\hat{y} = \alpha + \beta$
Not raining	0	$\hat{y} = \alpha + \beta*0$	$\hat{y} = \alpha$

α is the predicted
umbrellas sold when
it's not raining

A model with one binary input

Type	x	Equation $\hat{y} = \alpha + \beta x$	Simplified
Raining	1	$\hat{y} = \alpha + \beta * 1$	$\hat{y} = \alpha + \beta$
Not raining	0	$\hat{y} = \alpha + \beta * 0$	$\hat{y} = \alpha$

β is the *difference* in # umbrellas sold when raining vs. not raining

A model with one binary input

Type	x	Equation $\hat{y} = \alpha + \beta x$	Simplified
Raining	1	$\hat{y} = \alpha + \beta * 1$	$\hat{y} = \alpha + \beta$
Not raining	0	$\hat{y} = \alpha + \beta * 0$	$\hat{y} = \alpha$

How does this table change if we code not raining = 1, raining = 2?

A model with one binary input

Type	**x**	**Equation** $\hat{y} = \alpha + \beta x$	**Simplified**
Raining	2	$\hat{y} = \alpha + \beta*2$	$\hat{y} = \alpha + 2\beta$
Not raining	1	$\hat{y} = \alpha + \beta*1$	$\hat{y} = \alpha + \beta$

Now, $\alpha+ \beta$ is the predicted # umbrellas sold when it's not raining

A model with one binary input

Type	x	Equation $\hat{y} = \alpha + \beta x$	Simplified
Raining	2	$\hat{y} = \alpha + \beta * 2$	$\hat{y} = \alpha + 2\beta$
Not raining	1	$\hat{y} = \alpha + \beta * 1$	$\hat{y} = \alpha + \beta$

But, β is **still** the *difference* in # umbrellas sold when raining vs. not raining!

A model with one binary input

Type	x	Equation $\hat{y} = \alpha + \beta x$	Simplified
Raining	2	$\hat{y} = \alpha + \beta * 2$	$\hat{y} = \alpha + 2\beta$
Not raining	1	$\hat{y} = \alpha + \beta * 1$	$\hat{y} = \alpha + \beta$

Regardless of whether x is mapped to {0,1} or {1,2},
β still has the same interpretation:
if it's raining, predicted # umbrellas ŷ will be **β more** than if it's not raining

A model with one binary input

Type	x	Equation $\hat{y} = \alpha + \beta x$	Simplified
Raining	2	$\hat{y} = \alpha + \beta * 2$	$\hat{y} = \alpha + 2\beta$
Not raining	1	$\hat{y} = \alpha + \beta * 1$	$\hat{y} = \alpha + \beta$

Can we set a new value for the intercept in this second model so that we get the same $\hat{y}$'s as the first {0,1} model?

Can we get the same outputs?

x in {0, 1} Scenario	α	β	ŷ
ŷ when raining	4	2	4 + (2)*1 = **6**
ŷ when not raining			4 + (2)*0 = **4**

x in {1, 2} Scenario	α	β	ŷ
ŷ when raining	?	2	
ŷ when not raining			

We want these values to be the same

Can we get the same outputs?

x in {0, 1} Scenario	α	β	ŷ
ŷ when raining	4	2	4 + (2)*1 = 6
ŷ when not raining			4 + (2)*0 = 4

x in {1, 2} Scenario	α	β	ŷ
ŷ when raining	2	2	2+(2)*2 = 6
ŷ when not raining			2+(2)*1 = 4

We want these values to be the same

Can we get the same outputs?

The new intercept α to get the same outcome ŷ's can be calculated by subtracting **β from the first model's α**

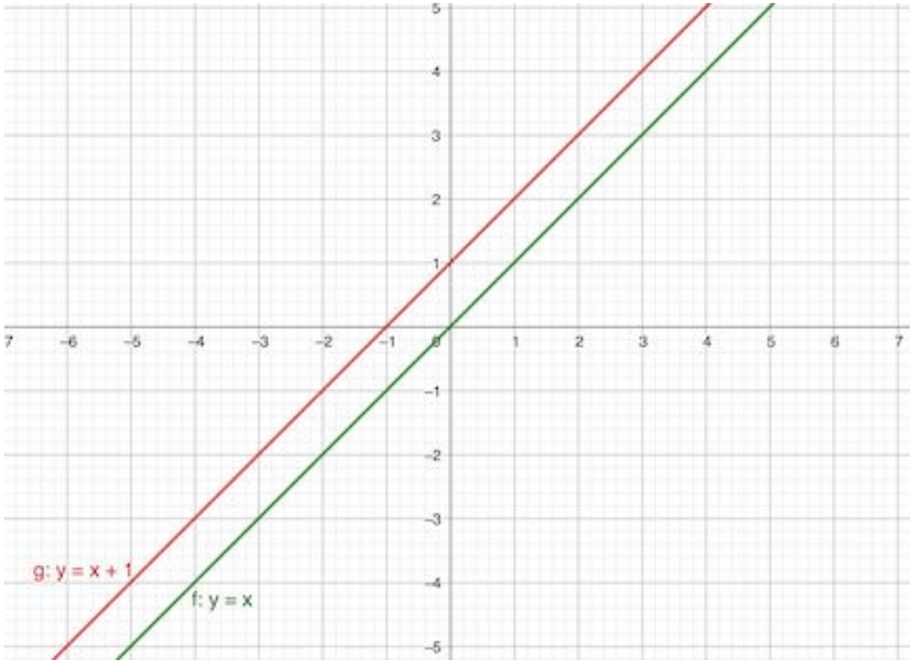
x in {0, 1} Scenario	α	β	ŷ
ŷ when raining	4	2	4 + (2)*1 = **6**
ŷ when not raining			4 + (2)*0 = **4**

x in {1, 2} Scenario	α	β	ŷ
ŷ when raining	**2 = 4 - 2**	2	**2+(2)*2 = 6**
ŷ when not raining			**2+(2)*1 = 4**

Do dummies need to be binary?

- x only has two values (True or False). What if we convert x from binary {0, 1} to "*binary*" {1, 2}?
  - Is raining → x=1
  - Is NOT raining → x=2
- **Changing to {1,2} is fine because this just shifts the intercept α**

Remapping x’s linearly just shifts a



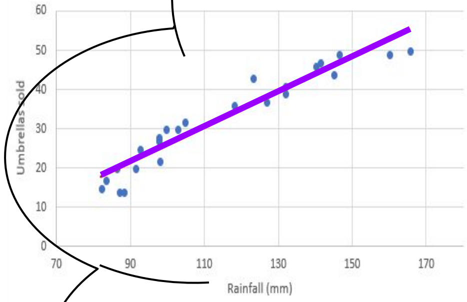
https://study.com/academy/lesson/horizontal-vertical-shifts-of-linear-functions.html

1 min break + attendance

tinyurl.com/mv9sa6sj

Regression notation

- We have a set of **points** that we want to draw a linear regression line through. That line will have form $y = \alpha + \beta x$
    - (Today, let's assume I'm just giving you a regression line, so α and β are known)



How do we find α and β?

- So far: how do we interpret regressions when they're handed to us?

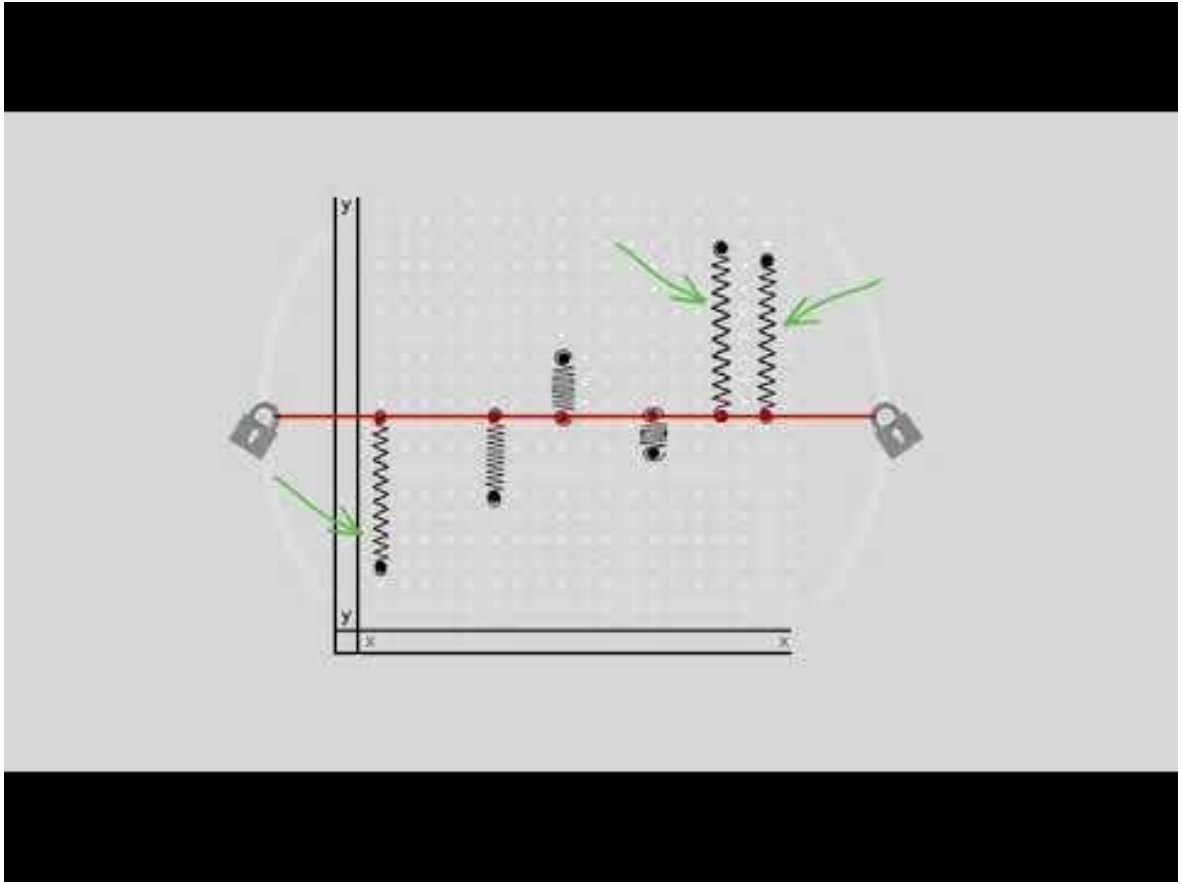
How do we find α and β?

- So far: how do we interpret regressions when they're handed to us?

- Now: how do we actually figure out the regression line **$y = \alpha + \beta x$**?
  - Want to get α, β that "best" fit the data

How do we find α and β? (Intuition)

- **Want to find α, β that "best" fit the data**
    - How do we determine what's "best"?
    - We want there to be minimal error across our points *i*
    - Physical intuition: what if each point has a spring attached to it?



https://www.jmp.com/en_us/statistics-knowledge-portal/what-is-regression/the-method-of-least-squares.html

How do we find α and β?

- **Want to find α, β that "best" fit the data**
    - We want there to be minimal error (i.e., least tension on springs) across our points *i*

How do we find α and β?

- **Want to find α, β that "best" fit the data**
  - We want there to be minimal error (i.e., least tension on springs) across our points *i*
  - But sometimes $\boldsymbol{\varepsilon_i}$ is positive, sometimes negative. Both are bad if values are big!
    - **Does this sound familiar?**

INFO2950_Lec3_20220829

File Edit View Insert Format Slide Arrange Tools Add-ons Help Last edit was 12 days ago

Means explained

- We know that the **mean** minimizes the sum of “squared distances”
  - $\sum(x-\mu)^2$

How do we find α and β?

- **Want to find α, β that "best" fit the data**
  - How do we determine what's "best"?
  - We want there to be minimal error across our points *i*
  - We want to **minimize our sum of squared error**
    - **Ever heard of "ordinary least-squares"?**

Regression notation

- We have a model: $y_i = \alpha + \beta x_i + \varepsilon_i$

Regression notation

- We have a model: $y_i = \alpha + \beta x_i + \varepsilon_i$
- We defined our prediction: $\hat{y}_i = \alpha + \beta x_i$
- Prediction error $\varepsilon_i = y_i - \hat{y}_i$
  (a.k.a. residual error)

Regression notation

- We have a model: $y_i = \alpha + \beta x_i + \varepsilon_i$
- We defined our prediction: $\hat{y}_i = \alpha + \beta x_i$
- Prediction error $\varepsilon_i = y_i - \hat{y}_i$
  (a.k.a. residual error)
- Squared prediction error $\varepsilon_i^2 = (y_i - \hat{y}_i)^2$

Regression notation

$y_3 = 14$

$\hat{y}_3 = -19 + 0.45*87$

$\varepsilon_3 = -6.15$

$\varepsilon_3^2 \approx 37.8$

- We have a model: $y_i = \alpha + \beta x_i + \varepsilon_i$
- We defined our prediction: $\hat{y}_i = \alpha + \beta x_i$
- Prediction error $\varepsilon_i = y_i - \hat{y}_i$
- Squared prediction error $\varepsilon_i^2 = (y_i - \hat{y}_i)^2$

How do we find α and β?

- **Take our sum (for i = 1 to n) of squared error:**

$$Q(\alpha, \beta) = \sum_{i=1}^{n} \hat{\varepsilon}_i^{\,2}$$

How do we find α and β?

- Take our sum (for i = 1 to n) of squared error:

$$Q(\alpha, \beta) = \sum_{i=1}^{n} \hat{\varepsilon}_i^{\,2} = \sum_{i=1}^{n} (y_i - \alpha - \beta x_i)^2$$

How do we find α and β?

- Take our sum (for i = 1 to n) of squared error:

$$Q(\alpha, \beta) = \sum_{i=1}^{n} \hat{\varepsilon}_i^{\,2} = \sum_{i=1}^{n} (y_i \boxed{- \alpha - \beta x_i})^2$$

How do we find α and β?

- Take our sum (for i = 1 to n) of squared error:

$$Q(\alpha, \beta) = \sum_{i=1}^{n} \hat{\varepsilon}_i^{\,2} = \sum_{i=1}^{n} (y_i \boxed{- \alpha - \beta x_i})^2$$

$-\hat{y}_i$

How do we find α and β?

- Take our sum (for i = 1 to n) of squared error:

$$Q(\alpha, \beta) = \sum_{i=1}^{n} \hat{\varepsilon}_i^{\,2} = \sum_{i=1}^{n} (y_i - \alpha - \beta x_i)^2$$

- To find the values of α, β that minimize Q

How do we find α and β?

- Take our sum (for i = 1 to n) of squared error:

$$Q(\alpha, \beta) = \sum_{i=1}^{n} \hat{\varepsilon}_i^{\,2} = \sum_{i=1}^{n} (y_i - \alpha - \beta x_i)^2$$

- To find the values of α, β that minimize Q
  - Take the derivative of Q (with respect to each of α, β) and set it to 0

Value of α minimizing Q?

$$Q = \sum_{i=1}^{n} (y_i - \alpha - \beta \cdot x_i)^2$$

Derivative w.r.t. α: $\frac{dQ}{d\alpha} = \sum_{i=1}^{n} (-2 \cdot (y_i - \hat{\alpha} - \hat{\beta} \cdot x_i)^2)$

Value of α minimizing Q?

Alpha and beta get hats when we're referring to the specific values at which Q is minimized

$$Q = \sum_{i=1}^{n}(y_i - \alpha - \beta \cdot x_i)^2$$

$$\frac{dQ}{d\alpha} = \sum_{i=1}^{n}(-2 \cdot (y_i - \hat{\alpha} - \hat{\beta} \cdot x_i)^2)$$

Value of α minimizing Q?

$$Q = \sum_{i=1}^{n} (y_i - \alpha - \beta \cdot x_i)^2$$

Solve for α:

$$\frac{dQ}{d\alpha} = \boxed{\sum_{i=1}^{n} (-2 \cdot (y_i - \hat{\alpha} - \hat{\beta} \cdot x_i)^2) = 0}$$

Value of α minimizing Q?

Solve for α: $\sum_{i=1}^{n}(-2 \cdot (y_i - \hat{\alpha} - \hat{\beta} \cdot x_i)^2) = 0$

Remember summation rules!

∑(2*x)=2*∑(x)
∑(x+y) = ∑(x)+∑(y)

Value of α minimizing Q?

Solve for α: $\sum_{i=1}^{n}(-2 \cdot (y_i - \hat{\alpha} - \hat{\beta} \cdot x_i)^2) = 0$

Divide by 2, split the summation $\sum_{i=1}^{n}(y_i - \hat{\beta} \cdot x_i) = \sum_{i=1}^{n}(\hat{\alpha})$

Value of α minimizing Q?

Solve for α: $\sum_{i=1}^{n}(-2\cdot(y_i-\hat{\alpha}-\hat{\beta}\cdot x_i)^2)=0$

$$\sum_{i=1}^{n}(y_i-\hat{\beta}\cdot x_i)=\sum_{i=1}^{n}(\hat{\alpha})$$

Expand out the summations

$$\sum_{i=1}^{n}(y_i)-\hat{\beta}\sum_{i=1}^{n}(x_i)=\hat{\alpha}\cdot n$$

Value of α minimizing Q?

Solve for α: $\sum_{i=1}^{n}(-2\cdot(y_i-\hat{\alpha}-\hat{\beta}\cdot x_i)^2)=0$

$$\sum_{i=1}^{n}(y_i-\hat{\beta}\cdot x_i)=\sum_{i=1}^{n}(\hat{\alpha})$$

$$\sum_{i=1}^{n}(y_i)-\hat{\beta}\sum_{i=1}^{n}(x_i)=\hat{\alpha}\cdot n$$

Isolate α-hat $\hat{\alpha}=(\sum_{i=1}^{n}(y_i))/n-(\hat{\beta}\sum_{i=1}^{n}(x_i))/n$

Can you rewrite this in terms of **x̄** (mean of x) and/or **ȳ** (mean of y)?

$$\hat{\alpha} = \boxed{(\sum_{i=1}^{n}(y_i))/n} - \boxed{(\hat{\beta}\sum_{i=1}^{n}(x_i))/n}$$

Value of α minimizing Q?

Solve for α: $\sum_{i=1}^{n}(-2 \cdot (y_i - \hat{\alpha} - \hat{\beta} \cdot x_i)^2) = 0$

$$\sum_{i=1}^{n}(y_i - \hat{\beta} \cdot x_i) = \sum_{i=1}^{n}(\hat{\alpha})$$

$$\sum_{i=1}^{n}(y_i) - \hat{\beta}\sum_{i=1}^{n}(x_i) = \hat{\alpha} \cdot n$$

$$\hat{\alpha} = (\sum_{i=1}^{n}(y_i))/n - (\hat{\beta}\sum_{i=1}^{n}(x_i))/n$$

Simplify using mean definitions! $\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}$

Value of α minimizing Q?

Solve for α: $\sum_{i=1}^{n}(-2 \cdot (y_i - \hat{\alpha} - \hat{\beta} \cdot x_i)^2) = 0$

$$\sum_{i=1}^{n}(y_i - \hat{\beta} \cdot x_i) = \sum_{i=1}^{n}(\hat{\alpha})$$

$$\sum_{i=1}^{n}(y_i) - \hat{\beta}\sum_{i=1}^{n}(x_i) = \hat{\alpha} \cdot n$$

$$\hat{\alpha} = (\sum_{i=1}^{n}(y_i))/n - (\hat{\beta}\sum_{i=1}^{n}(x_i))/n$$

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

This is the α from our regression that minimizes the sum of squared error!

Value of β minimizing Q?

Similar to finding α-hat, now we take the derivative of Q with respect to **β**. Skipping the messy math, we get:

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

Does anything look familiar?

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

Does anything look familiar?

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

N * cov(x,y)

N * var(x)

Does anything look familiar?

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

= cov(x,y) / var(x)

Can you express regression slope in terms of correlation?

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

= cov(x,y) / var(x)

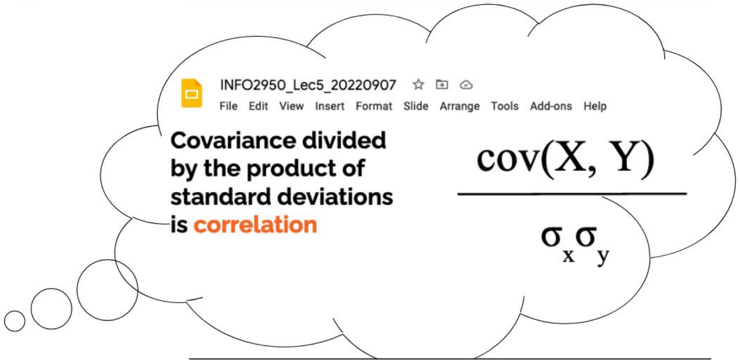
= ?

Can you express regression slope in terms of correlation?

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

= cov(x,y) / var(x)

= ?



Can you express regression slope in terms of correlation?

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

$= \mathrm{cov}(x,y) / \mathrm{var}(x) = [\mathrm{corr}(x,y) * \sigma_y \sigma_x] / (\sigma_x \sigma_x)$

$= \mathrm{corr}(x,y) * \sigma_y / \sigma_x$

If the slope for umbrellas ~ rain is negative, what do we know about the slope of rain ~ umbrellas?

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

$$= \text{cov}(x,y) / \text{var}(x)$$

$$= \text{corr}(x,y) * \sigma_y / \sigma_x$$

If the slope for umbrellas ~ rain is negative, what do we know about the slope of rain ~ umbrellas?

$\hat{\beta}$ = corr(x,y) * σ_y / σ_x < 0

If the slope for umbrellas ~ rain is negative, what do we know about the slope of rain ~ umbrellas?

$$\hat{\beta} = \boxed{\text{corr}(x,y)} * \sigma_y / \sigma_x < 0$$

corr(x,y) = corr(y,x)

If the slope for umbrellas ~ rain is negative, what do we know about the slope of rain ~ umbrellas?

$\hat{\beta}$ = corr(x,y) * $\boxed{\sigma_y / \sigma_x}$ < 0

Standard deviations cannot be negative! (sqrt(Var))

If the slope for umbrellas ~ rain is negative, what do we know about the slope of rain ~ umbrellas?

$\hat{\beta}$ = corr(x,y) * σ_y / σ_x **< 0**

The β-hat for rain ~ umbrellas will be corr(y,x) * sd(x) / sd(y), which will also be negative since none of the signs change in the constituent parts

Under what circumstance would reversing input and output variables result in the same slope?

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

= cov(x,y) / var(x)

= corr(x,y) * σ_y / σ_x

Under what circumstance would reversing input and output variables result in the same slope?

Want to find when

$$\text{corr}(x,y) * \sigma_y / \sigma_x = \text{corr}(y,x) * \sigma_x / \sigma_y$$

Under what circumstance would reversing input and output variables result in the same slope?

Want to find when

$\cancel{\text{corr}(x,y)} * \sigma_y / \sigma_x = \cancel{\text{corr}(y,x)} * \sigma_x / \sigma_y$

The correlation terms cancel since corr is symmetric!

Under what circumstance would reversing input and output variables result in the same slope?

Want to find when

$\cancel{\text{corr(x,y)}} * \sigma_y / \sigma_x = \cancel{\text{corr(y,x)}} * \sigma_x / \sigma_y$

$\rightarrow \sigma_y * \sigma_y = \sigma_x * \sigma_x$

Under what circumstance would reversing input and output variables result in the same slope?

Want to find when

$\cancel{corr(x,y)} * \sigma_y / \sigma_x = \cancel{corr(y,x)} * \sigma_x / \sigma_y$

$\rightarrow \sigma_y * \sigma_y = \sigma_x * \sigma_x$

$\rightarrow$ var(y) = var(x)

Under what circumstance would reversing input and output variables result in the same slope?

Want to find when

$\cancel{\text{corr}(x,y)} * \sigma_y / \sigma_x = \cancel{\text{corr}(y,x)} * \sigma_x / \sigma_y$

$\rightarrow \sigma_y * \sigma_y = \sigma_x * \sigma_x$

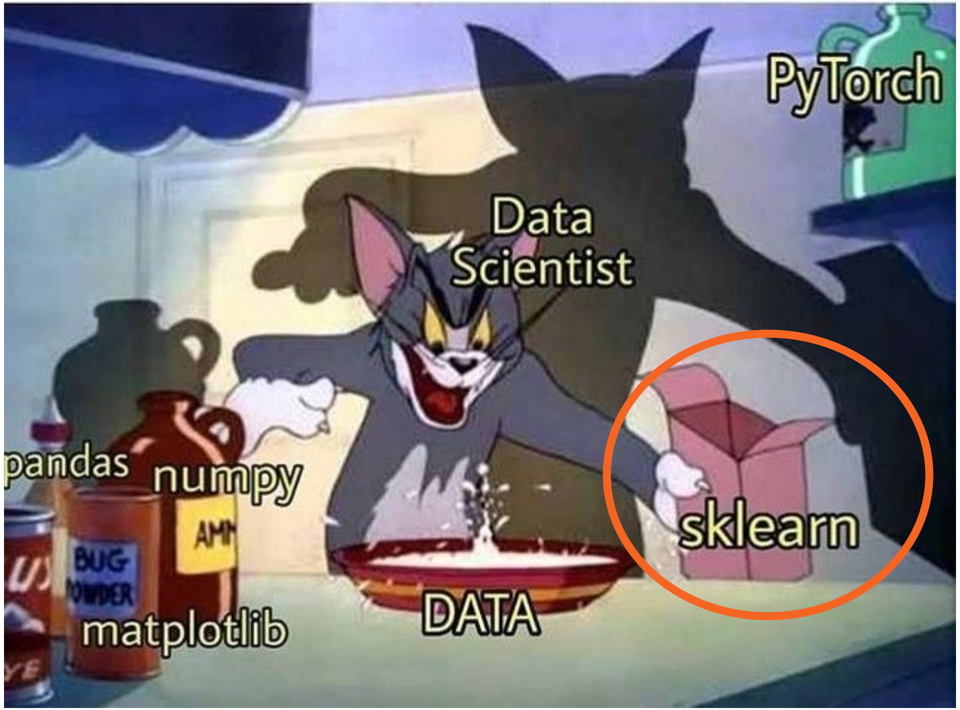
$\rightarrow$ var(y) = var(x)

Flipping input/output only gives the same slope if they have the same variances!

Takeaways from regression math

- For “ordinary least squares” regression: slope and intercept are values calculated from minimizing the squared prediction error
- The resulting regression line is closely related to the covariance or the correlation between the input (x) and output (y) variables

1 min break



Up next!

https://in.pinterest.com/pin/454652524889134329/

How do we actually run a regression in Python?

- You have a df with two columns: x and y
- You “input” to Python both the df input(x) and df output(y) and run a regression y~x

How do we actually run a regression in Python?

- You have a df with two columns: x and y
- You “input” to Python both the df input(x) and df output(y) and run a regression y~x
- Python “outputs” estimates for α-hat and β-hat
- How does Python do this?

scikit-learn

- We use a package!
- scikit-learn is a fundamental tool for everything from basic data science to advanced ML
- install scikit-learn via Anaconda
- `from sklearn.linear_model import LinearRegression`

LinearRegression()

```
x = np.array([6, 16, 26, 36, 46, 56]).reshape((-1, 1))
y = np.array([4, 23, 10, 12, 22, 35])

# Create an instance of a linear regression model and fit it to the data with the fit() function:
model = LinearRegression().fit(x, y)

# Print the Intercept:
print('intercept:', model.intercept_)

# Print the Slope:
print('slope:', model.coef_)

# Predict a Response and print it:
y_pred = model.predict(x)
print('Predicted response:', y_pred, sep='\n')
```

https://www.activestate.com/resources/quick-reads/how-to-run-linear-regressions-in-python-scikit-learn/

LinearRegression()

Give scikit-learn your data (inputs and outputs)

```
x = np.array([6, 16, 26, 36, 46, 56]).reshape((-1, 1))
y = np.array([4, 23, 10, 12, 22, 35])

# Create an instance of a linear regression model and fit it to the data with the fit() function:
model = LinearRegression().fit(x, y)

# Print the Intercept:
print('intercept:', model.intercept_)

# Print the Slope:
print('slope:', model.coef_)

# Predict a Response and print it:
y_pred = model.predict(x)
print('Predicted response:', y_pred, sep='\n')
```

https://www.activestate.com/resources/quick-reads/how-to-run-linear-regressions-in-python-scikit-learn/

LinearRegression() dimensions

x is a *nx1* vector

```
x
✓ 0.3s

array([[ 6],
       [16],
       [26],
       [36],
       [46],
       [56]])
```

y is a *n* vector

```
y
✓ 0.2s

array([ 4, 23, 10, 12, 22, 35])
```

LinearRegression()

```
x = np.array([6, 16, 26, 36, 46, 56]).reshape((-1, 1))
y = np.array([4, 23, 10, 12, 22, 35])

# Create an instance of a linear regression model and fit it to the data with the fit() function:
model = LinearRegression().fit(x, y)

# Print the Intercept:
print('intercept:', model.intercept_)

# Print the Slope:
print('slope:', model.coef_)

# Predict a Response and print it:
y_pred = model.predict(x)
print('Predicted response:', y_pred, sep='\n')
```

Fit your linear regression on (input, output) data

https://www.activestate.com/resources/quick-reads/how-to-run-linear-regressions-in-python-scikit-learn/

LinearRegression()

```
x = np.array([6, 16, 26, 36, 46, 56]).reshape((-1, 1))
y = np.array([4, 23, 10, 12, 22, 35])

# Create an instance of a linear regression model and fit it to the data with the fit() function:
model = LinearRegression().fit(x, y)

# Print the Intercept:
print('intercept:', model.intercept_)

# Print the Slope:
print('slope:', model.coef_)

# Predict a Response and print it:
y_pred = model.predict(x)
print('Predicted response:', y_pred, sep='\n')
```

Step 1: create a LinearRegression object

https://www.activestate.com/resources/quick-reads/how-to-run-linear-regressions-in-python-scikit-learn/

LinearRegression()

```
x = np.array([6, 16, 26, 36, 46, 56]).reshape((-1, 1))
y = np.array([4, 23, 10, 12, 22, 35])

# Create an instance of a linear regression model and fit it to the data with the fit() function:
model = LinearRegression().fit(x, y)

# Print the Intercept:
print('intercept:', model.intercept_)

# Print the Slope:
print('slope:', model.coef_)

# Predict a Response and print it:
y_pred = model.predict(x)
print('Predicted response:', y_pred, sep='\n')
```

Step 2: immediately call the fit() function with a DataFrame as input and a Series as output.
This function returns the regression object.

https://www.activestate.com/resources/quick-reads/how-to-run-linear-regressions-in-python-scikit-learn/

LinearRegression()

```
x = np.array([6, 16, 26, 36, 46, 56]).reshape((-1, 1))
y = np.array([4, 23, 10, 12, 22, 35])

# Create an instance of a linear regression model and fit it to the data with the fit() function:
model = LinearRegression().fit(x, y)

# Print the Intercept:
print('intercept:', model.intercept_)

# Print the Slope:
print('slope:', model.coef_)

# Predict a Response and print it:
y_pred = model.predict(x)
print('Predicted response:', y_pred, sep='\n')
```

Look at what the model spits out: a value

```
intercept: 4.026666666666667
```

LinearRegression()

```
x = np.array([6, 16, 26, 36, 46, 56]).reshape((-1, 1))
y = np.array([4, 23, 10, 12, 22, 35])

# Create an instance of a linear regression model and fit it to the data with the fit() function:
model = LinearRegression().fit(x, y)

# Print the Intercept:
print('intercept:', model.intercept_)

# Print the Slope:
print('slope:', model.coef_)

# Predict a Response and print it:
y_pred = model.predict(x)
print('Predicted response:', y_pred, sep='\n')
```

Look at what the model spits out: β value

```
slope: [0.44]
```

https://www.activestate.com/resources/quick-reads/how-to-run-linear-regressions-in-python-scikit-learn/

LinearRegression()

```
x = np.array([6, 16, 26, 36, 46, 56]).reshape((-1, 1))
y = np.array([4, 23, 10, 12, 22, 35])

# Create an instance of a linear regression model and fit it to the data with the fit() function:
model = LinearRegression().fit(x, y)

# Print the Intercept:
print('intercept:', model.intercept_)

# Print the Slope:
print('slope:', model.coef_)

# Predict a Response and print it:
y_pred = model.predict(x)
print('Predicted response:', y_pred, sep='\n')
```

```
Predicted response:
[ 6.66666667 11.06666667 15.46666667 19.86666667 24.26666667 28.66666667]
```

We can generate the regression line (i.e., for every input x, what are our y-hats?)

https://www.activestate.com/resources/quick-reads/how-to-run-linear-regressions-in-python-scikit-learn/

Another way to do OLS regression in Python: statsmodels

```
>>> import statsmodels.api as sm
>>> import numpy as np
```

https://www.statsmodels.org/stable/generated/statsmodels.regression.linear_model.OLS.html

Another way to do OLS regression in Python: statsmodels

Define your X and Y arrays (here we're grabbing the relevant columns from an existing dataset, instead of creating our own arrays)

```
>>> 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()
>>> results.params
const        10.603498
education     0.594859
dtype: float64
```

https://www.statsmodels.org/stable/generated/statsmodels.regression.linear_model.OLS.html

Another way to do OLS regression in Python: statsmodels

We have to explicitly tell statsmodels to include a coefficient α in our regression

```
>>> 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()
>>> results.params
const        10.603498
education     0.594859
dtype: float64
```

https://www.statsmodels.org/stable/generated/statsmodels.regression.linear_model.OLS.html

Another way to do OLS regression in Python: statsmodels

Again, you have to call .fit() after you define your model

```
>>> 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()
>>> results.params
const        10.603498
education     0.594859
dtype: float64
```

https://www.statsmodels.org/stable/generated/statsmodels.regression.linear_model.OLS.html

Another way to do OLS regression in Python: statsmodels

```
>>> 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()
>>> results.params
const        10.603498
education     0.594859
dtype: float64
```

α

β

.params shows us the OLS regression coefficients

https://www.statsmodels.org/stable/generated/statsmodels.regression.linear_model.OLS.html

Another way to do OLS regression in Python: statsmodels

One reason to use statsmodels over scikit-learn is its better regression table formatting... but more on that next week!

```
>>> 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()
>>> results.params
const        10.603498
education     0.594859
dtype: float64
```

https://www.statsmodels.org/stable/generated/statsmodels.regression.linear_model.OLS.html

Final projects

- If you added the course late, OR if you're a team willing to take on an additional teammate, please fill out the partner-finding form pinned on Ed:
    - https://docs.google.com/forms/d/e/1FAIpQLSdg1NnPCGACrAiCky2IUBWvzKLqi2MGeomC1lwXTqEuRF3k2Q/viewform

Phase 1

Due by **September 21, 11:59 p.m:**

- A link to your group's private Github repository
  - Make sure to give your TA access!
- Three ideas for datasets. For each idea, include:
  - A short description of the dataset.
  - A short discussion of the availability of the data (including links to online data sources).
- Questions for reviewers

Phase 1: Dataset example #1

- Description: Our first dataset idea is a dataset for movie reviews and ratings. These would be reviews and ratings written by critics and user reviews. We would also need additional metadata about the movie's budget, actors, and genre.
- Availability: Some datasets, like the Rotten Tomatoes dataset, have already been scraped and collected for us and contain critics' reviews and ratings. We could combine this data with user reviews scraped from websites like Letterboxd or IMDb. IMDb has a bulk data download or an API. We would need to merge datasets on the movie's name and year released, but should be able to access all of the data.

Phase 1: Dataset example #2

- Description: A second dataset idea is about coffee quality and the climate in regions where the coffee beans are grown. This dataset needs to contain some type of quality index about coffee along with information about a given location's climate, especially precipitation and temperature.
- Availability: We have found a dataset already collected from the Coffee Quality Institute with data up until 2018. We looked to the CQI's website and there was a message stating that the information should not be scraped. We would like to combine the existing data with another dataset related to amounts of precipitation in different areas, which we still need to locate.
    - **Question for reviewers** → Should we be concerned that we can't collect any more CQI data than what was already scraped?

Phase 1: Dataset example #3

- Description: Our final dataset idea is different data about NBA draft picks and how many games each team won in the following season. Ideally, this data would be up until the most recent year and as far back in NBA history as possible.
- Availability: We know that there is a website with historical draft picks and we have inspected the HTML and have found that it should be scrapeable. We found another dataset on Kaggle that has NBA tournament information from 2004 to 2020. We would need to scrape the draft picks and then find a way to link it with the NBA tournament dataset.

Reshaping dataframes: what?

- Sometimes your data will come to you in a form where you either wish the columns were rows, or vice versa

Reshaping dataframes: why?

- Sometimes your data will come to you in a form where you either wish the columns were rows, or vice versa
- Getting it into the format you want can help you do aggregations (e.g. group by, sum), mutations (make a new column using old columns), run regressions (using x and y as columns), etc.

Which is better for… Plotting? Regression?

input_x	is_high	output_y
2023-09-19	high	77
2023-09-19	low	58
2023-09-20	high	73
2023-09-20	low	55
2023-09-21	high	80
2023-09-21	low	57

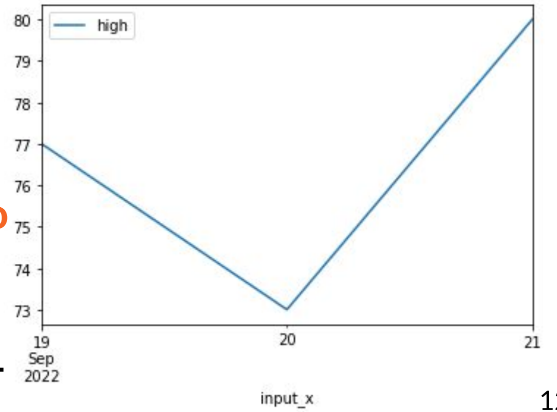
input_x	high	low
2023-09-19	77	58
2023-09-20	73	55
2023-09-21	80	57

input_x	is_high	output_y
2023-09-19	high	77
2023-09-19	low	58
2023-09-20	high	73
2023-09-20	low	55
2023-09-21	high	80
2023-09-21	low	57

input_x	high	low
2023-09-19	77	58
2023-09-20	73	55
2023-09-21	80	57

```
df.plot('input_x','high')
```

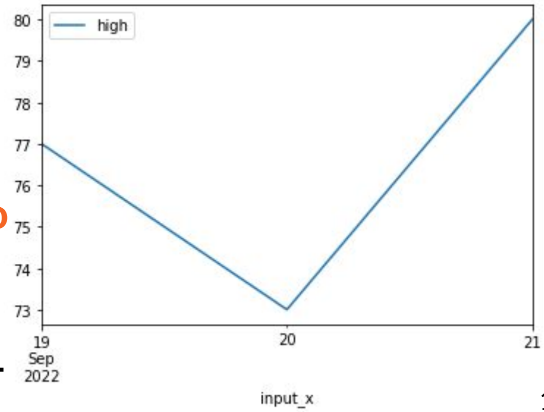
Easier to plot a subset of data, or do high ~ input_x



input_x	is_high	output_y
2023-09-19	high	77
2023-09-19	low	58
2023-09-20	high	73
2023-09-20	low	55
2023-09-21	high	80
2023-09-21	low	57

input_x	high	low
2023-09-19	77	58
2023-09-20	73	55
2023-09-21	80	57

```
df.plot('input_x','high')
```

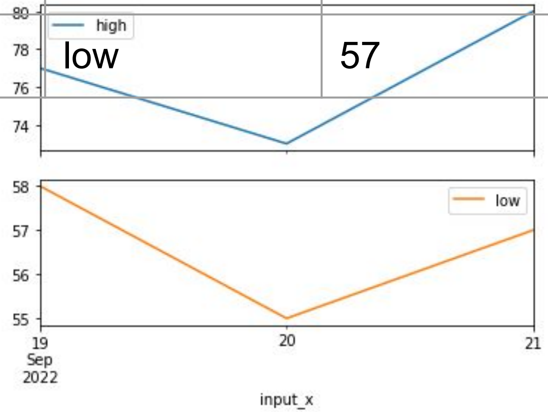


Easier to plot a subset of data, or do high ~ input_x

*note: x's need to be numeric in a regression, so you'd likely convert input_x to be # days since a certain date

input_x	is_high	output_y
2023-09-19	high	77
2023-09-19	low	58
2023-09-20	high	73
2023-09-20	low	55
2023-09-21	high	80
2023-09-21	low	57

input_x	high	low
2023-09-19	77	58
2023-09-20	73	55
2023-09-21	80	57



Easier to plot grouped data, or do multivariable regression

```
pd.pivot_table(df.reset_index(),
        index='input_x', columns='is_high', values='output_y'
        ).plot(subplots=True)
```

When to reshape?

input_x	is_high	output_y
2023-09-19	high	77
2023-09-19	low	58
2023-09-20	high	73
2023-09-20	low	55
2023-09-21	high	80
2023-09-21	low	57

input_x	high	low
2023-09-19	77	58
2023-09-20	73	55
2023-09-21	80	57

should be same units
for interpretability

input_x	time_of_day	output_y
2023-09-19	high	77
2023-09-19	avg	67.5
2023-09-19	low	58
2023-09-20	high	73
2023-09-20	avg	64
2023-09-20	low	55
2023-09-21	high	80
2023-09-21	avg	68.5
2023-09-21	low	57

Which one is...
“Wide” vs. “Long”*?

*long a.k.a. tidy, skinny, tall

input_x	high	low	avg
2023-09-19	77	58	67.5
2023-09-20	73	55	64.0
2023-09-21	80	57	68.5

input_x	time_of_day	output_y
2023-09-19	high	77
2023-09-19	avg	67.5
2023-09-19	low	58
2023-09-20	high	73
2023-09-20	avg	64
2023-09-20	low	55
2023-09-21	high	80
2023-09-21	avg	68.5
2023-09-21	low	57

"Long"

"Wide"

input_x	high	low	avg
2023-09-19	77	58	67.5
2023-09-20	73	55	64.0
2023-09-21	80	57	68.5

Reshaping: how?

- When in doubt, Google search for how to convert from “long to wide” or “wide to long” with Pandas

Reshaping: how?

- When in doubt, Google search for how to convert from “long to wide” or “wide to long” with Pandas
- Many methods, some fancier (e.g. multiple columns at a time)

Long to wide	**Wide to long**
pivot() unstack() long_to_wide()	melt() stack() wide_to_long()

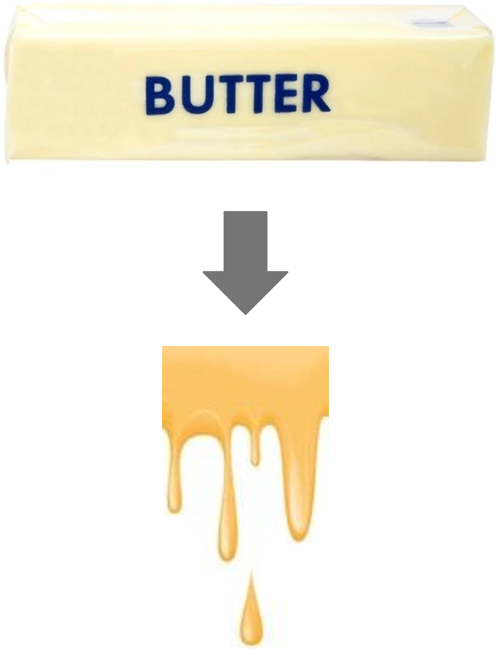
Reshaping: how?

- When in doubt, Google search for how to convert from “long to wide” or “wide to long” with Pandas
- Many methods, some fancier (e.g. multiple columns at a time)

Long to wide	**Wide to long**
pivot() unstack() long_to_wide()	melt() stack() wide_to_long()

https://dev.to/stevenmcgown/data-science-zero-to-hero-12-pandas-5hb0

Reshaping: how?



- When in doubt, Google search for how to convert from “long to wide” or “wide to long” with Pandas
- Many methods, some fancier (e.g. multiple columns at a time)

Long to wide	**Wide to long**
pivot() unstack() long_to_wide()	$\boxed{\text{melt()}}$ stack() wide_to_long()

https://www.shutterstock.com/search/stick-of-butter , https://stock.adobe.com/hk/search/images?k=melted+butter

Start with ‘wide_df’

input_x	high	low
2023-09-19	77	58
2023-09-20	73	55
2023-09-21	80	57

To make this long, what do we use: melt or pivot?

‘wide_df’ to long: melt

input_x	high	low
2023-09-19	77	58
2023-09-20	73	55
2023-09-21	80	57

input_x	is_high	output_y
2023-09-19	high	77
2023-09-19	low	58
2023-09-20	high	73
2023-09-20	low	55
2023-09-21	high	80
2023-09-21	low	57

id_vars = a list of column names that should remain the same

value_vars = a list of column names that are going to be reshaped

‘wide_df’ to long: melt

input_x	high	low
2023-09-19	77	58
2023-09-20	73	55
2023-09-21	80	57

input_x	is_high	output_y
2023-09-19	high	77
2023-09-19	low	58
2023-09-20	high	73
2023-09-20	low	55
2023-09-21	high	80
2023-09-21	low	57

id_vars = a list of column names that should remain the same

value_vars = a list of column names that are going to be reshaped

‘wide_df’ to long: melt

input_x	high	low
2023-09-19	77	58
2023-09-20	73	55
2023-09-21	80	57

input_x	is_high	output_y
2023-09-19	high	77
2023-09-19	low	58
2023-09-20	high	73
2023-09-20	low	55
2023-09-21	high	80
2023-09-21	low	57

id_vars = a list of column names that should remain the same

value_vars = a list of column names that are going to be reshaped

‘wide_df’ to long: melt

input_x	high	low
2022-09-19	77	58
2022-09-20	73	55
2022-09-21	80	57

id_vars = a list of column names that should remain the same

value_vars = a list of column names that are going to be reshaped

```
wide_df.melt(
   id_vars = ‘input_x’,
   value_vars = [‘high’, ‘low’]
   )
```

‘wide_df’ to long: melt

input_x	high	low
2022-09-19	77	58
2022-09-20	73	55
2022-09-21	80	57

	input_x	variable	value
0	2022-09-19	high	77
1	2022-09-20	high	73
2	2022-09-21	high	80
3	2022-09-19	low	58
4	2022-09-20	low	55
5	2022-09-21	low	57

id_vars = a list of column names that should remain the same

value_vars = a list of column names that are going to be reshaped

```
wide_df.melt(
   id_vars = ‘input_x’,
   value_vars = [‘high’, ‘low’]
   )
```

‘wide_df’ to long: melt

input_x	high	low
2022-09-19	77	58
2022-09-20	73	55
2022-09-21	80	57

	input_x	variable	value
0	2022-09-19	high	77
1	2022-09-20	high	73
2	2022-09-21	high	80
3	2022-09-19	low	58
4	2022-09-20	low	55
5	2022-09-21	low	57

Default column names:

- “variable” contains the column name used to be (in the list of value_vars)
- “value” is the contents of the cells in the columns within value_vars
- You can change these column names in the same melt() statement using var_name and value_name options

‘wide_df’ to long: melt

input_x	high	low
2022-09-19	77	58
2022-09-20	73	55
2022-09-21	80	57

```
tall_df = wide_df.melt(
              id_vars = 'input_x',
              value_vars = ['high','low'],
              var_name = 'is_high',
              value_name = 'output_y')
tall_df.sort_values(by=['input_x','is_high'])
```

input_x	is_high	output_y
2022-09-19	high	77
2022-09-19	low	58
2022-09-20	high	73
2022-09-20	low	55
2022-09-21	high	80
2022-09-21	low	57

Start with ‘tall_df’

input_x	is_high	output_y
2022-09-19	high	77
2022-09-19	low	58
2022-09-20	high	73
2022-09-20	low	55
2022-09-21	high	80
2022-09-21	low	57

To make this wide, what do we use: melt or pivot?

‘tall_df’ to wide: pivot

input_x	is_high	output_y
2022-09-19	high	77
2022-09-19	low	58
2022-09-20	high	73
2022-09-20	low	55
2022-09-21	high	80
2022-09-21	low	57

input_x	high	low
2022-09-19	77	58
2022-09-20	73	55
2022-09-21	80	57

index = (a list of) column names that should remain the same

columns = (a list of) variable to split into separate columns

values = values to fill the new columns

‘tall_df’ to wide: pivot

input_x	is_high	output_y
2022-09-19	high	77
2022-09-19	low	58
2022-09-20	high	73
2022-09-20	low	55
2022-09-21	high	80
2022-09-21	low	57

input_x	high	low
2022-09-19	77	58
2022-09-20	73	55
2022-09-21	80	57

index = (a list of) column names that should remain the same

columns = (a list of) variable to split into separate columns

values = values to fill the new columns

'tall_df' to wide: pivot

input_x	is_high	output_y
2022-09-19	high	77
2022-09-19	low	58
2022-09-20	high	73
2022-09-20	low	55
2022-09-21	high	80
2022-09-21	low	57

input_x	high	low
2022-09-19	77	58
2022-09-20	73	55
2022-09-21	80	57

index = (a list of) column names that should remain the same

columns = (a list of) variable to split into separate columns

values = values to fill the new columns

‘tall_df’ to wide: pivot

input_x	is_high	output_y
2022-09-19	high	77
2022-09-19	low	58
2022-09-20	high	73
2022-09-20	low	55
2022-09-21	high	80
2022-09-21	low	57

input_x	high	low
2022-09-19	77	58
2022-09-20	73	55
2022-09-21	80	57

index = (a list of) column names that should remain the same

columns = (a list of) variable to split into separate columns

values = values to fill the new columns

'tall_df' to wide: pivot

input_x	is_high	output_y
2022-09-19	high	77
2022-09-19	low	58
2022-09-20	high	73
2022-09-20	low	55
2022-09-21	high	80
2022-09-21	low	57

is_high	high	low
input_x		
2022-09-19	77	58
2022-09-20	73	55
2022-09-21	80	57

```
tall_df.pivot(
    index = 'input_x',
    columns='is_high',
    values='output_y')
```

‘tall_df’ to wide: pivot

input_x	is_high	output_y
2022-09-19	high	77
2022-09-19	low	58
2022-09-20	high	73
2022-09-20	low	55
2022-09-21	high	80
2022-09-21	low	57

is_high	high	low
input_x		
2022-09-19	77	58
2022-09-20	73	55
2022-09-21	80	57

Pivot tables in pandas:
`wide_df.shape` still returns (3,2)
but now includes extra headers

Reshaping takeaways

- Wide to long: **melt**
- Long to wide: **pivot**
- Search for pandas documentation when you need to use reshaping functions

Course reminders

- Fill out the excused absence form on Qualtrics if you are out
- HW2 due tomorrow (Thurs)
- HW3 posted tomorrow (Thurs)
- Phase 1 due next Thursday
- **Come to Friday discussion prepared with an empty GitHub repo for your project group (make sure all teammates have access)**