

Lecture #13: Missing Data and Imputation

aka STAT109A, AC209A, CSCIE-109A

CS1090A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai and Chris Gumb



Lecture Outline

- **Dealing with Missingness**
 - Types of Missingness
 - Identifying Missingness
 - Dropping
 - Simple Imputation Methods
 - Missingness Indicator Variable
 - Model-Based Imputation

Types of Missingness

What is missing data?

Often, when data are collected, there are some missing values apparent in the dataset. This leads to a few questions to consider:

- How does this show up in pandas?
- How does sklearn handle these NaNs?
- How does this effect our modeling?
- What are the simplest ways to handle missing data?

Sources of Missingness (Examples)

Missing data can arise from various places in data:

- A survey was conducted and values were just randomly missed when being entered in the computer.
- A respondent chooses not to respond to a question like "Have you ever recreationally used opioids?".
- You decide to start collecting a new variable (due to new actions: like a pandemic) partway through the data collection of a study.
- You want to measure the speed of meteors, and some observations are just 'too quick' to be measured properly.

Types of Missingness

There are 3 major types of missingness to be concerned about:

1. **Missing Completely at Random (MCAR)** - the probability of missingness in a variable is the same for all units. Like randomly poking holes in a data set.
2. **Missing at Random (MAR)** - the probability of missingness in a variable depends only on available information (in other predictors).
3. **Missing Not at Random (MNAR)** - the probability of missingness depends on information that has not been recorded and this information also predicts the missing values.

Missing completely at random (MCAR)

Missing Completely at Random is the best-case scenario, and the easiest to handle:

- Examples: a coin is flipped to determine whether an entry is removed. Or when values were just randomly missed when being entered in the computer.
- Effect if you ignore: there is no effect on inferences.
- How to handle: lots of options, but best to impute (more on next slide).

Missing at random (MAR)

Missing at Random is still a case that can be handled.

- Example(s): men and women respond at different rates to the question, "have you ever felt harassed at work?" (and may be harassed at different rates).
- Effect if you ignore: inferences are biased, and predictions usually suffer.
- How to handle: use the information in the other predictors to build a model and **impute** a value for the missing entry.

Key: we can fix any biases by modeling and imputing the missing values based on what is observed!

Missing Not at Random (MNAR)

Missing Not at Random is the worst-case scenario, and impossible to handle properly:

- Example(s): patients drop out of a study because they experience some bad side effect that was not measured. Or cheaters are less likely to respond when asked if you've ever cheated.
- Effect if you ignore: there are major effects on inferences or predictions.
- How to handle: you can 'improve' things by dealing with it like it is MAR, but you [likely] may never completely fix the bias. Simply incorporating a **missingness indicator variable** may be the best approach (if it a predictor that is missing).

What type of missingness is present?

Can you ever tell based on your data what type of missingness is present?

It generally cannot be determined whether data really are missing at random, or whether the missingness depends on unobserved predictors or the missing data themselves. The problem is that these potential “lurking variables” are unobserved (by definition) and so can never be completely ruled out.

In practice, a model with as many predictors as possible is used so that the ‘missing at random’ assumption is reasonable.

Identifying Missingness

Do I Have Missing Data?

The first step to addressing missingness is to recognize it in your data. So, what does missingness look like in a Pandas DataFrame?

- Pandas uses Numpy's special **NaN** datatype to denote missing values.
- It is also common to see the **None** type used to signify a missing value.
- `pd.isna(df)` will return a boolean matrix with True in each index that had either NaN or None and False everywhere else.

Example: Missing Migrants Project

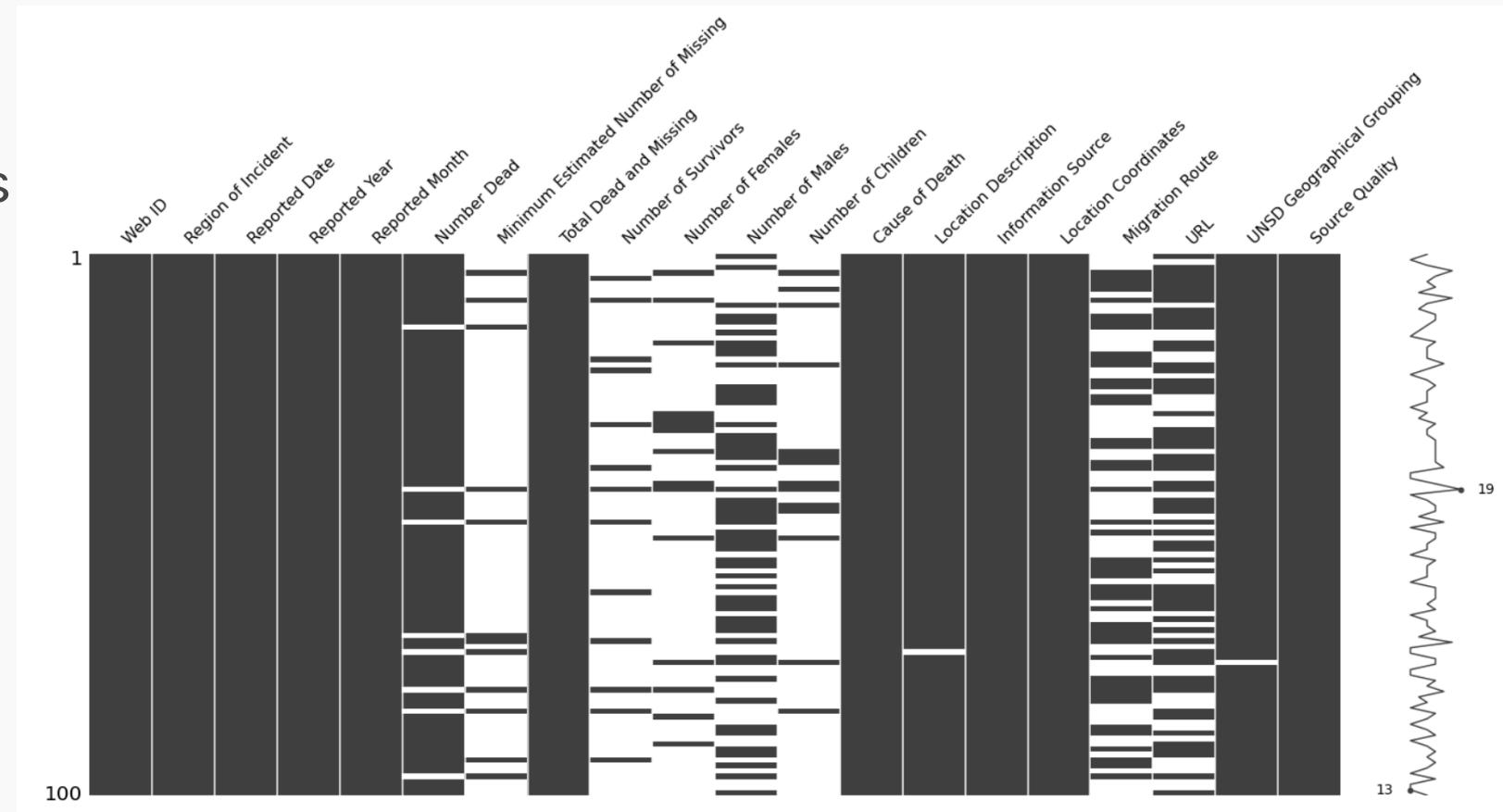
Migrants Project records since 2014 people who die in the process of migration towards an international destination, regardless of their legal status.



Example: Missing Migrants Project

```
msno.matrix(df.sample(100))
```

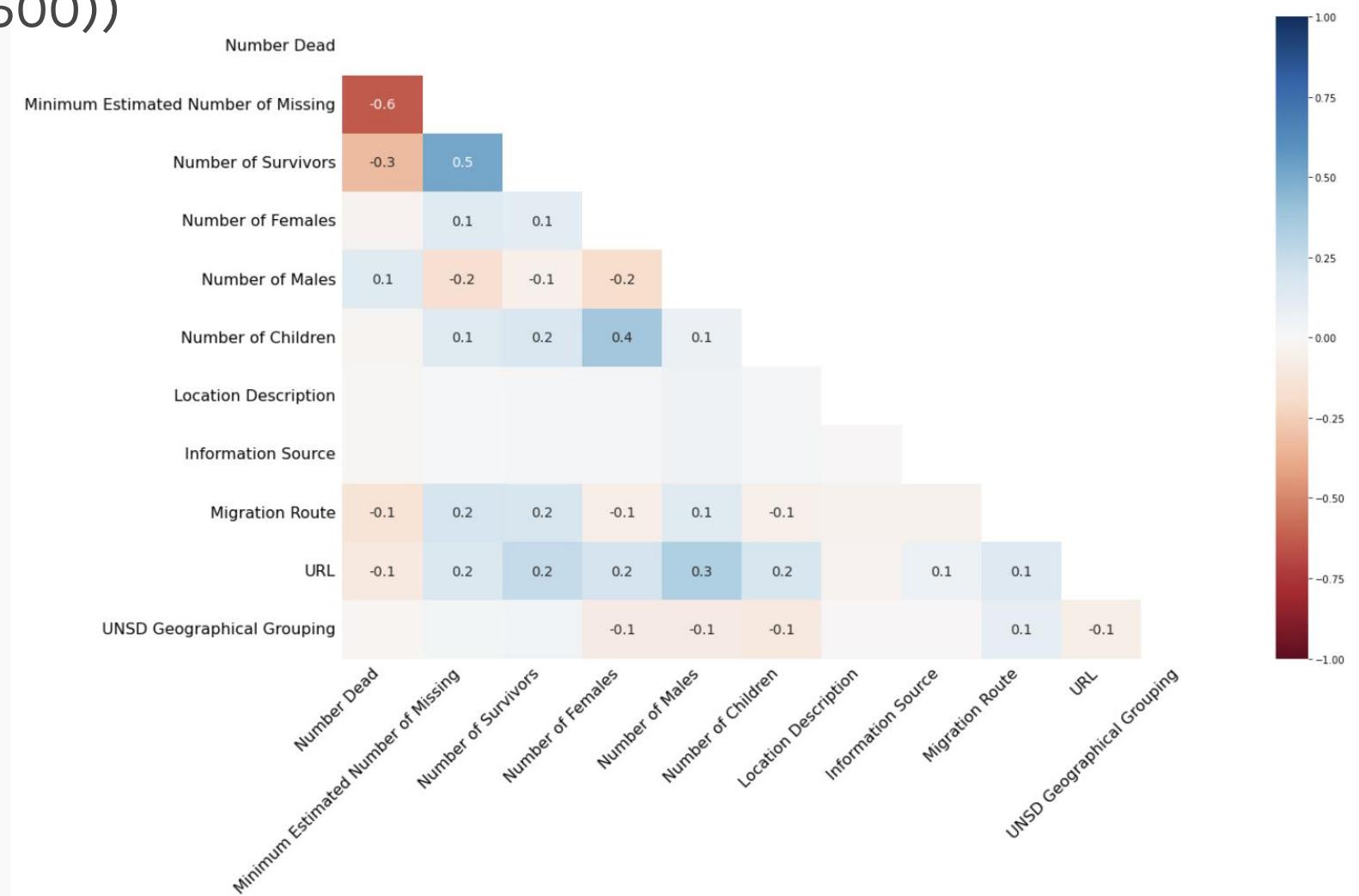
- nullity matrix: quickly visualize missing patterns
- sparkline on the right summarizes the general shape of the data completeness



Example: Missing Migrants Project

```
msno.heatmap(df.sample(500))
```

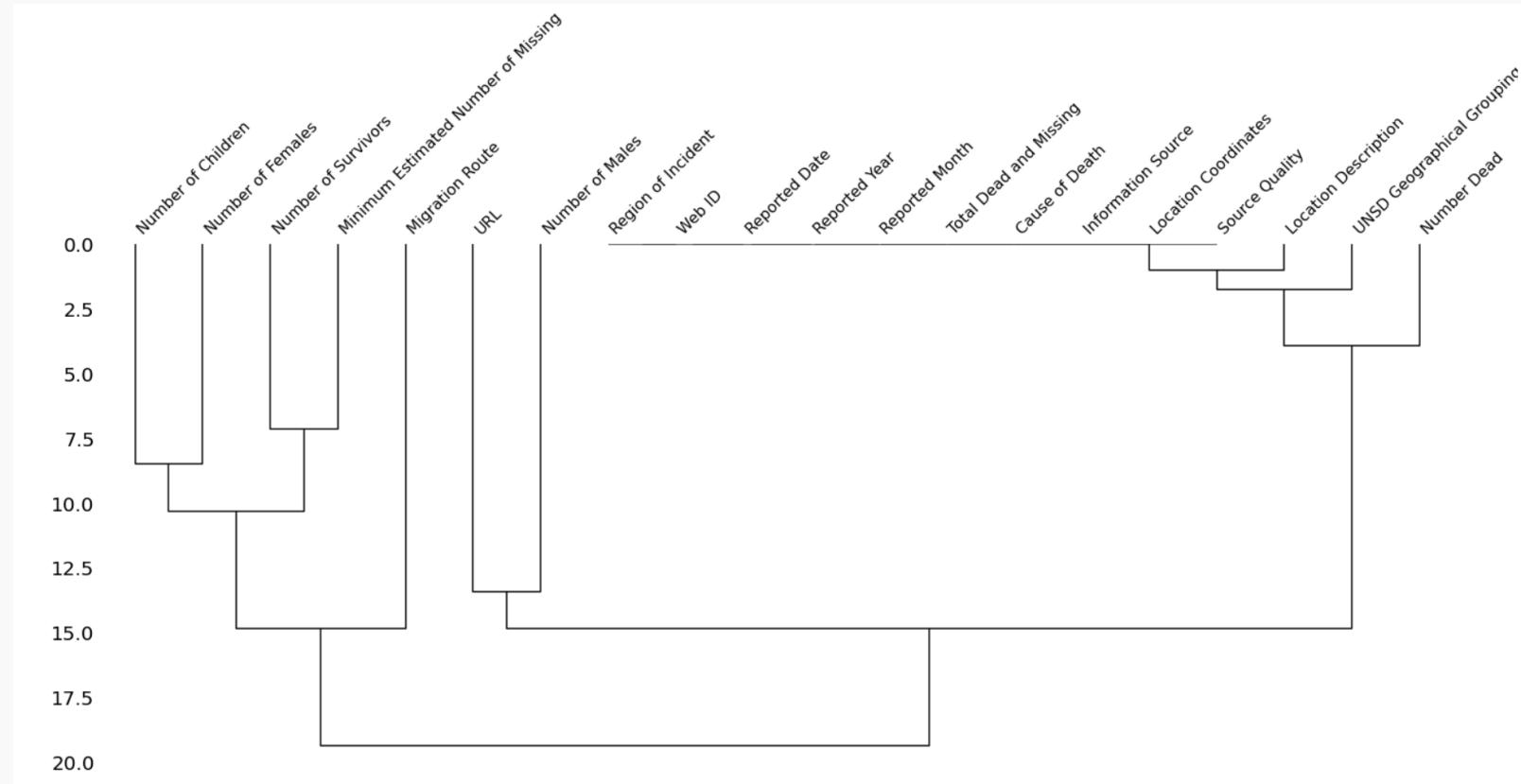
- missingno correlation heatmap: how strongly the presence or absence of one variable affects the presence of another
- sign of missing not completely at random



Example: Missing Migrants Project

```
msno.dendrogram(df.sample(500))
```

- dendrogram: fully correlate variable completion
- cluster leaves linked together at a distance of zero fully predict one another's presence
- some variables are 'glued' because they are present in every record



What Does SKLearn Do With Missingness?

- NaN is not a numerical value. Does this mean you will get an error if you try to fit an SKLearn model?
- Yes! You will need to resolve the missingness before you can fit an SKLearn model like LinearRegression on your data.
- So how do we resolve the missingness?

Dropping

Naively Handling Missingness (Dropping)

- What's the simplest ways to handle missing data?
- We can just throw away or ‘**drop**’ the observations that have any missing values.
- `df.dropna(axis=0)` will drop any rows with one or more missing values from your Pandas DataFrame.
- Dropping is ‘easy,’ but when is it advisable?

Dropping Columns & Rows

- It is probably a good idea to drop a **predictor column** if its values are missing for a large proportion of our observations.
- Similarly, if a given **observation row** has many missing values for its predictors, then we might want to drop it as well.
- But dropping rows can have **negative consequences**, and not just because we are throwing away data!

Dropping Columns & Rows

- If the observations with missing values differ systematically from observations without missing values, then we risk **biasing our dataset** by dropping rows with missingness!
- That is, if certain values are not missing at random but for some underlying *reason*, or, if the missingness is itself correlated with some other properties of an observation, then dropping such rows can get us into trouble.
- These are the **MAR** and **MNAR** cases we spoke about earlier.

Imputation

- Rather than simply dropping missing values, we could try to **fill them in** with well-chosen values.
- This ‘filling in’ is called ‘**imputation**’.
- The hope is that this will allow us to retain the relevant information in observations with missingness. Dropping would have just thrown it away!

Simple Imputation Methods

Simple Imputation Methods

- A common method is to impute using some sort of average value.
- What we mean by ‘average’ can depend on the type of the predictor variable we want to impute:
- Quantitative: Impute the **mean** or **median** value
- Categorical: Impute the **mode**, that is, the most common class

Missingness Indicator Variable

Here's a visual imputation example for a dataset with two predictors:

- X_1 which is quantitative
- X_2 which is categorical
- For X_1 we can replace its missing values by imputing the column's mean value.
- For X_2 we can impute using its mode, or most common value.



x_1X_1	x_2X_2	$x_1X_1^*$	$x_2X_2^*$
10	.	10	0
5	1	5	1
21	0	21	0
15	0	15	0
16	.	16	0
.	.	14.29	0
21	1	21	1
12	0	12	0
.	0	14.29	0

Missingness Indicator Variable

Why use a Missingness Indicator Variable?

How does this missingness indicator variable improve the model?

Because the group of individuals with a missing entry may be **systematically different** than those with that variable measured. Treating them equivalently could lead to bias in quantifying relationships and underperform in prediction.

For example: imagine a survey that asks if someone has ever recreationally used opioids, and some people chose not to respond. Does the fact that they did not respond provide extra information? Should we treat them equivalently as never-users?

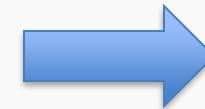
This approach essentially creates a third group for this predictor: the “did not respond” group.

Missingness Indicator Variable

One simple way to handle missingness in a variable, X_j :

- impute a value (like 0 or \bar{X}_j)
- create a new variable, $X_{j,miss}$, that indicates this observation had a missing value
- include both $X_{j,miss}$ and X_j as predictors in any model

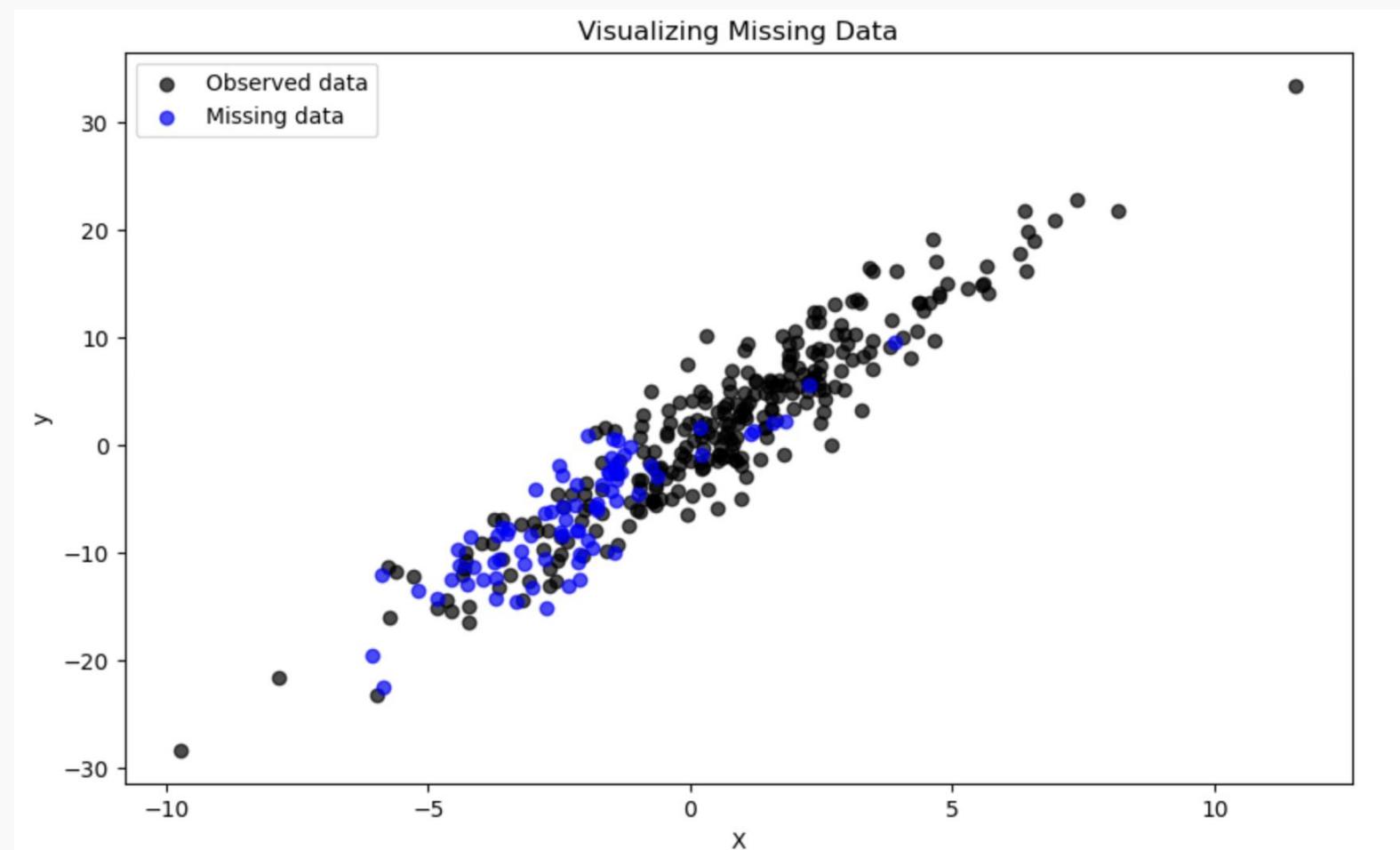
$x1X_1$	$x2X_2$
10	.
5	1
21	0
15	0
16	.
.	.
21	1
12	0
.	1



$x1X_1^*$	$x2X_2^*$	$x1_m$	$x2_m$
$X_{1,miss}$	$X_{2,miss}$		
10	0	0	1
5	1	0	0
21	0	0	0
15	0	0	0
16	0	0	0
0	0	0	1
21	1	1	1
12	0	0	0
0	1	0	0
		1	0

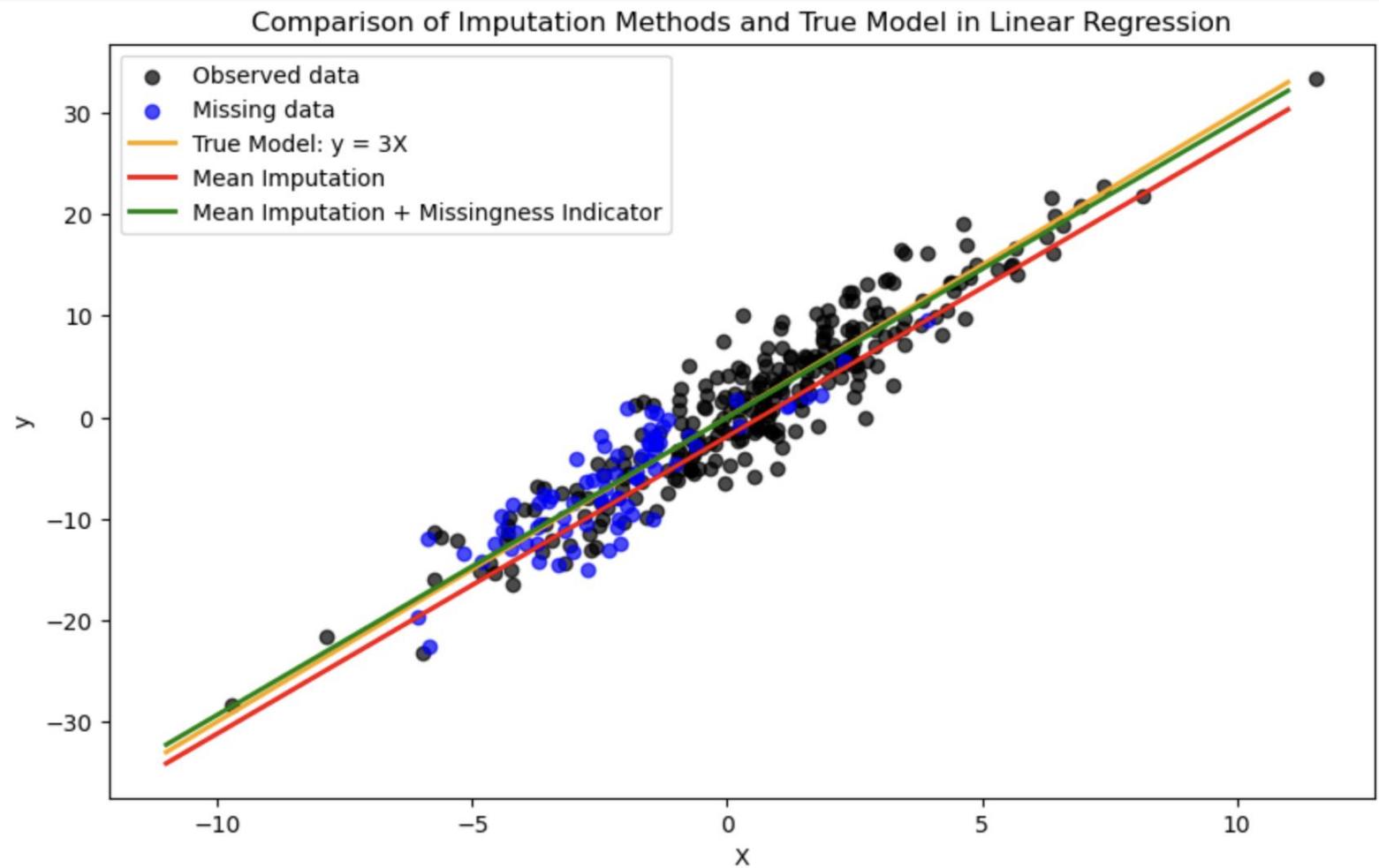
Example: Different Imputation Methods in Linear Regression

- Missing at Random
- units with smaller values of X are more likely to be missing



Example: Different Imputation Methods in Linear Regression

- mean imputation leads to biased inference
- mean imputation + missingness indicator performs better



Model-Based Imputation

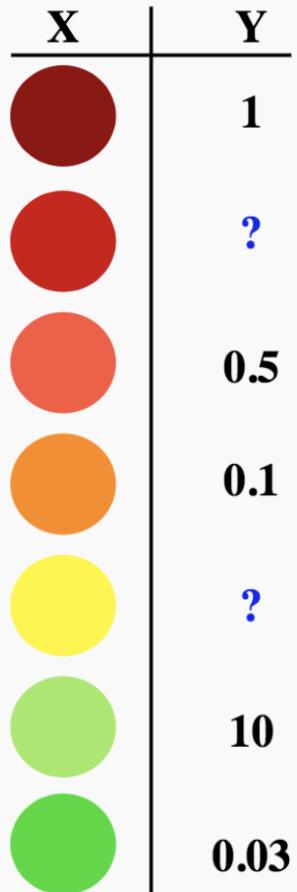
Imputation Methods

There are several different approaches to imputing missing values:

1. **Impute the mean or median** (quantitative) or most common class (categorical) for all missing values in a variable.
2. Create a new variable that is an **indicator of missingness**, and include it in any model to predict the response (also plug in zero or the mean in the actual variable).
3. **Hot deck imputation**: for each missing entry, randomly select an observed entry in the variable and plug it in.
4. **Model the imputation**: plug in predicted values (\hat{y}) from a model based on the other observed predictors.
5. **Model the imputation with uncertainty**: plug in predicted values plus randomness ($\hat{y} + \epsilon$) from a model based on the other observed predictors.

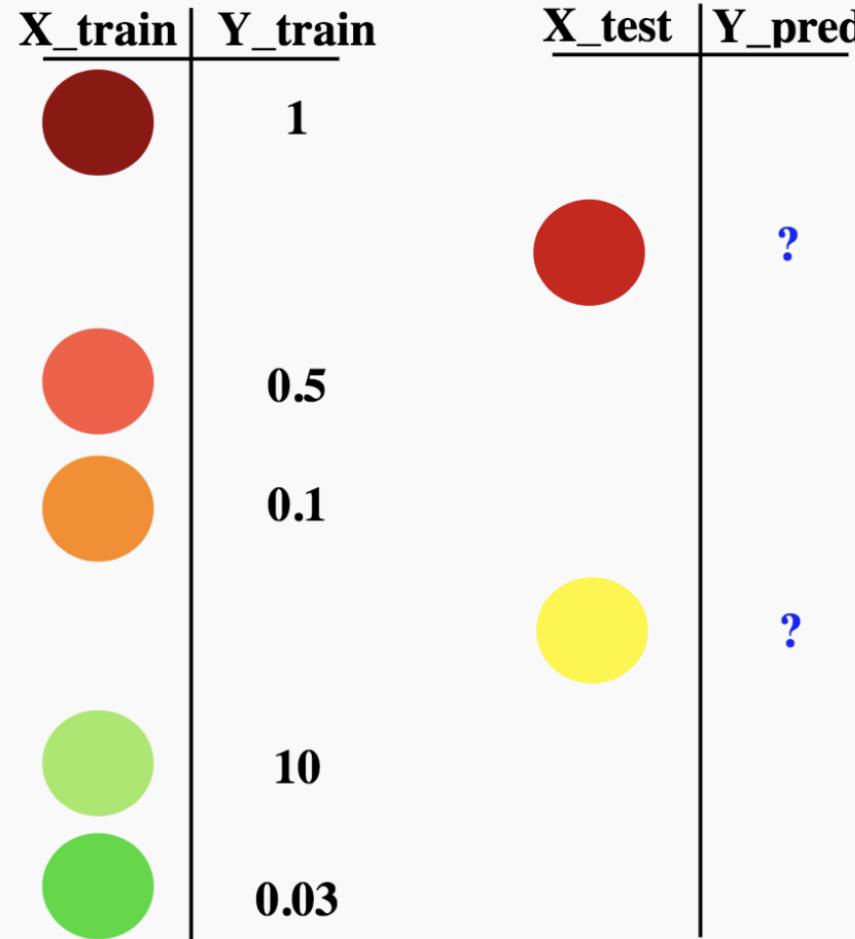
Schematic: imputation through modeling

How do we use models to fill in missing data?



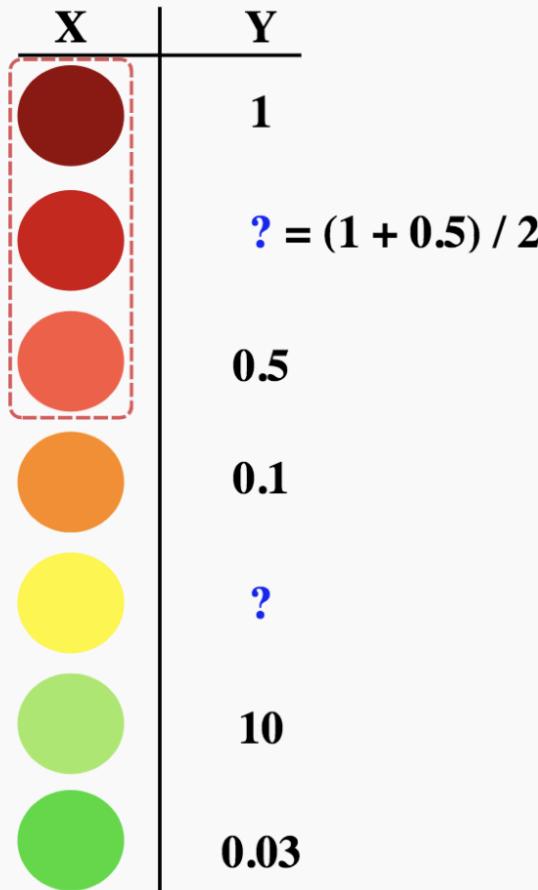
Schematic: imputation through modeling

How do we use models to fill in missing data?



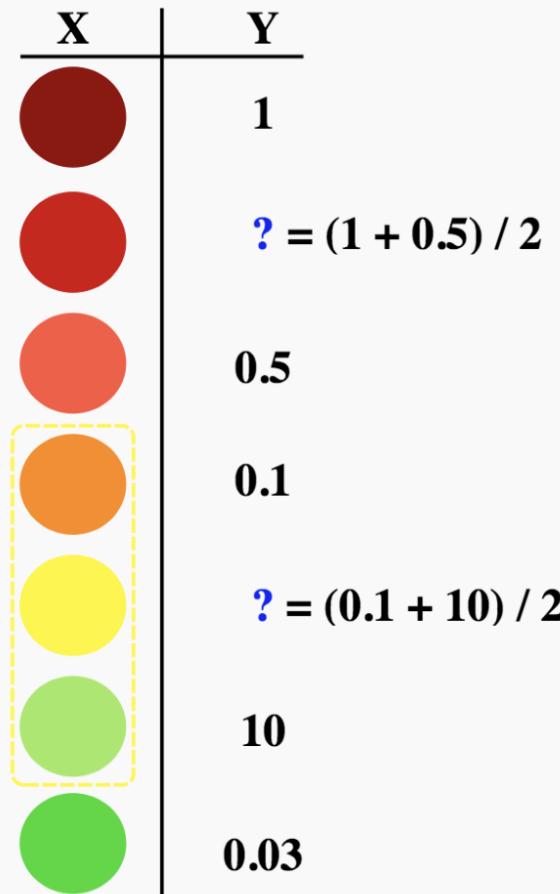
Schematic: imputation through modeling

How do we use models to fill in missing data? Using k -NN for $k = 2$?



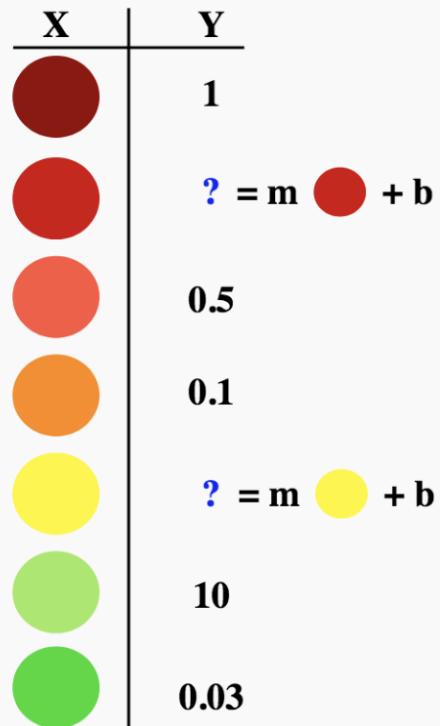
Schematic: imputation through modeling

How do we use models to fill in missing data? Using k -NN for $k = 2$?



Schematic: imputation through modeling

How do we use models to fill in missing data? Using linear regression?



Where m and b are computed from the observations (rows) that do not have missingness (we should call them $b = \beta_0$ and $m = \beta_1$).

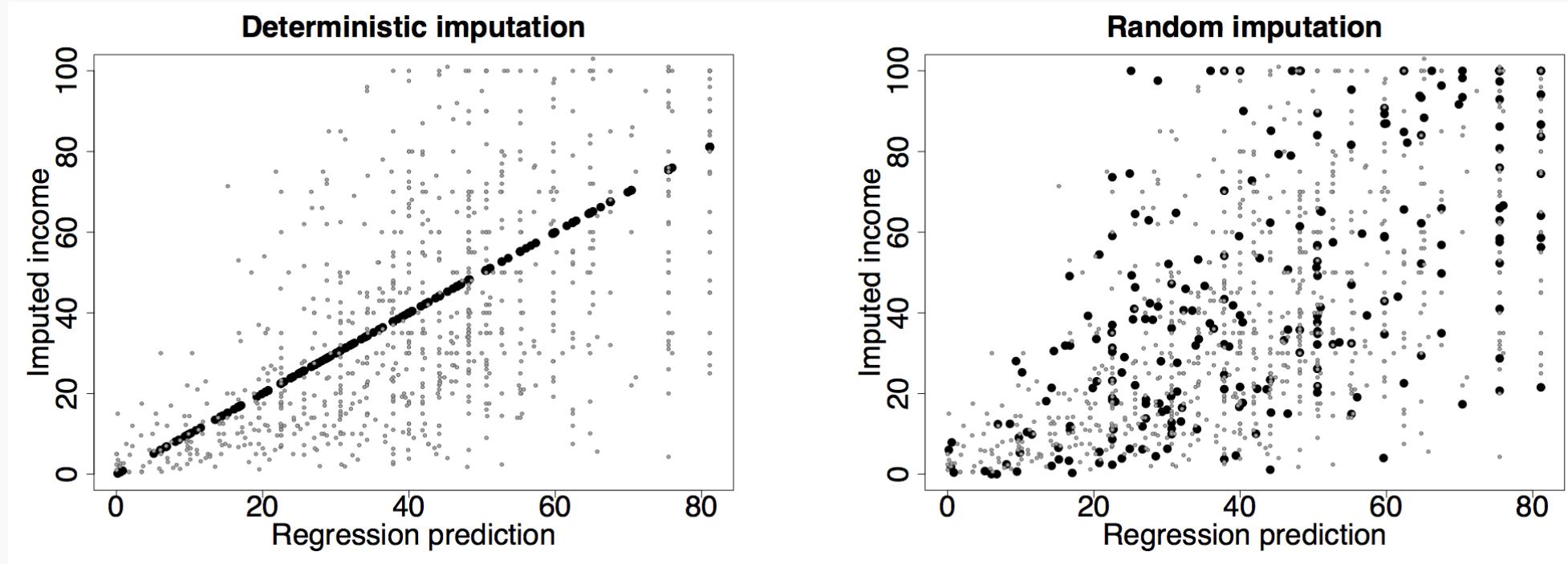
Imputation through modeling with uncertainty

The schematic in the last few slides ignores the fact of imputing with uncertainty. What happens if you ignore this fact and just use the ‘best’ model to impute values solely on \hat{y} ?

The distribution of the imputed values will be too narrow and not represent real data (see next slide for illustration). The goal is to impute values that include the uncertainty of the model.

How can this be done in practice in k -NN? In linear regression? In logistic regression?

Imputation: modeling with uncertainty (an illustration)



Imputation: modeling with uncertainty (an algorithm)

Recall the probabilistic model in linear regression:

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_P + \epsilon \quad \text{where } \epsilon \sim N(0, \sigma^2)$$

How can we take advantage of this model to impute with uncertainty?

It's a 3 step process:

1. Fit a model to predict the predictor variable with missingness from all the other predictors.
2. Predict the missing values from the model in the previous part.
3. Add in a measure of uncertainty to this prediction by randomly sampling from a $N(0, \sigma^2)$ distribution*, where σ^2 is the mean square error (MSE) from the model.

*Instead of sampling from a $N(0, \sigma^2)$ distribution, you can bootstrap sample from the observed residuals!

Imputation: modeling with uncertainty (k -NN regression)

How can we use k -NN regression to impute values that mimic the error in our observations?

Two ways:

- Use $k = 1$.
- Use any other k , but randomly select from the nearest neighbors in \mathcal{N}_0 . This can be done with equal probability or with some weighting (inverse to the distance measure used).

Imputation through modeling with uncertainty: classifiers

For classifiers, this imputation with uncertainty/randomness is a little easier process. How can it be implemented?

If a classification model (logistic, k -NN, etc...) is used to predict the variable with missingness on the observed predictors, then all you need to do is flip a ‘biased coin’ (or multi-sided die) with the probabilities of coming up for each class equal to the predicted probabilities from the model.

Warning: do not just classify blindly using the predict command in sklearn!

Imputation across multiple variables

If only one variable has missing entries, life is easy. But what if all the predictor variables have a little bit of missingness (with some observations having multiple entries missing)? How can we handle that?

It's an iterative process. Impute X_1 based on X_2, \dots, X_p . Then impute X_2 based on X_1 and X_3, \dots, X_p . And continue down the line.

Any issues? Yes, not all the missing values may be imputed with just one 'run' through the data set. You will have to repeat these 'runs' until you have a filled in data set.

Multiple imputation: beyond this class

What is an issue with treating your now ‘complete’ data set (a mixture of actually observed values and imputed values) as simply all observed values?

Any inferences or predictions carried out will be tuned and potentially overfit to the random entries imputed for the missing entries. How can we prevent this phenomenon?

By performing **multiple imputation**: rerun the imputation algorithm many times, refit the model on the response many times (one time each), and then ‘average’ the predictions or estimates of β coefficients to perform inferences (also incorporating the uncertainty involved).

Note: this is beyond what we would expect in this class. But it generally a good thing to be aware of.

sklearn's impute module

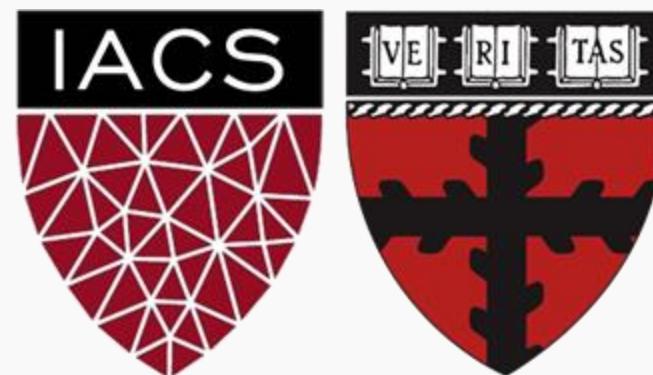
Want to do imputation in sklearn? Of course there are functions for that!

The 4 main functions in this module are:

1. **sklearn.impute.SimpleImputer**: perform mean, median, mode, or any specific value to impute
2. **sklearn.impute.IterativeImputer**: perform multivariate imputation that estimates each predictor from all the others (user-supplied model/estimator).
3. **sklearn.impute.KNNImputer**: perform imputation automatically from a k -NN model from all the other predictors.
4. **sklearn.impute.MissingIndicator**: to create binary indicator(s) for where the missing values occur.

Principal Component Analysis

CS1009A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai, Chris Gumb



Lecture Outline: High Dimensionality and PCA

- Motivating Example -- High Dimensionality
- Principal Components Analysis (PCA)
- PCA for Visualization
- PCA for Regression (PCR)
- Non-linear Dimensionality Reduction (t-SNE)



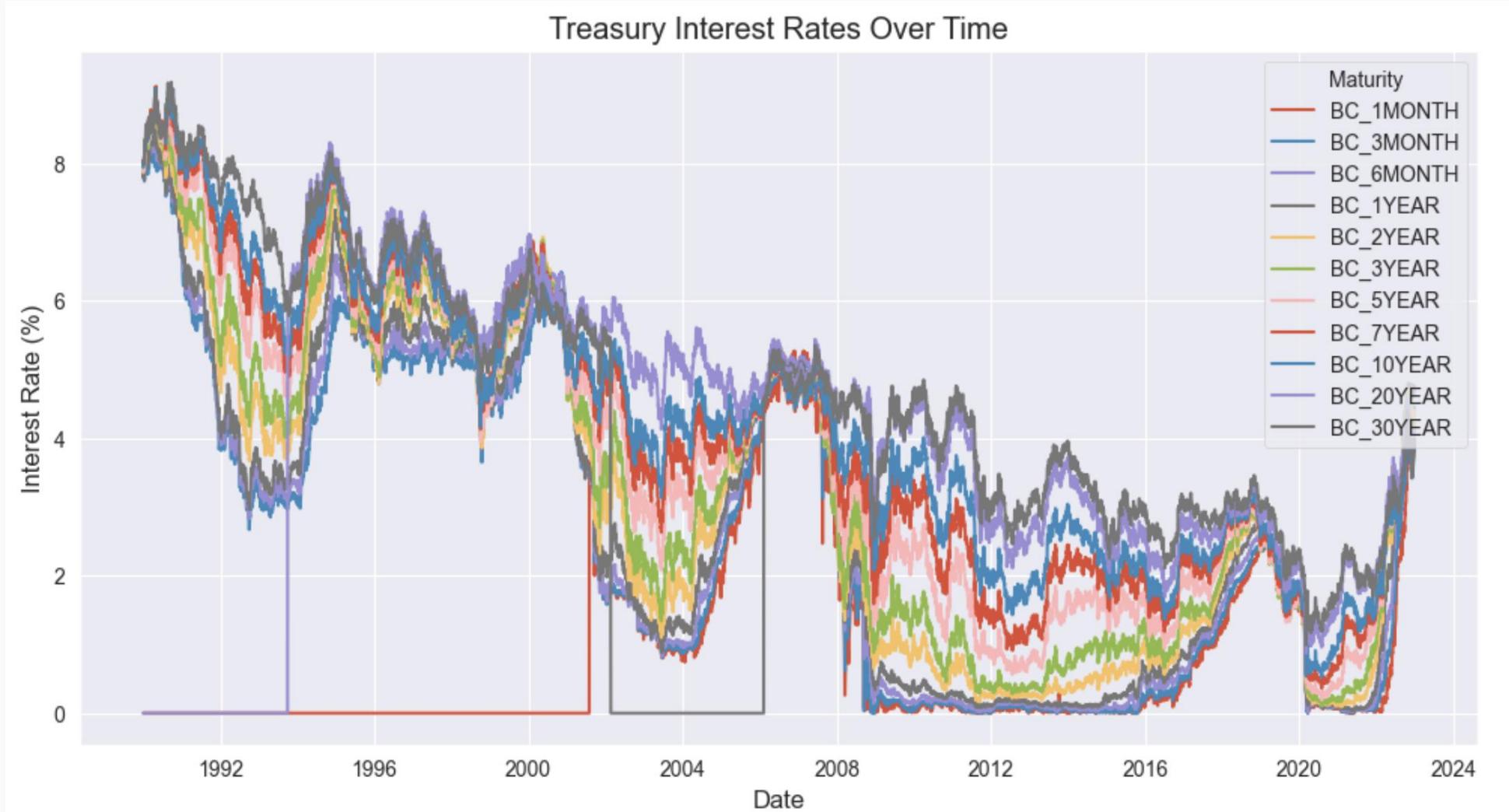
Interest Rates Data from the US Treasury

	BC_1MONTH	BC_3MONTH	BC_6MONTH	BC_1YEAR	BC_2YEAR	BC_3YEAR	BC_5YEAR	BC_7YEAR	BC_10YEAR	BC_20YEAR	BC_30YEAR
date											
1990-01-02	0.00	7.83	7.89	7.81	7.87	7.90	7.87	7.98	7.94	0.00	8.00
1990-01-03	0.00	7.89	7.94	7.85	7.94	7.96	7.92	8.04	7.99	0.00	8.04
1990-01-04	0.00	7.84	7.90	7.82	7.92	7.93	7.91	8.02	7.98	0.00	8.04
1990-01-05	0.00	7.79	7.85	7.79	7.90	7.94	7.92	8.03	7.99	0.00	8.06
1990-01-08	0.00	7.79	7.88	7.81	7.90	7.95	7.92	8.05	8.02	0.00	8.09
...
2022-12-23	3.80	4.34	4.67	4.66	4.31	4.09	3.86	3.83	3.75	3.99	3.82
2022-12-27	3.87	4.46	4.76	4.75	4.32	4.17	3.94	3.93	3.84	4.10	3.93
2022-12-28	3.86	4.46	4.75	4.71	4.31	4.18	3.97	3.97	3.88	4.13	3.98
2022-12-29	4.04	4.45	4.73	4.71	4.34	4.16	3.94	3.91	3.83	4.09	3.92
2022-12-30	4.12	4.42	4.76	4.73	4.41	4.22	3.99	3.96	3.88	4.14	3.97

8256 rows × 11 columns



Time Series Plot of the Interest Rates Data



“Co-movement” of the time series....

Visualizing high dimensional data

Visualization is often the quickest ways to gain insight into the relationships and patterns within a dataset.

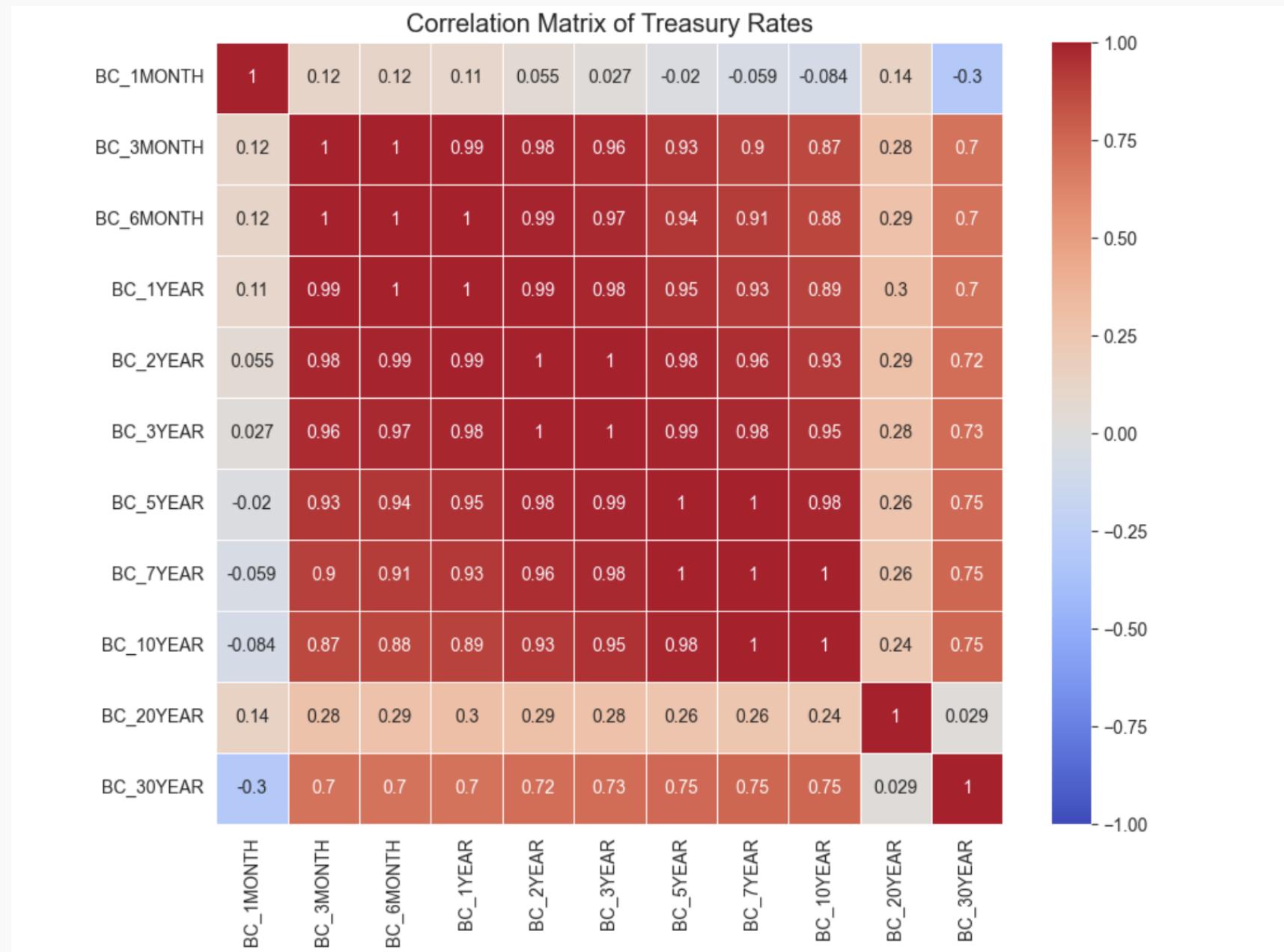
But how can we possibly visualize data beyond 2 or 3 dimensions?

One option is to compromise and look only at small ‘slices’ of the predictor space at time.

For example, we can create a series of pair-wise 2-D scatter plots between all the predictors.



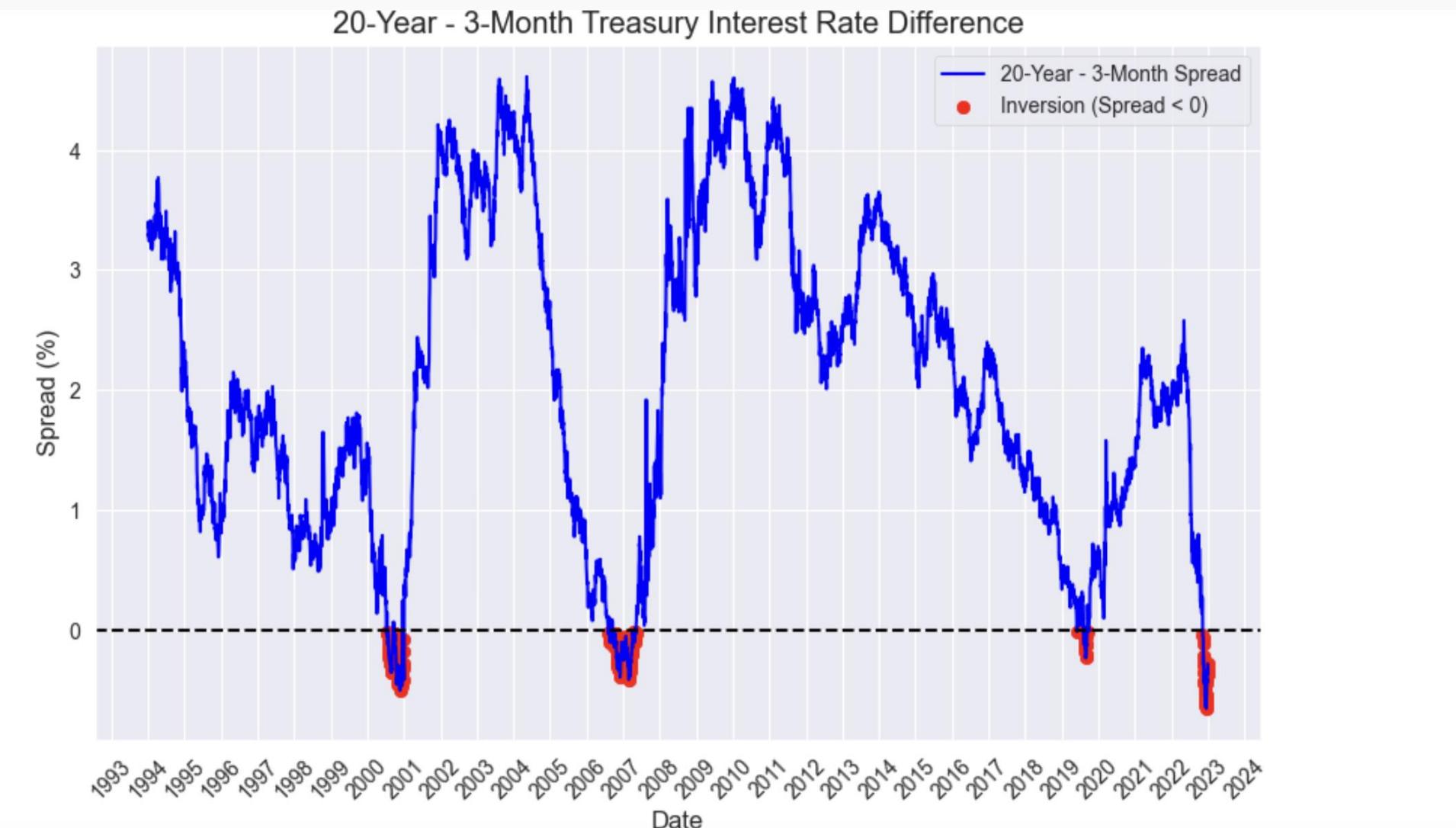
Correlation Matrix



Long Term vs. Short Term Rates.. Any patterns?



What happened the Market during Each of the Inversion Event?



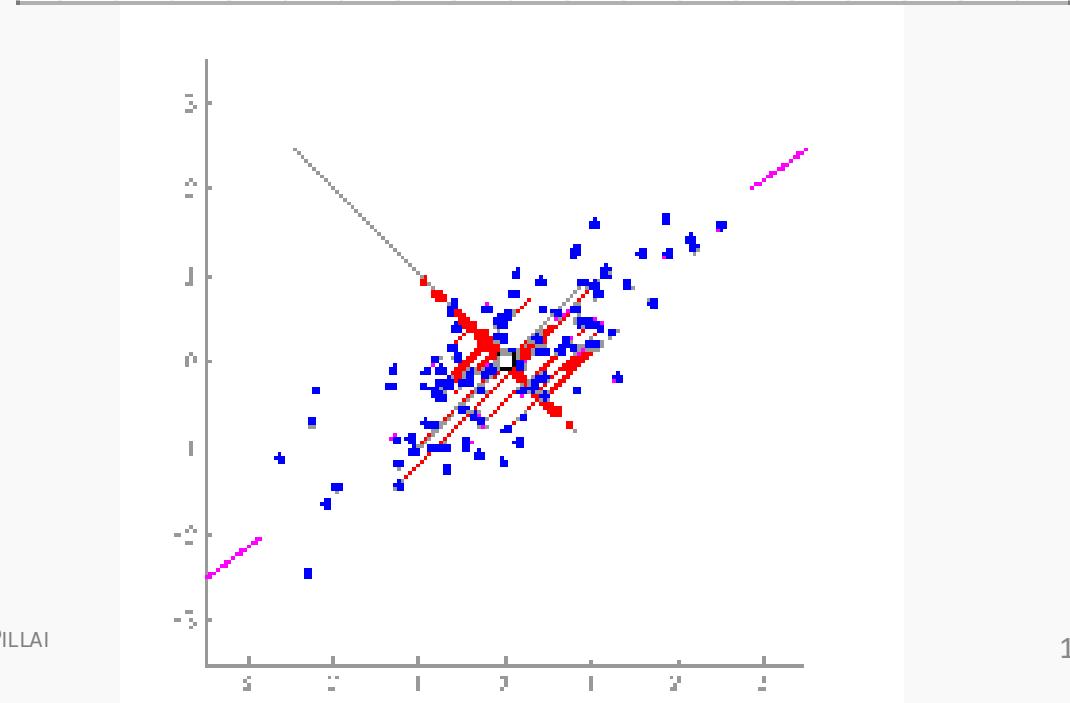
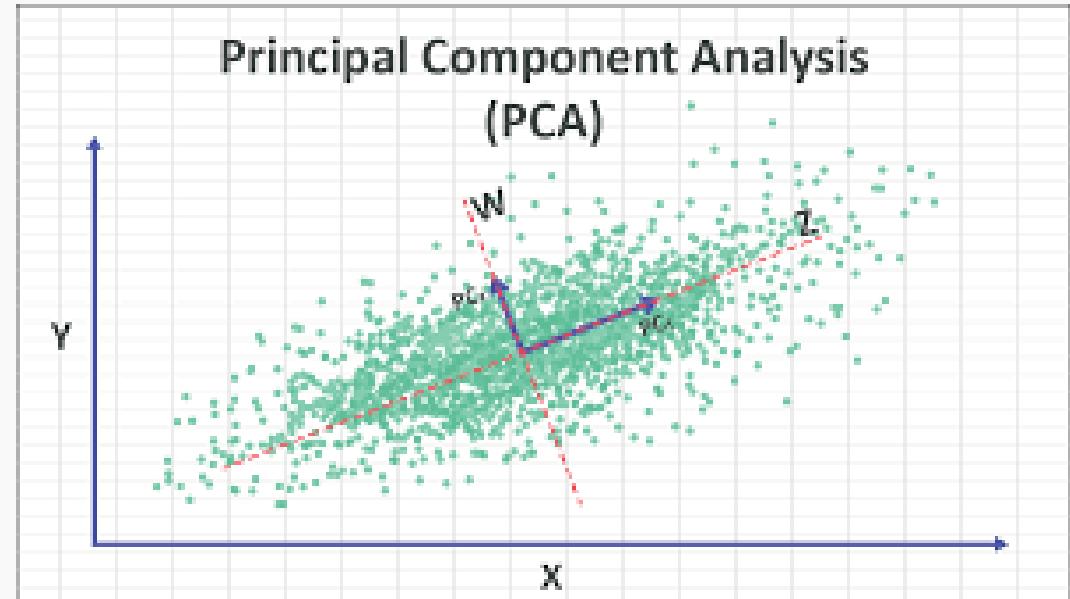
Why is this data/analysis different from Regression?

- In Regression, we have X (features) and Y (outcome). Thus we can relate Y to X. Labelled data.
- Called “Supervised” Learning.
- In our example, no “Y”; just X.
- X contains “interesting” patterns; how to extract these hidden patterns?
- Called “Unsupervised” Learning.



Principal Component Analysis (PCA)

- PCA provides a “geometric” way looking at variation in data
- Find “directions” in which the variation is **maximum**
- These directions reveal hidden patterns in the data.
- Most importantly, it is a “dimension-reduction” technique. We only need to look at a few directions (even for large p)



A geometric view of looking at data

One strategy is to remove those predictors which explain the least amount of variance in the data, or, equivalently, retain those predictors that **explain the most variance**.



In this example, almost all the variance is explained by X_1 .

Keeping only this predictor would retain most of the information while cutting the dimensionality in half.



An example with high correlation

A slightly different data set

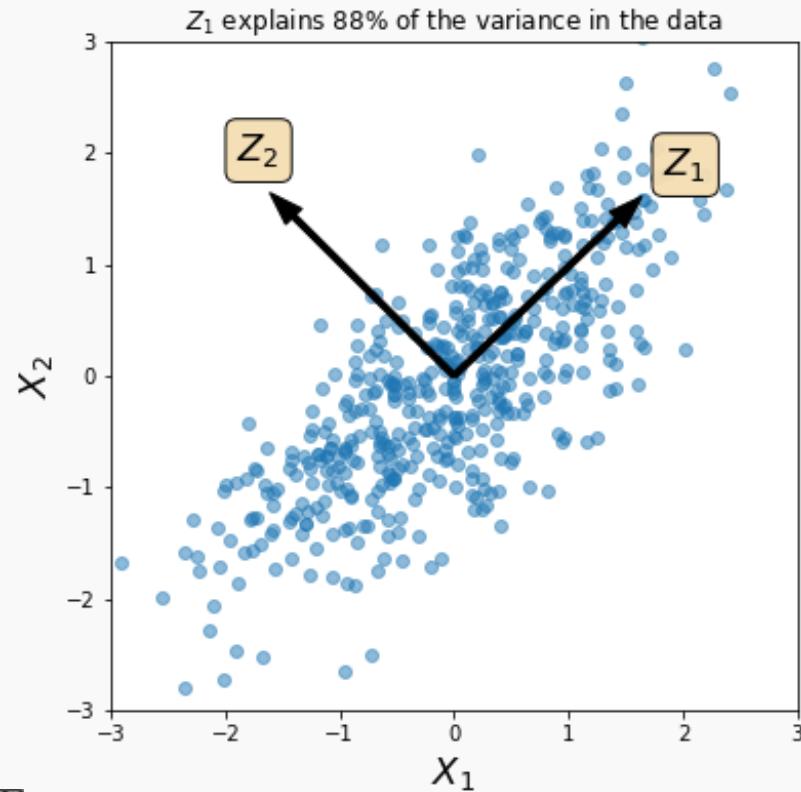


We can clearly see there are *some directions in the predictor space* that have more spread than others. They just don't lie along the axes (i.e., the basis vectors)



Directions of maximum variance

The vector Z_1 represents the direction of maximum variance in the predictor space. Z_2 is orthogonal and captures the remaining unexplained variance.

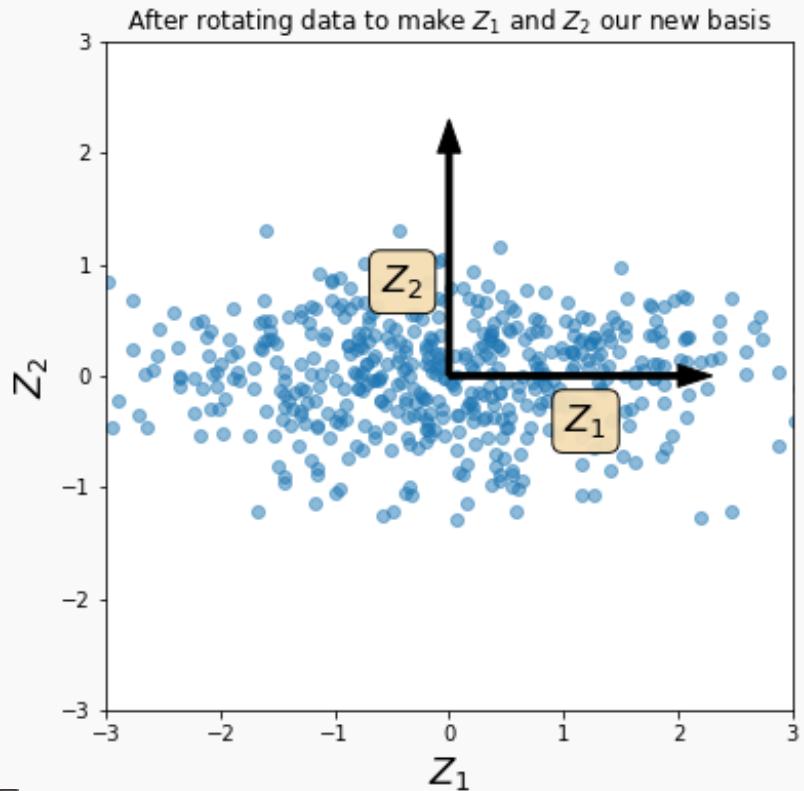


If our data were represented in terms of Z_1 and Z_2 then we could decide to keep only the Z_1 component.

This would again cut the dimensionality in half while retaining the most information.

Rotation for a change of basis

We can rotate our data so that the orthogonal Z_1 and Z_2 vectors now lie along the axes. This is equivalent to a change of basis from X_1 and X_2 to Z_1 and Z_2



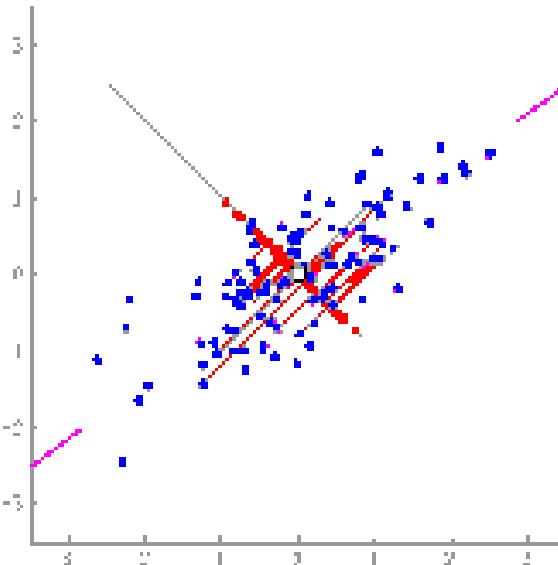
We can think of Z_1 and Z_2 as *a new set of predictors*. The decision of which predictor to keep to retain the most variance is once again clear.

Great! But how did we discover the optimal vectors Z_1 and Z_2 !?



The Intuition Behind PCA

Transforming our observed data means **projecting** our dataset onto the space defined by the top m PCA components. These components become our new predictors.



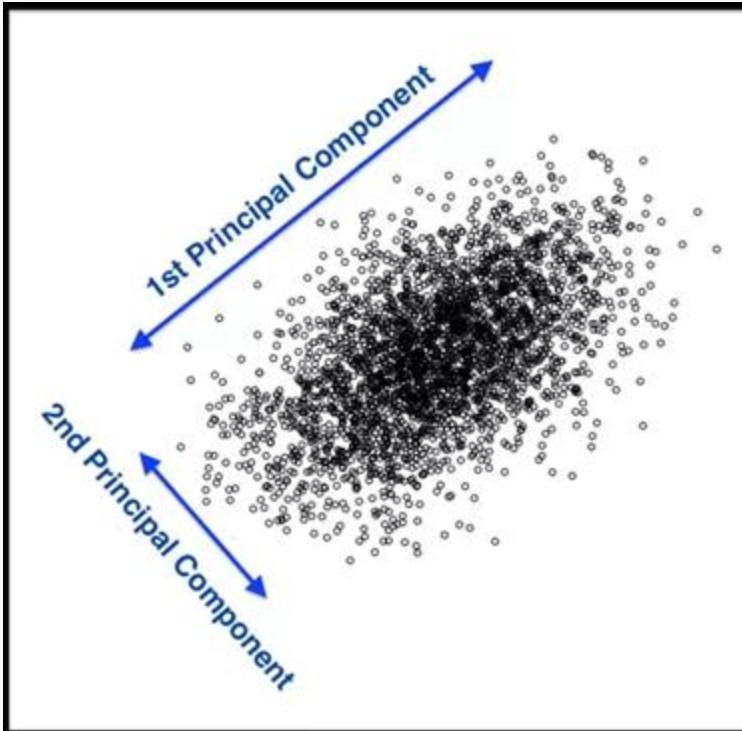
The animation shows projections onto several candidate vectors in **red**.

You can see the points retain the most spread when projected onto what turns out to be the first PCA component.



Principal Components Analysis (PCA)

Principal Components Analysis (PCA) is a method to identify a new set of predictors, as **linear combinations** of the original ones, that captures the **maximum amount of variance** in the observed data.



How are the PC components calculated?

Data(X)

	X1	X2	X3
1	2.0	4.0	1.0
2	1.0	3.0	5.0
3	4.0	2.0	3.0
4	3.0	5.0	2.0
5	5.0	1.0	4.0

Loadings (l)

Loading 1	Loading 2	Loading 3
0.5	-0.3	0.2

$$PC1 = +0.50 \times X1$$

(PC1)

PC1
1.0
0.5
2.0
1.5
2.5



A simpler dataset: 3 Penguin Species

```
In [12]: penguins = sns.load_dataset('penguins')
```

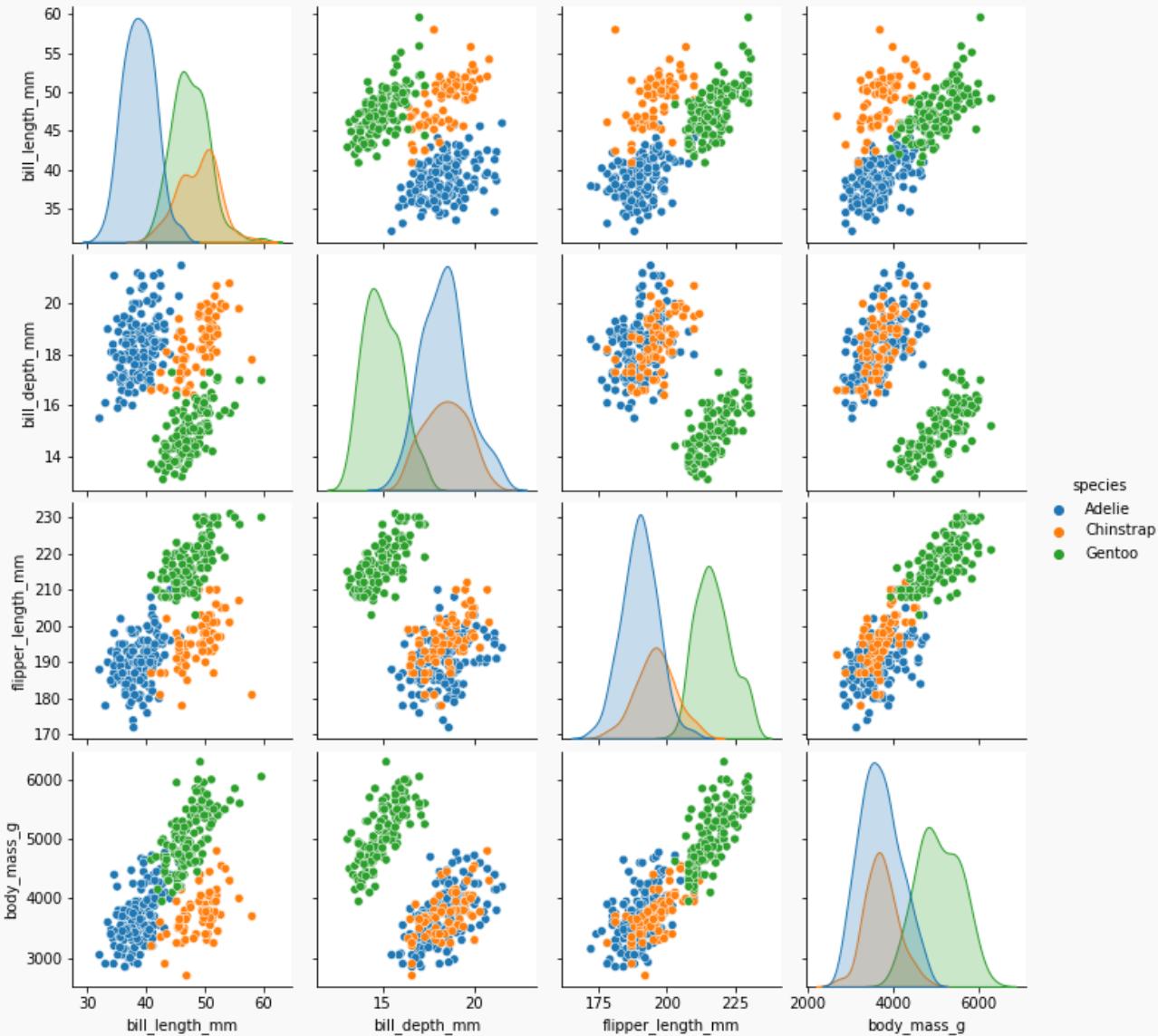
```
In [13]: penguins.head()
```

Out[13]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE



3 Penguin Species: Pair-plot of features



We have 4 measurements (4 X s) from 3 species of penguins: body mass, flipper length, bill length, and bill depth.

We can see some separation of the species in a few of the scatter plots. But what about relationships involving more than just 2 predictors?

This approach also becomes unwieldy with more than just a few predictors.



PCA fitting

```
X = pd.get_dummies(penguins, columns=['island', 'sex'], drop_first=True)
X = X.dropna()
```

```
X_std = StandardScaler().fit_transform(X.drop('species', axis=1))
```

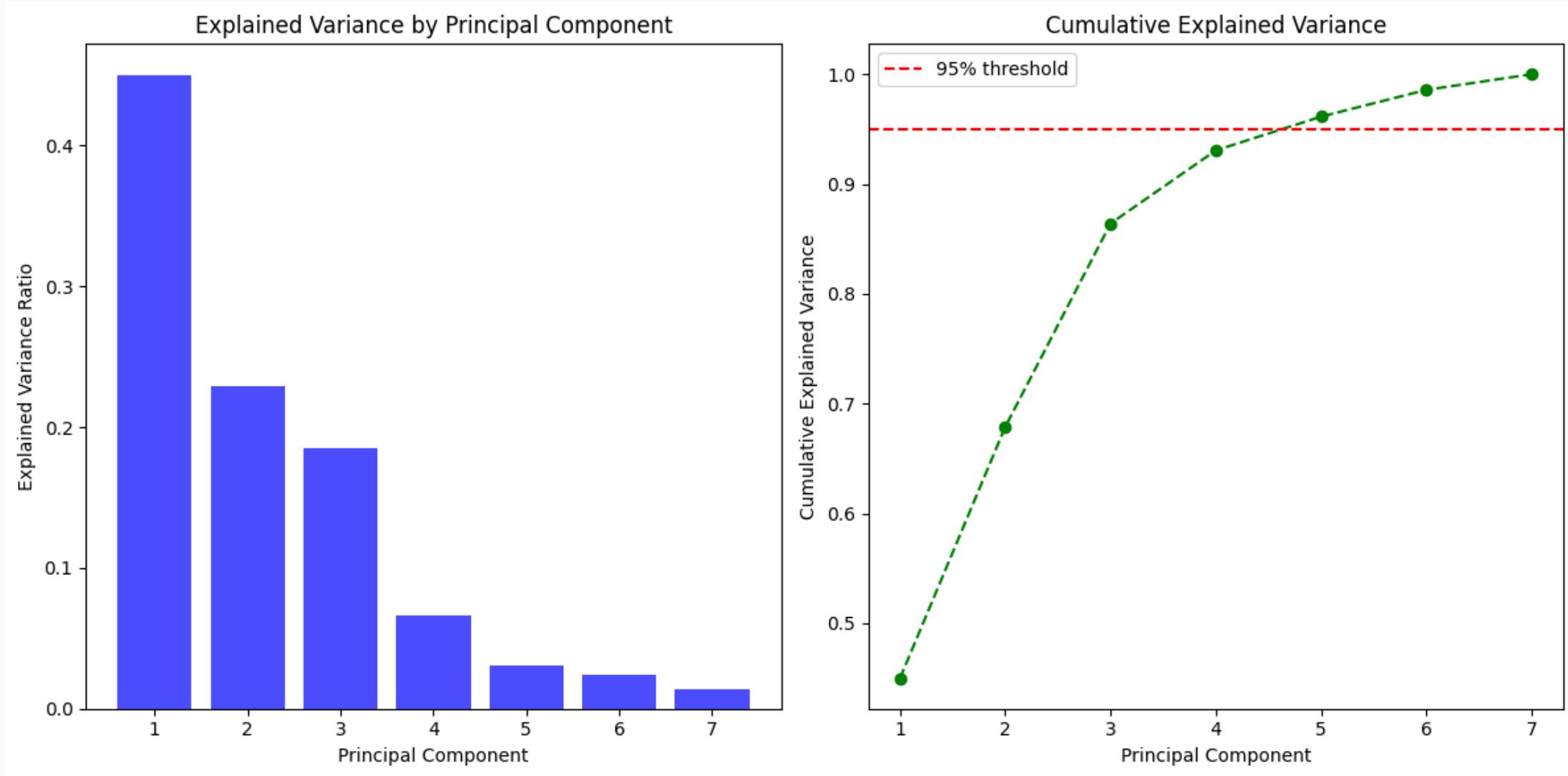
```
# Perform PCA (keeping all components for now)
pca = PCA(n_components=None)
X_pca = pca.fit_transform(X_std)

# Explained variance ratio for each principal component
explained_variance = pca.explained_variance_ratio_

# Cumulative explained variance
cumulative_explained_variance = np.cumsum(explained_variance)
```



Explained Variance

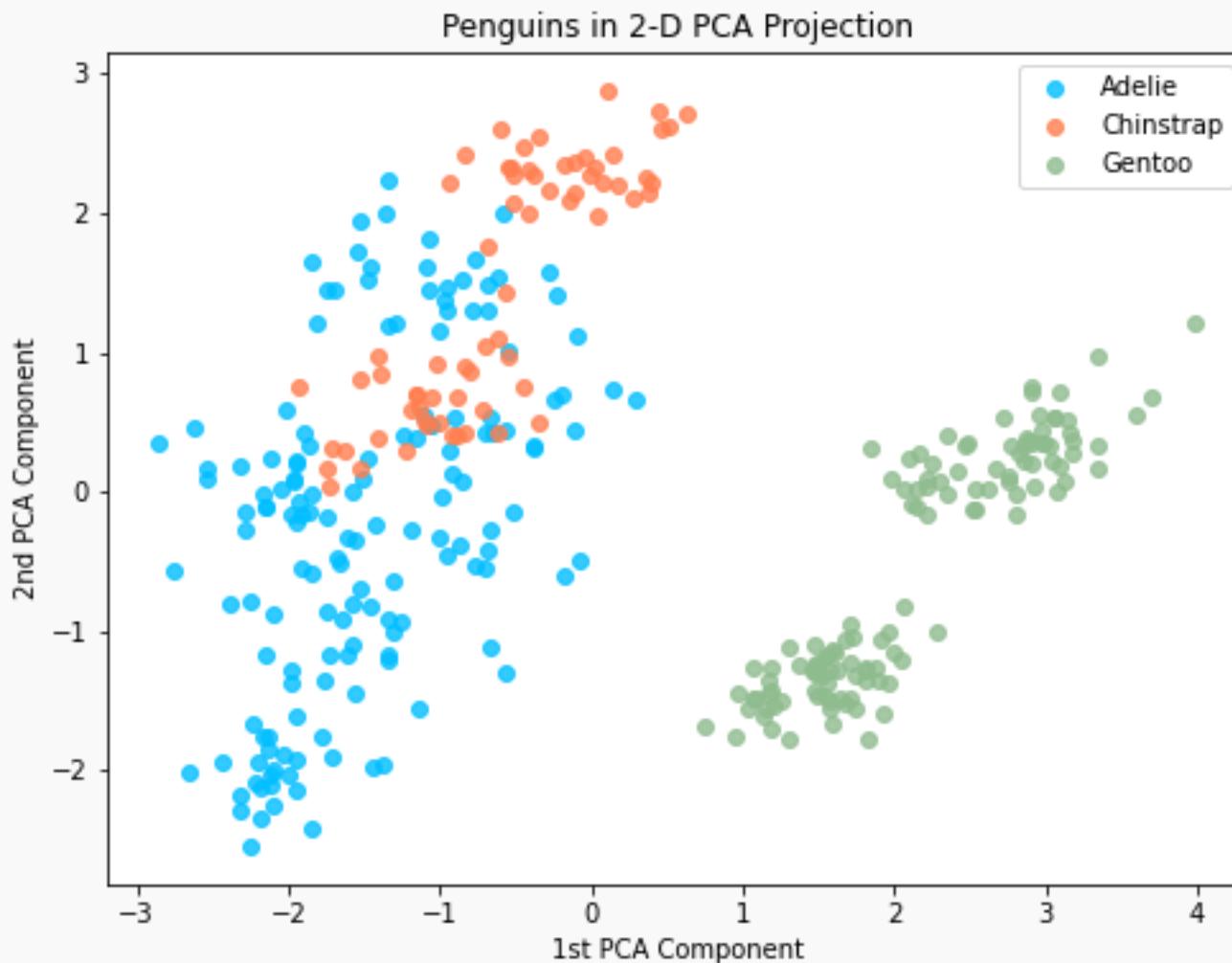


Loadings: Unique only up to signs!!

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
bill_length_mm	0.410902	0.349188	-0.082807	0.678847	-0.441822	-0.187035	0.106528
bill_depth_mm	-0.357584	0.453812	0.364238	0.014070	-0.307439	0.661835	-0.027885
flipper_length_mm	0.535030	-0.031667	0.027397	0.104100	0.294947	0.402166	-0.672622
body_mass_g	0.524285	0.026440	0.175142	-0.162403	0.259477	0.322410	0.704344
island_Dream	-0.253579	0.546865	-0.393841	0.234672	0.648157	0.004212	0.079561
island_Torgersen	-0.219095	-0.327456	0.618131	0.573507	0.351530	-0.080348	0.061185
sex_Male	0.172792	0.513937	0.540233	-0.343184	0.110781	-0.504810	-0.171112



3 Penguin Species: Visualized with PCA

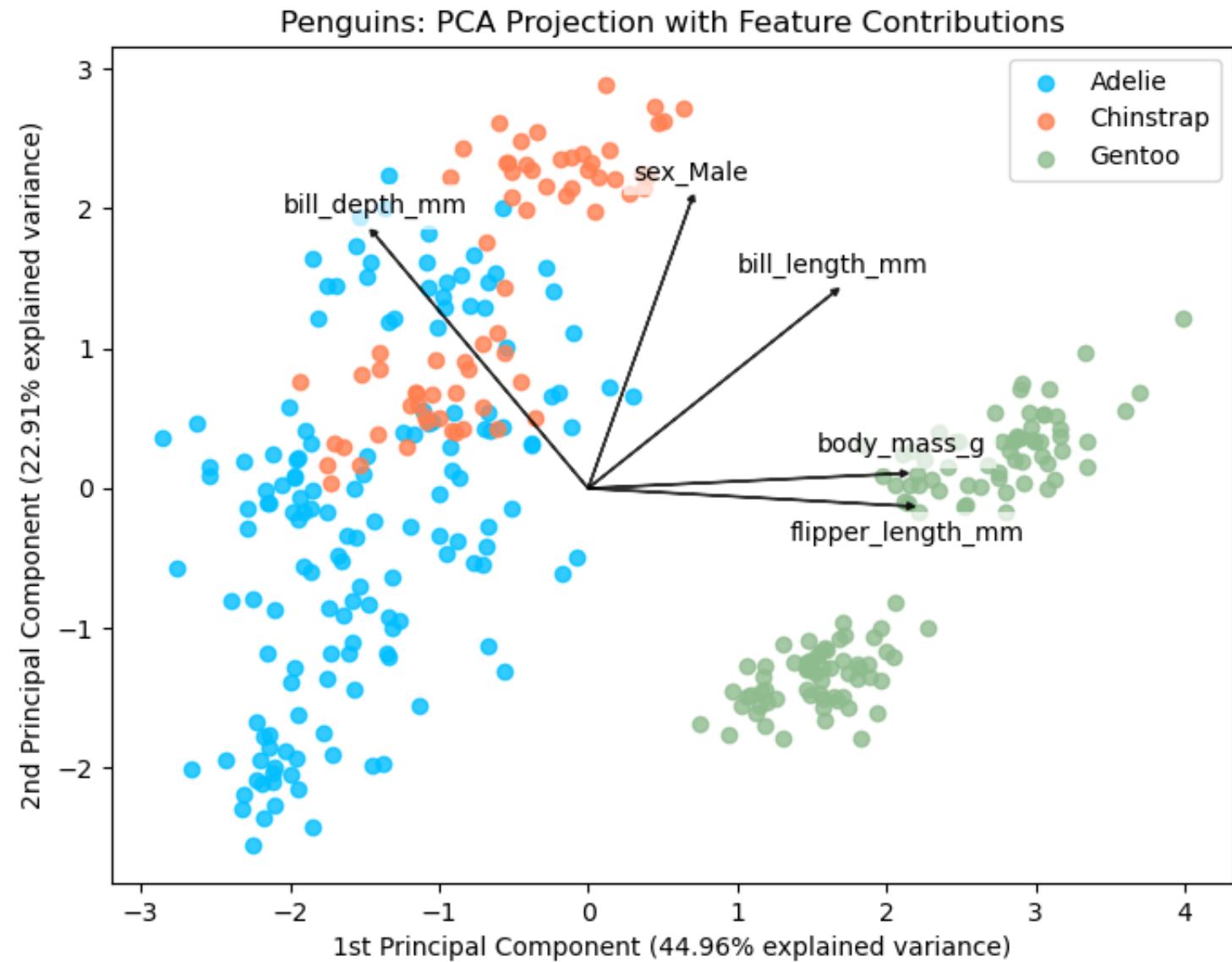


Here is the data projected onto the first 2 principal components.

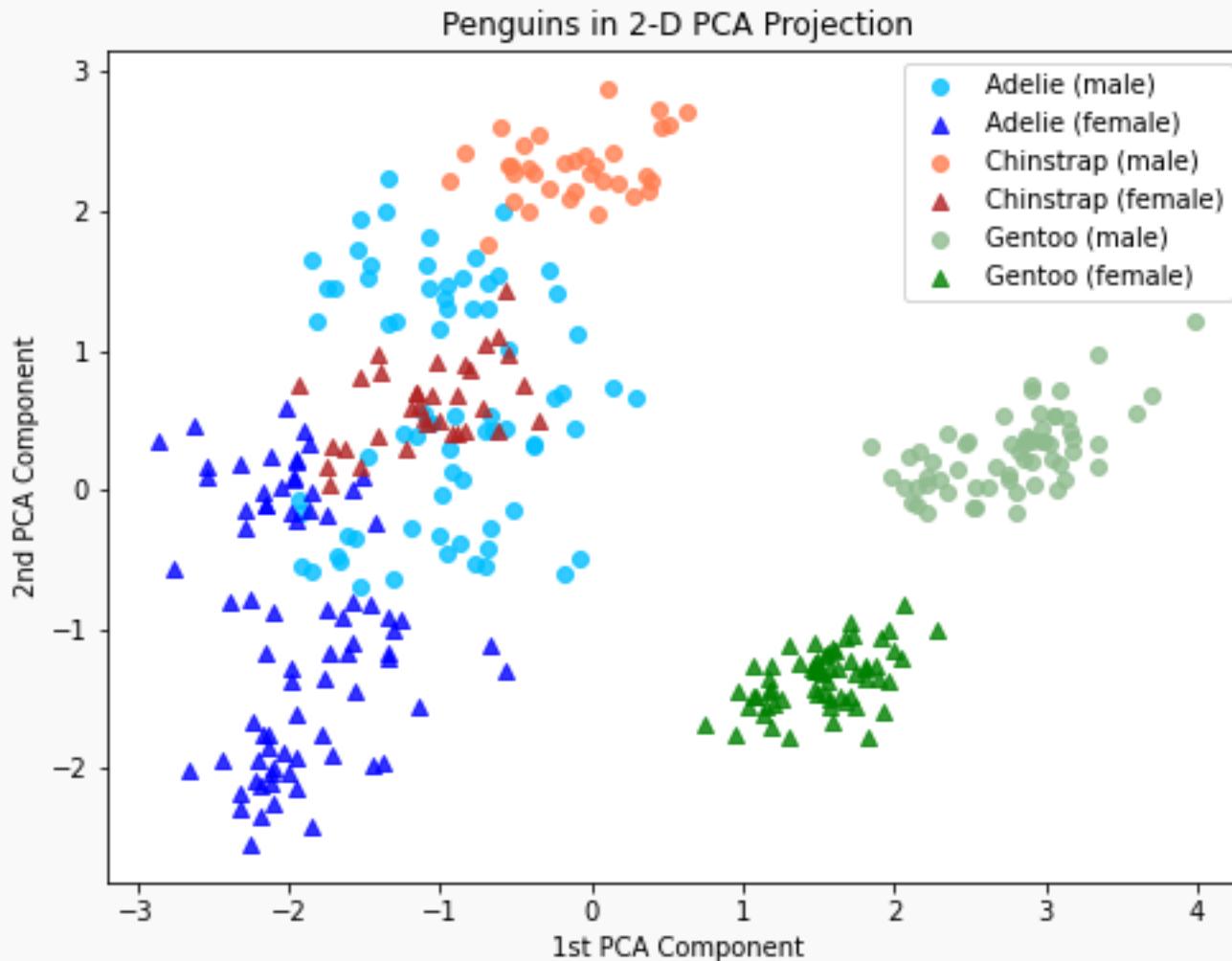
Important point: we have massively reduced dimensions!!!



3 Penguin Species: Interpretability



3 Penguin Species: Visualized with PCA

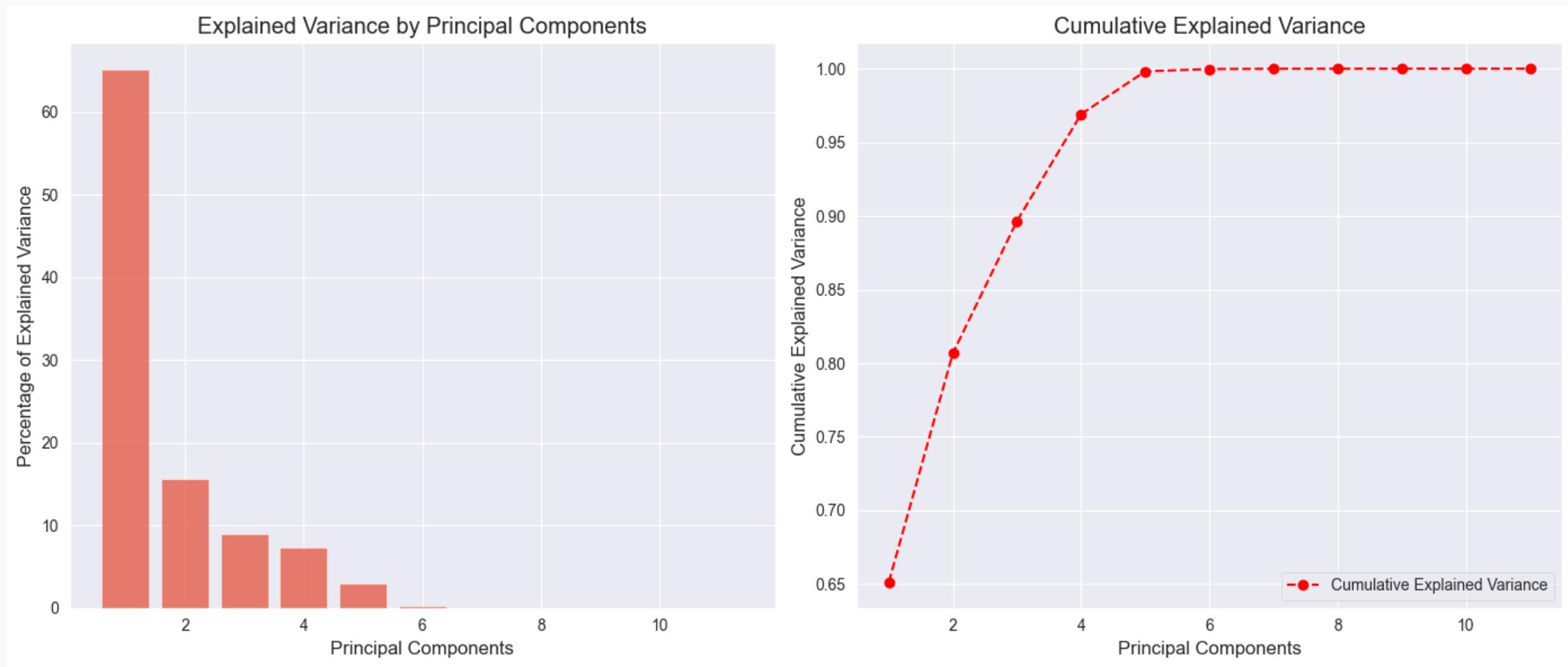


It turns out that these within species clusters correspond to male and female penguins.

Now let's try another example with a much higher dimensional dataset...



PCA for the interest rates data. First three Principal Components Dominate



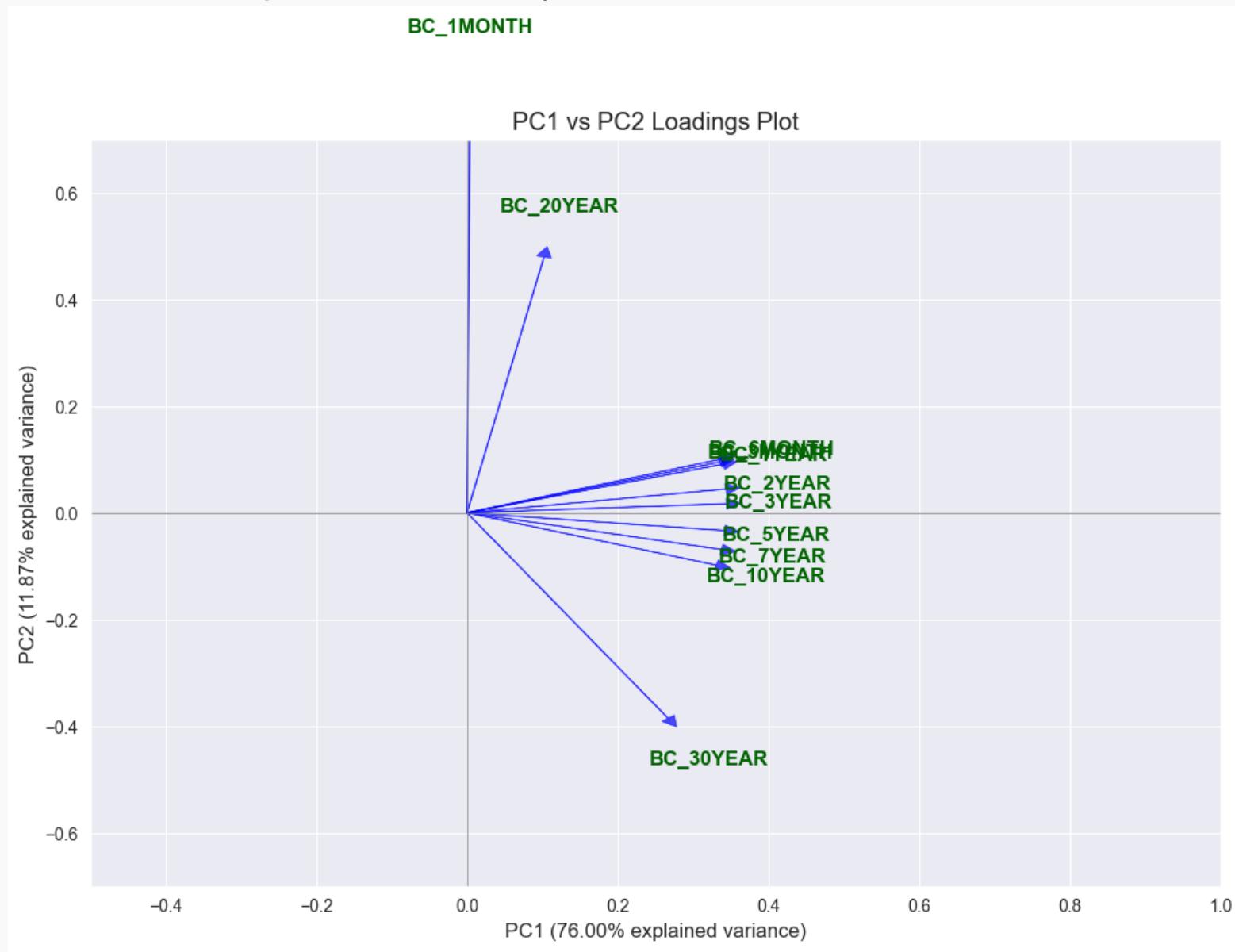
Understanding the Loadings

df_loadings

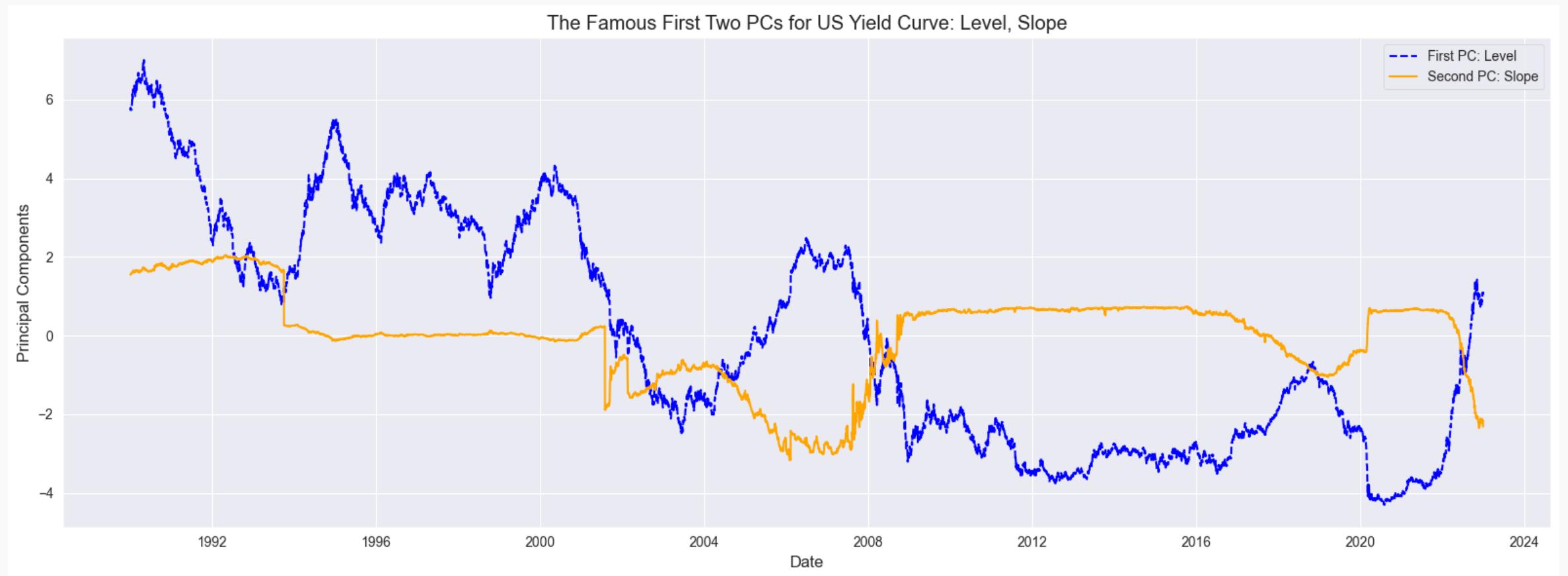
	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11
BC_1MONTH	0.003884	0.759330	-0.497909	0.261360	0.326251	-0.022919	-0.007349	-0.008473	-0.009024	0.002632	0.001284
BC_3MONTH	0.335358	0.095067	-0.098885	0.055223	-0.401221	0.537819	-0.483162	-0.370692	0.117841	0.174460	-0.016144
BC_6MONTH	0.336874	0.100567	-0.088690	0.043383	-0.377082	0.226047	0.196436	0.491260	-0.235412	-0.580388	0.034264
BC_1YEAR	0.339706	0.091051	-0.064455	0.006561	-0.308415	-0.129561	0.511945	0.203964	0.307951	0.597991	0.063948
BC_2YEAR	0.343529	0.044519	-0.037764	-0.079538	-0.149961	-0.432133	0.123669	-0.444379	-0.423842	-0.056834	-0.517096
BC_3YEAR	0.344336	0.017043	-0.021784	-0.127764	-0.014820	-0.456816	-0.175505	-0.214892	0.095228	-0.208806	0.725117
BC_5YEAR	0.341843	-0.033323	0.000477	-0.186855	0.228424	-0.204045	-0.347311	0.303758	0.568896	-0.166145	-0.434932
BC_7YEAR	0.337698	-0.068740	0.025266	-0.214769	0.357009	0.088821	-0.276198	0.365566	-0.554810	0.413768	0.100922
BC_10YEAR	0.330845	-0.098031	0.036843	-0.230143	0.502487	0.441122	0.471754	-0.331655	0.124917	-0.171576	0.044006
BC_20YEAR	0.102257	0.479515	0.852282	0.176357	0.042625	0.010165	-0.012244	-0.003277	0.001569	-0.006008	-0.001747
BC_30YEAR	0.266917	-0.385129	-0.002598	0.859627	0.196519	-0.052028	-0.010878	-0.003775	-0.002192	-0.001016	0.001171



Interest rate -- PC components overlay



The evolution of principal components overtime



The Math behind PCA

Let \mathbf{X} be the $n \times p$ matrix with columns X_1, \dots, X_p (our original predictors), each standardized to have mean zero and variance one, and without the intercept)

Let let \mathbf{W} be the $p \times p$ matrix whose columns are the **eigenvectors** of the **correlation matrix**, $\mathbf{X}^T \mathbf{X}$

Let \mathbf{Z} be the $n \times p$ matrix with columns Z_1, \dots, Z_p (the principal components)

$$\mathbf{Z}_{n \times p} = \mathbf{X}_{n \times p} \mathbf{W}_{p \times p}$$



Implementation of PCA using linear algebra

To implement PCA yourself you can perform the following steps:

- I. Standardize your predictors (so they each have mean = 0, var = 1).
- II. Calculate the [eigenvectors](#) of $\mathbf{X}^T \mathbf{X}$ and arrange them as columns in order of descending [eigenvalues](#) in a new matrix, \mathbf{W} .
- III. Use matrix multiplication to project \mathbf{X} onto the eigenvectors to create the new feature matrix, $\mathbf{Z} = \mathbf{X}\mathbf{W}$

Note: this is not efficient from a computational perspective. This can be sped up using [Cholesky decomposition](#).

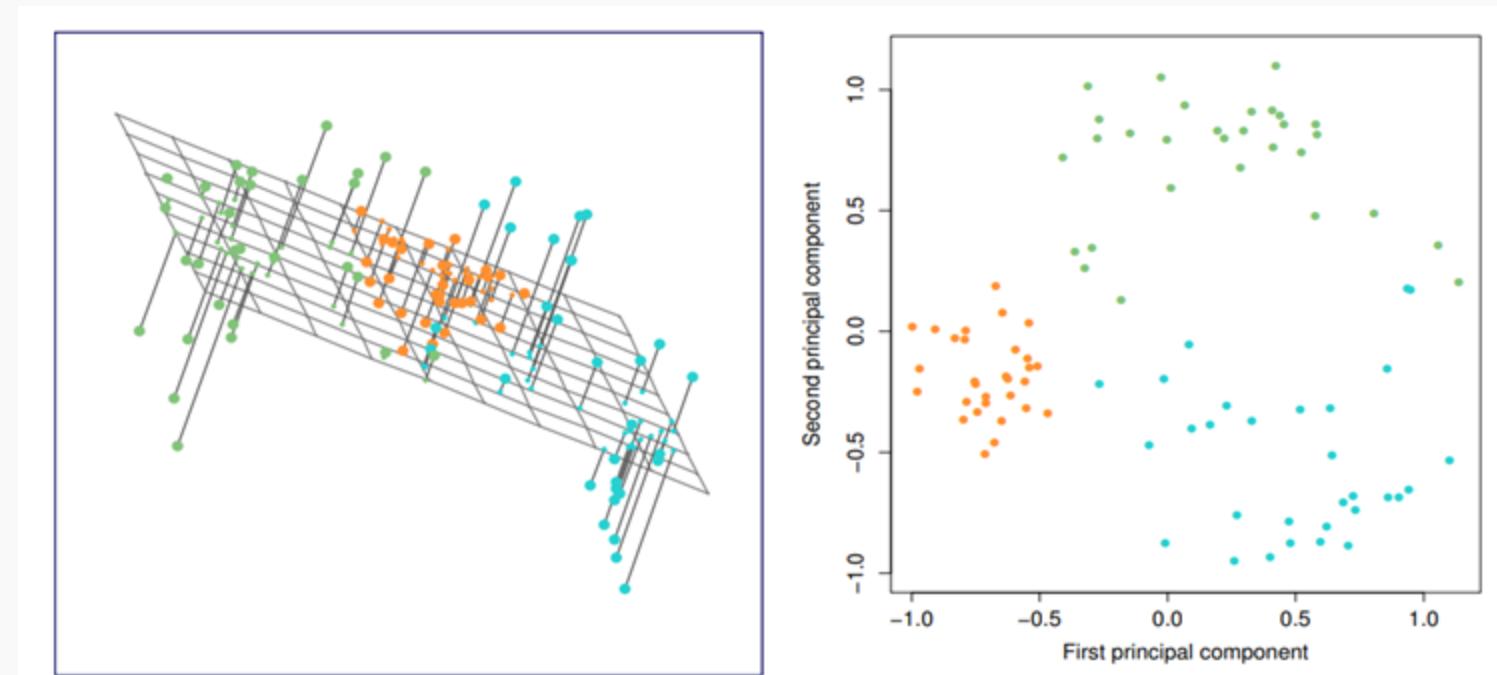
PCA is easy to perform in Python using sklearn's `decomposition.PCA` class.



An Alternative Interpretation of PCA

We've seen an interpretation of PCA as finding the directions in the predictor space along which the data varies the most.

An alternative interpretation is that PCA finds a low-dimensional linear surface which is *closest* to the data points.



PCA example in sklearn

```
x = homes[['sqft','beds','baths','lotsize','dist']]  
  
pca = PCA().fit(x)  
  
pcaX = pca.transform(x)  
W = pca.components_.T  
  
  
print("First PCA Component vector (w1):",W[0,:].round(7))  
print("Second PCA Component vector (w2):",W[1,:].round(7))  
  
print("Variance explained by each component:",pca.explained_variance_ratio_.round(7))  
  
First PCA Component vector (w1): [ 3.237999e-01  9.461243e-01 -7.237000e-04  9.225000e-04  1.009800e-03]  
Second PCA Component vector (w2): [ 4.902000e-04  1.144900e-03  6.274349e-01 -7.753376e-01 -7.194070e-02]  
Variance explained by each component: [9.097255e-01 9.027410e-02 2.000000e-07 1.000000e-07 1.000000e-07]
```



Images as high dimensional data

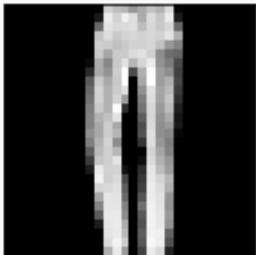
The **Fashion MNIST** data set consists of thousands of 28x28 grayscale images of articles of clothing from 10 different categories. Each pixel is essentially a feature. $28 \times 28 = 784$ features!

And obviously plotting pair-wise scatter plots of pixel values wouldn't be insightful.

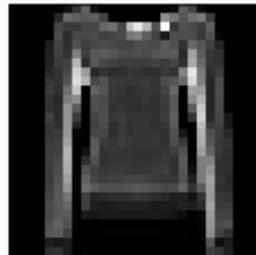
T-shirt/top



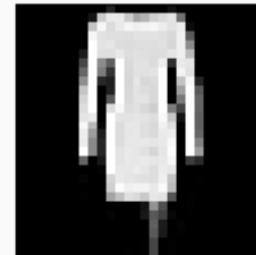
Trouser



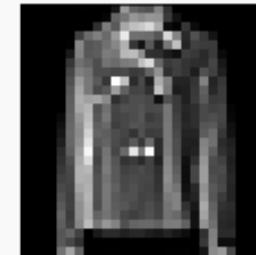
Pullover



Dress



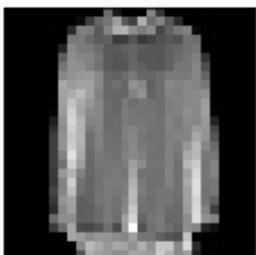
Coat



Sandal



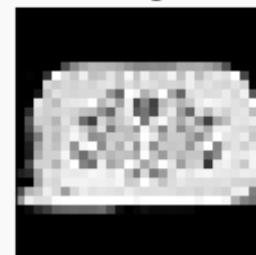
Shirt



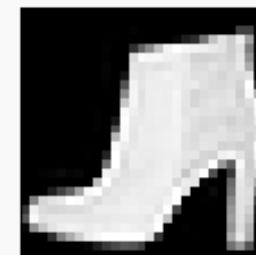
Sneaker



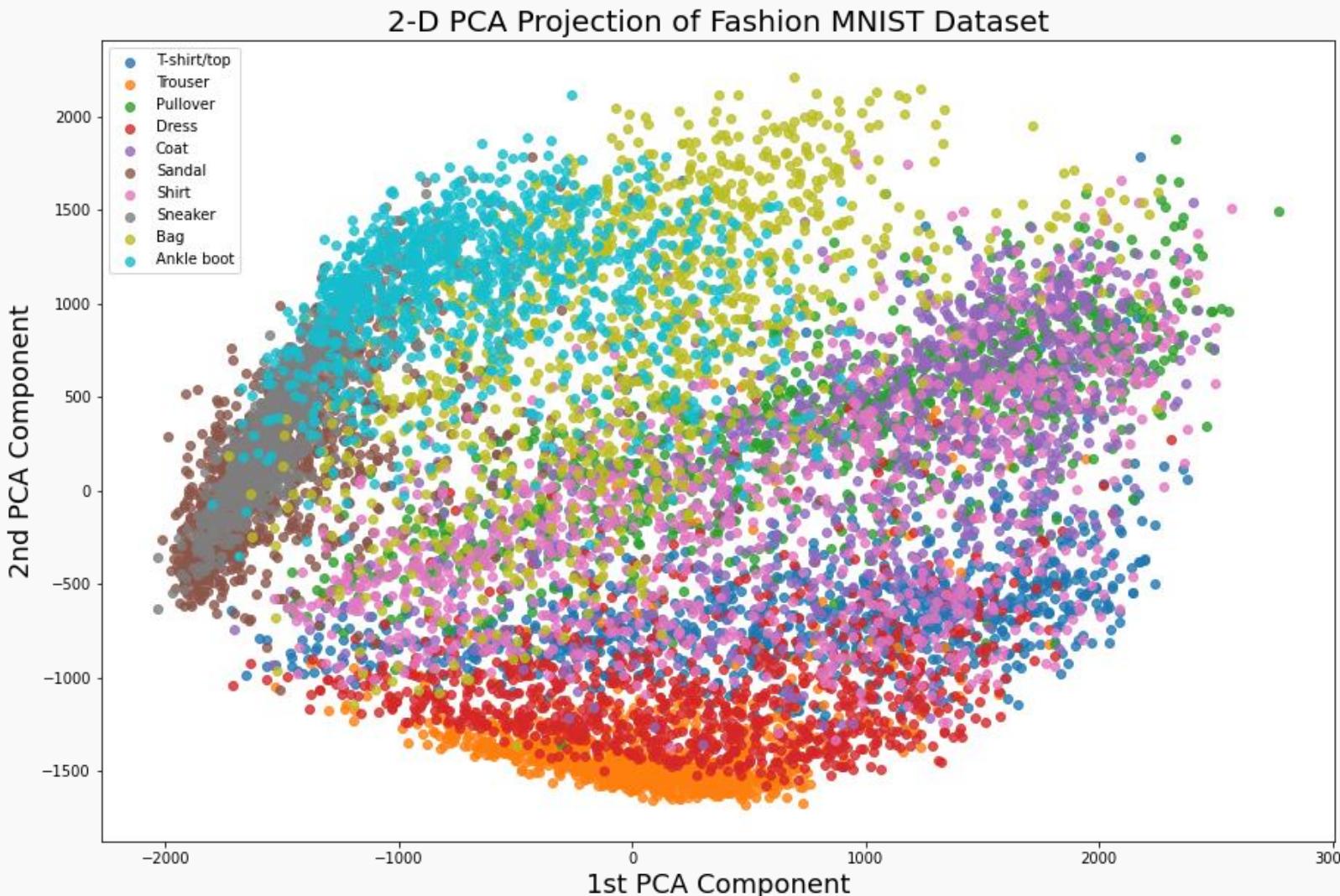
Bag



Ankle boot



PCA for visualizing image data



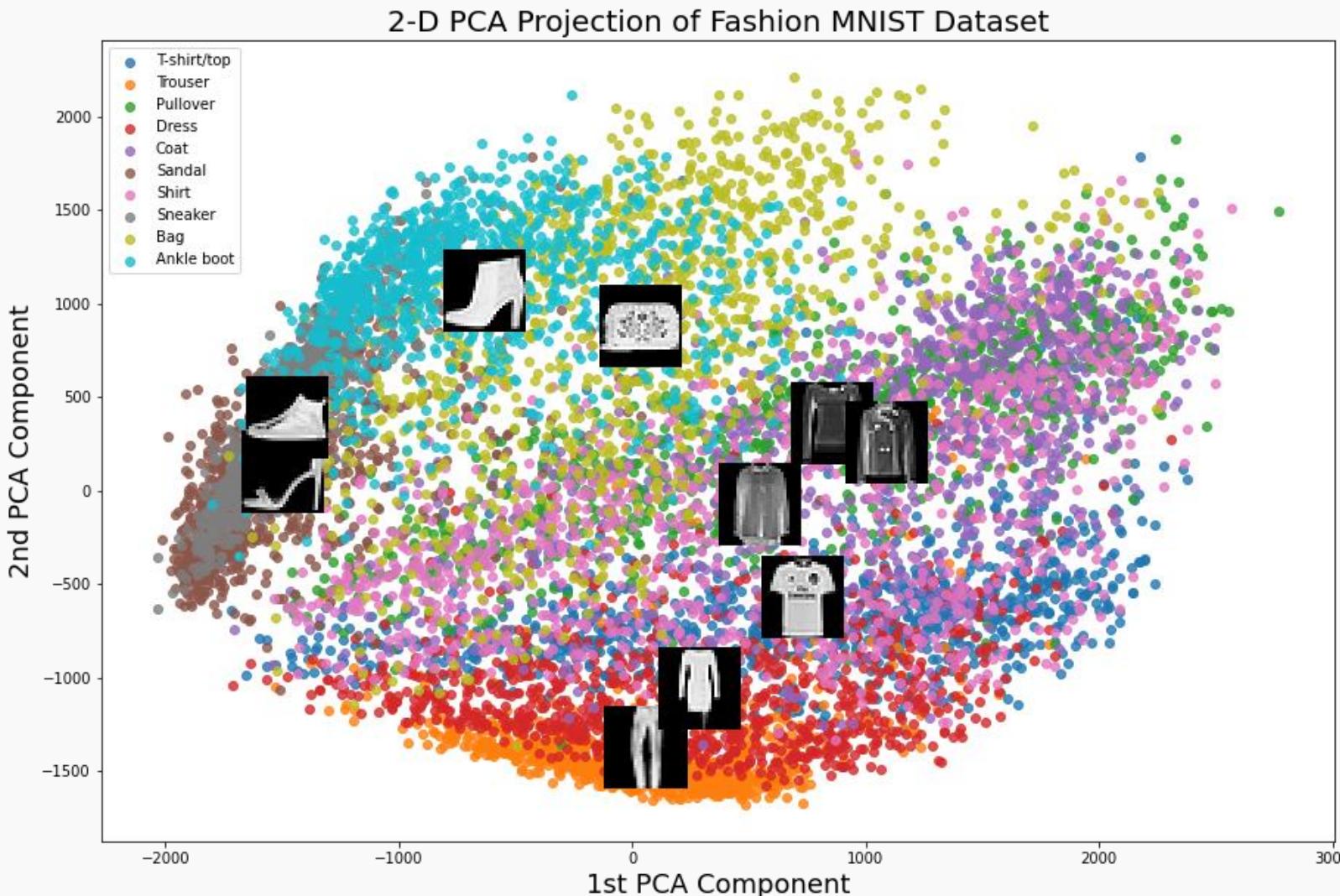
Projecting onto the top 2 principal components allows us to visualize the dataset.

While there exists considerable overlap, we see that most categories are grouped together.

And remember, PCA never actually saw the category labels!



PCA for visualizing image data



Displaying examples at the center of their respective cluster we can appreciate that similar categories are close to one another in the projected space.

All this was done by finding linear combinations of pixels that explained the most variance in the data!



Brief Interlude: What is ‘Big Data’?

In the world of Data Science, the term *Big Data* gets thrown around a lot. What does *Big Data* mean?

A rectangular data set has two dimensions: number of observations (n) and the number of predictors (p). Both can play a part in defining a problem as a *Big Data* problem.

What are some issues when:

- n is big (and p is small to moderate)?
- p is big (and n is small to moderate)?
- n and p are both big?



When n is big

A very large sample size can pose computational a challenge.

- Algorithms can take forever to finish. Estimating the coefficients of a regression model, especially one that does not have closed form (like LASSO), can take a while.
- Model selection and hyperparameters tuning, especially when using cross-validation, exacerbates the problem.

What can we do to fix this computational issue?

- Perform ‘preliminary’ steps (model selection, tuning, etc.) on a subset of the training data set. 10% or less can be justified.



Note: statistical inference is usually ok when n is VERY big.

Keep in mind, big n doesn't solve everything

The era of Big Data (aka, large n) can help us answer lots of interesting scientific and application-based questions, but it does not fix everything.

Remember the old adage: “**garbage in; garbage out**”. If the data are not representative of the population, then modeling results can be terrible. Random sampling ensures representative data.

Xiao-Li Meng does a wonderful job describing the subtleties involved (WARNING: it’s a little technical, but digestible):

<https://www.youtube.com/watch?v=8YLdIDOMEZs>



When p is big

When the number of predictors is large (many interactions, polynomial terms, etc.), lots of issues can occur.

What are some issues that may arise?

- Matrices may not be invertible (issue in OLS).
- Multicollinearity is likely to be present
- Models are susceptible to overfitting

This situation is called *High Dimensionality* and needs to be accounted for when performing data analysis and modeling.



When Does High Dimensionality Occur?

The problem of high dimensionality can occur when the number of parameters exceeds or is close to the number of observations. This can happen when we consider lots of interaction terms, like in our previous example. But this can also happen when the number of main effects is high.

For example:

- When we are performing polynomial regression with a high degree and a large number of predictors.
- When the predictors are genomic markers (and possible interactions) in a computational biology problem.
- When the predictors are the counts of all English words appearing in a text.



How Does sklearn handle unidentifiability?

In a parametric approach: if we have an over-specified model ($p > n$), the parameters are unidentifiable: we only need $n - 1$ predictors to perfectly predict every observation ($n - 1$ because of the intercept).

So, what happens to the ‘extra’ parameter estimates (the extra β ’s)?

The remaining $p - (n - 1)$ predictors’ coefficients can be estimated to be *anything!* Thus, there are an infinite number of sets of estimates that will give us identical predictions. There is not one unique set of β ’s!



Perfect Multicollinearity

The $p > n$ situation leads to perfect collinearity of the predictor set.
But this can also occur with redundant predictors.

Let's see what sklearn does in this situation:

```
regress_sqft = sk.linear_model.LinearRegression().fit(X = homes[['sqft']], y = homes['price'])
print("Intercept =", regress_sqft.intercept_.round(2), ", Slope =", regress_sqft.coef_[0].round(4))

regress_sqft2 = sk.linear_model.LinearRegression().fit(X = homes[['sqft', 'sqft']], y = homes['price'])
print("Intercept =", regress_sqft2.intercept_.round(2), ", Slopes =", regress_sqft2.coef_.round(4))

Intercept = 247438.24 , Slope = 589.7823
Intercept = 247438.24 , Slopes = [294.8911 294.8911]
```



Multicollinearity: A Challenge for Regression

In datasets with many features, especially when they are highly correlated, traditional linear regression struggles.

Correlated predictors can lead to unstable coefficient estimates, making the model less reliable and harder to interpret.

This motivates **Principal Component Regression (PCR)**

PCR is a two-step technique designed to address multicollinearity and reduce dimensionality of the predictors for a regression problem.



PCA for Regression (PCR)

PCA Step: First, can we use the components derived from PCA to transform the original predictors, X , into a set of new uncorrelated predictors, Z .

Regression Step: We then perform linear regression using a subset of these Z predictors.

By using a subset, PCR can simplify the model, reduce overfitting, and improve performance on unseen data.



PCA for Regression (PCR)

PCA is easy in Python, so how can we use it for regression modeling in a real-life problem?

If we use all p of the new Z_i , then we have *not* reduced the dimensionality. To do so, we must select only the first m PCA variables, Z_1, \dots, Z_m , to use as predictors for some $m < p$.

The choice of m is important and can vary from application to application. It depends on various things, like how collinear the predictors are, how truly related they are to the response, etc.

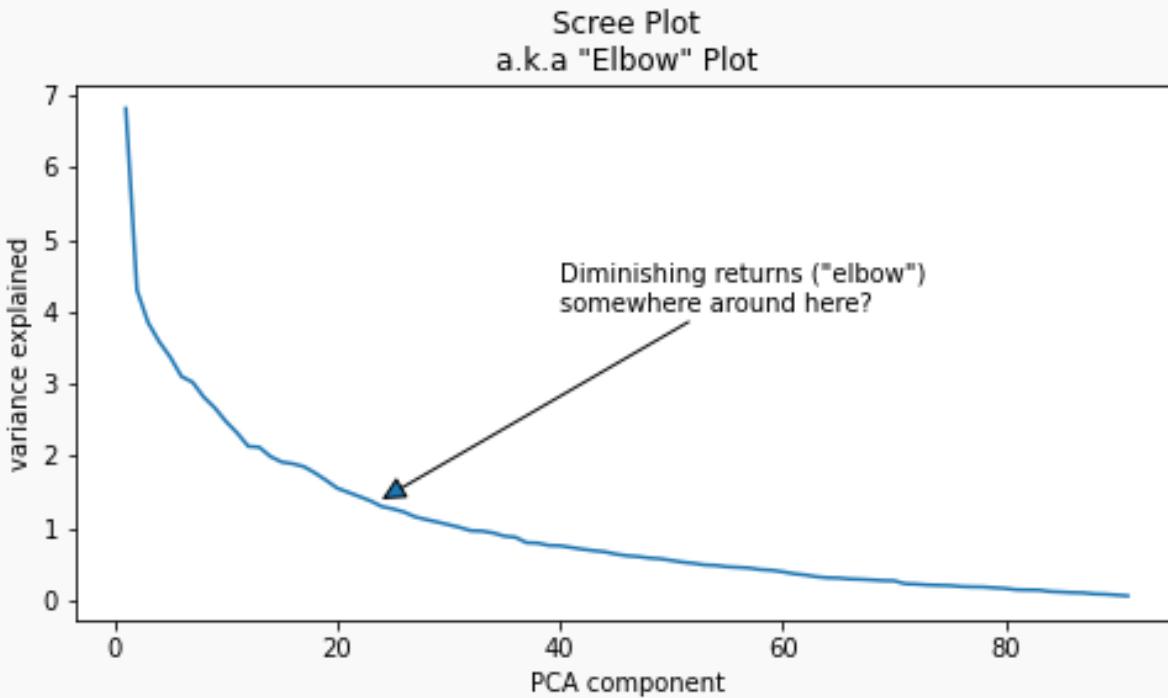
So how should we decide on a value of m ?



How many components to keep?

One approach for deciding on the number of components to keep is to just “eyeball” it.

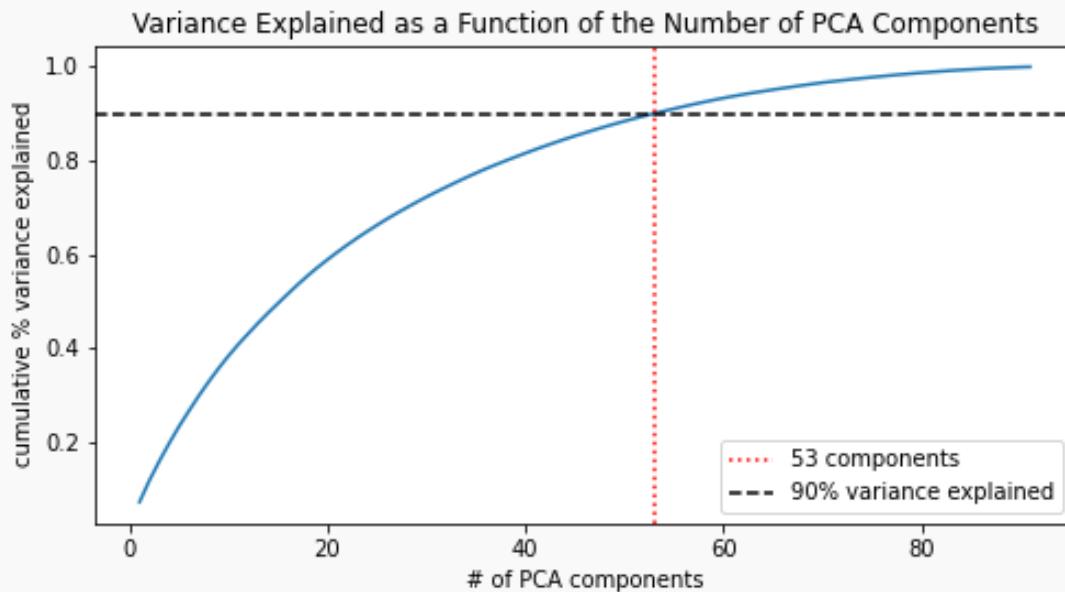
We look for a point of diminishing returns, or “elbow”, in a plot showing the variance explained by each component.



How many components to keep? (cont.)

Another approach is to specify how much of the total variance we want explained by our m components. For example, 90% of the total variance.

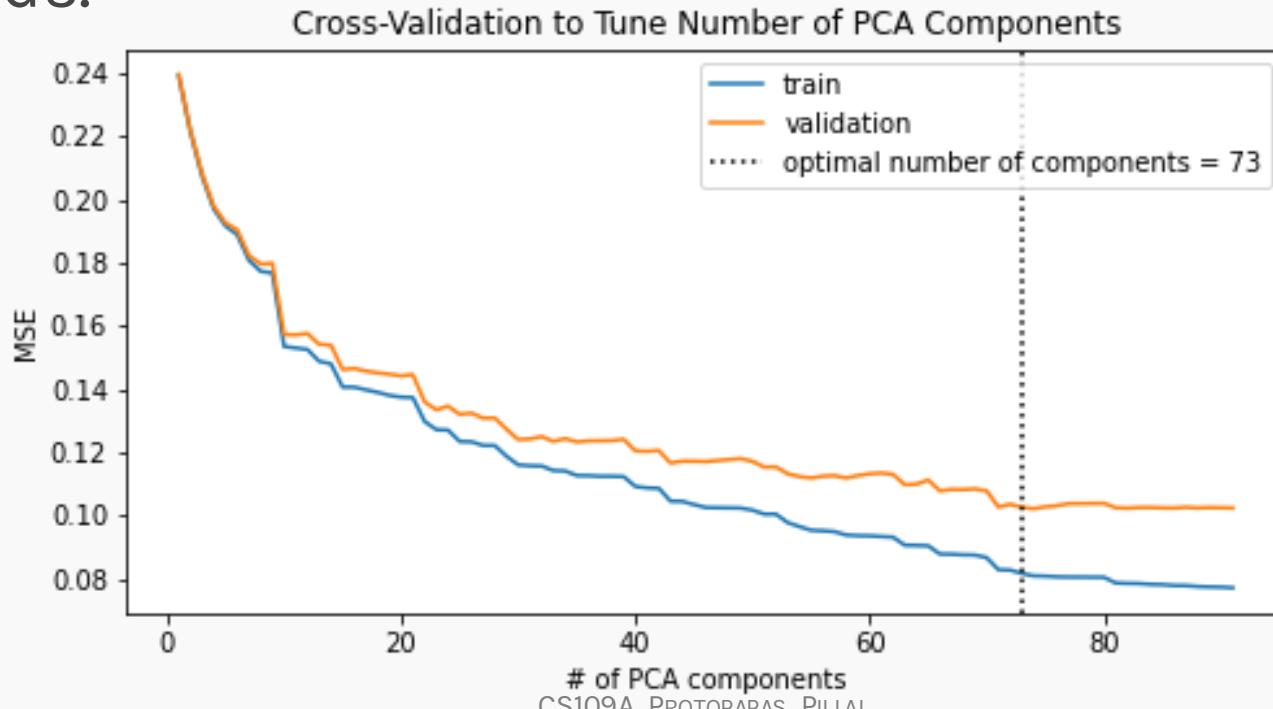
We then inspect the cumulative variance explained by each additional component and stop when we meet our goal.



Cross-validation to tune # of components in PCR

The two previous strategies **did not consider model performance!**

If modeling is your end goal, then you can use cross-validation to *tune* the number of components like any other *hyperparameter*, selecting the number that received the best mean validation error across all folds.



Interpreting the Results of PCR

A PCR can be interpreted in terms of original predictors...very carefully.

Each estimated β coefficient in the PCR can be *distributed* across the predictors via the associated component vector, w .

An example is worth a thousand words:

```
pcaX_df = pd.DataFrame(pcaX, columns=[f'PCA{i}' for i in range(1,X.shape[1]+1)])  
  
PCR_simple = sk.linear_model.LinearRegression().fit(X = pcaX_df[['PCA1']], y = homes['price'])  
print("PCA Intercept =",PCR_simple.intercept_.round(2),", PCA Slope =",PCR_simple.coef_[0].round(4))  
  
print("First PCA Component vector (w1):",W[0,:])  
  
PCA Intercept = 1244.2 , PCA Slope = 0.2049  
First PCA Component vector (w1): [ 3.23799939e-01  9.46124307e-01 -7.23671683e-04  9.22493237e-04  
 1.00975436e-03]
```

So how can this be transformed back to the original variables?

$$\begin{aligned}\hat{Y} &= \hat{\beta}_0 + \hat{\beta}_1 Z_1 = \hat{\beta}_0 + \hat{\beta}_1 (\vec{w}_1^T X) = \hat{\beta}_0 + (\hat{\beta}_1 \vec{w}_1^T) X \\ &= 1244.2 + 0.2049 \cdot (0.3238X_1 + 0.9461X_2 - 0.00072X_3 + 0.00092X_4 + 0.00101X_5) \\ &= 1244.2 + 0.0663X_1 + 0.194X_2 - 0.00015X_3 + 0.00019X_4 + 0.00021X_5\end{aligned}$$



Interpreting the Results of PCR

You can always put the PCR coefficients in terms of the original predictors.

$$\hat{y} = \mathbf{Z}\beta_Z = (\mathbf{X}\mathbf{W}_m)\beta_Z = \mathbf{X}(\mathbf{W}_m\beta_Z) = \mathbf{X}\beta_X$$

\mathbf{X} is our original dataset, centered to have a mean of zero

\mathbf{W}_m is the matrix whose columns are the first m eigenvectors of $\mathbf{X}^\top \mathbf{X}$

\mathbf{Z}_m is the transformed dataset created by projecting \mathbf{X} onto the principal components contained in \mathbf{W}_m

β_Z are the coefficients for the PCR model

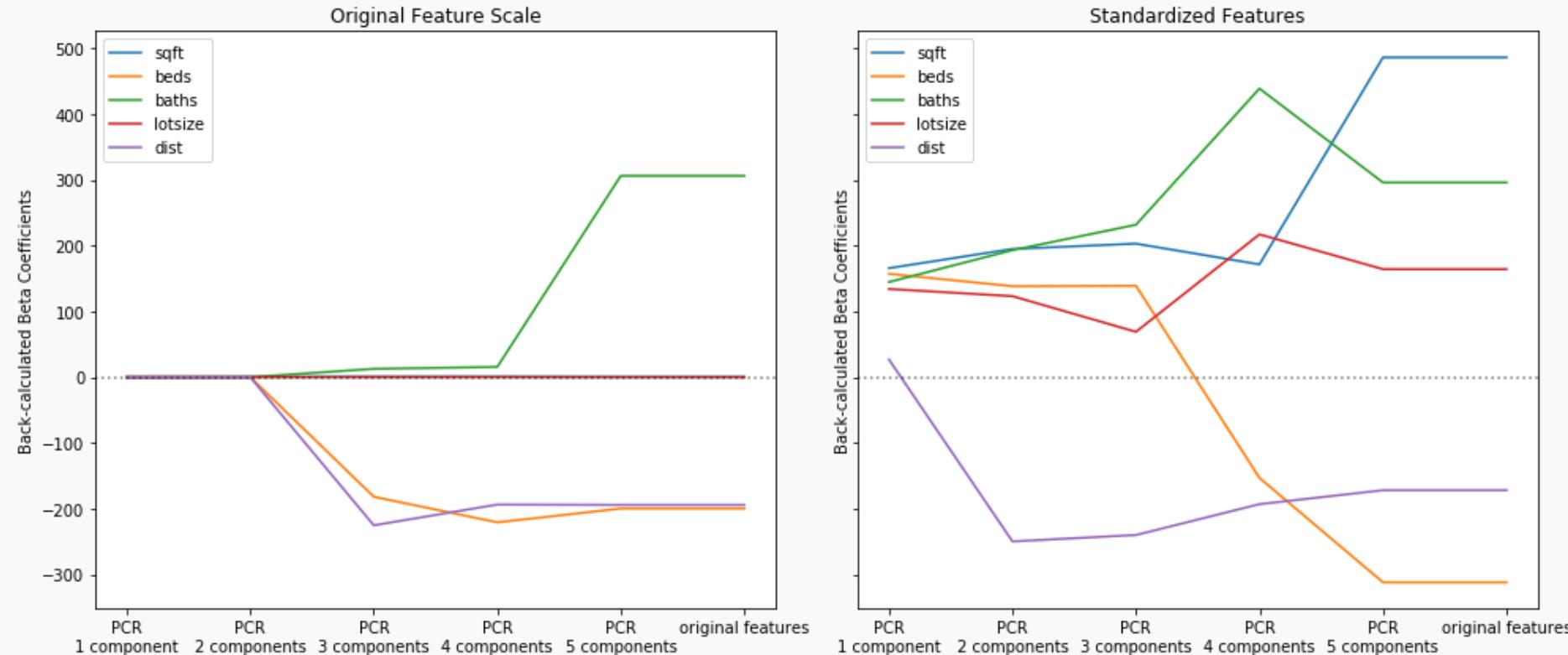
β_X are coefficients in terms of the original predictors

When $m = p$ then the recovered coefficients are identical to those from a model fit on the original predictors.



Coefficient ‘paths’ as more components enter the PCR

Coefficients for the PCR model using all components are identical to coefficients found when fitting on the original predictors.



Notice the effect standardizing has on the coefficients.



A few notes on using PCA

- PCA **cons**:
 1. PCA *knows nothing about the response variable*. Component vectors as predictors might not be ordered from best to worst!
 2. *Direct* interpretation of coefficients in PCR is lost, so if easy interpretation is important to you, perhaps steer clear.
 3. PCA can often fail to improve the predictive power of a model.
- PCA **pros**:
 1. Can help avoid the curse of dimensionality and overfitting.
 2. Easy to visualize how predictive your features are of the response variable, especially in the classification setting.
 3. Reduces multicollinearity, and so can improve the computational time when fitting models.



Matrix Completion (a brief aside)

Matrix Completion is another application of PCA and is suitable for imputing data which are missing at random. This approach is commonly used in *recommender systems* which deal in very large sparse matrices.

Consider an $n \times p$ matrix of movie ratings by Netflix customers where n is the number of customers and p is the number of movies. Such a matrix will certainly contain many missing entries.

Imputing these missing values well is equivalent to predicting what customers will think of movies they haven't seen yet.

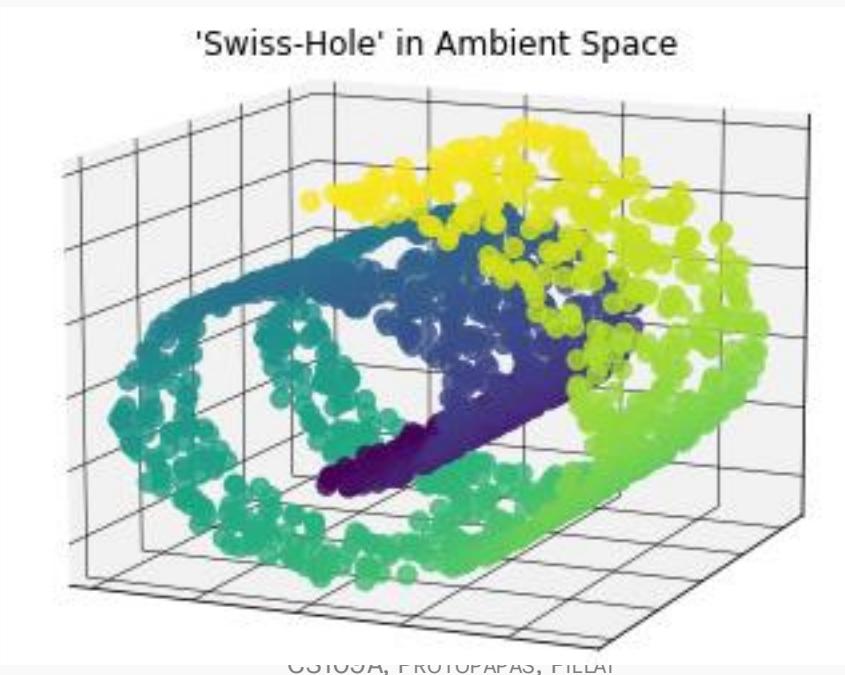
You can read more about the matrix completion algorithm in your [textbook](#).
(section 12.3 pg. 510)



PCA: Limitations of Linearity

We've seen that PCA projects your data onto the hyperplane (i.e., a linear subspace) that minimizes the distance between the original data and the projected data...

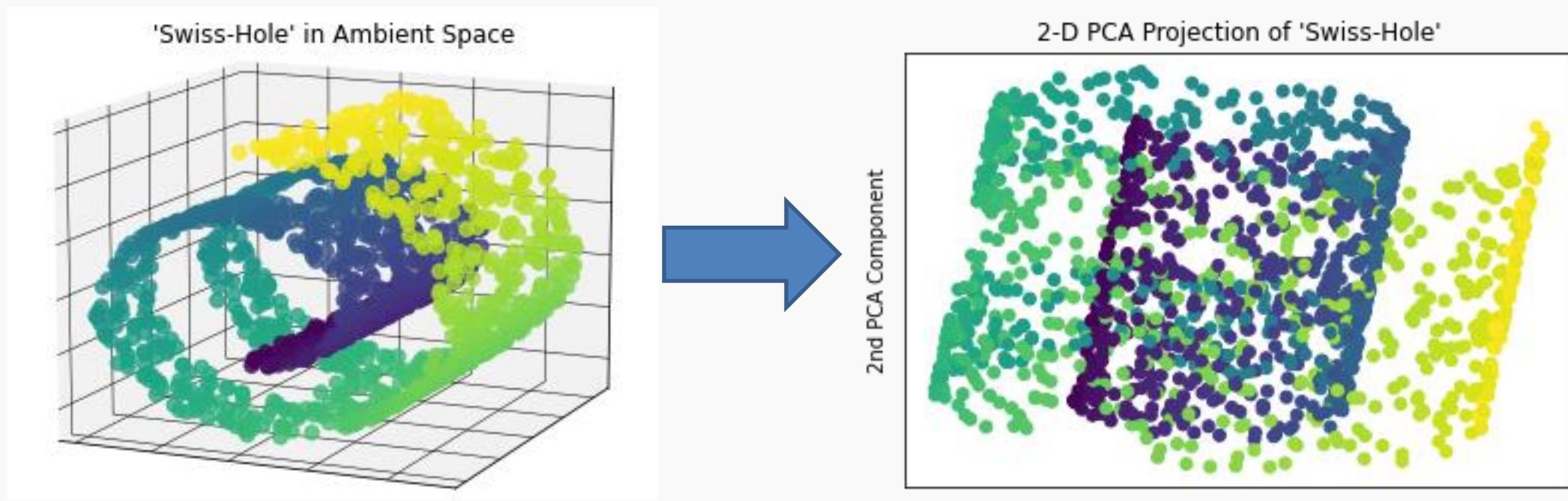
...but what if your data clearly do *not* lie on a hyperplane within the ambient space!



PCA ‘squashes’ the Swiss-Hole

What are the results of trying PCA here? It isn’t pretty.

PCA doesn’t respect the coiled, 2-D manifold embedded in the 3-D space. Points that were far apart on the original manifold end up close together in the PCA projection.



PCA: Preserving Global Pairwise Distances

You can think of PCA as striving to preserve the structure of the original data. PCA interprets ‘structure’ as ‘pairwise distances.’

As a result, points which are (relatively) close in the original space should remain close in the projected space. Likewise, points which were distant in the original space should remain distance once projected.

A different strategy would be to relax this demand a bit. What if we focus on preserving only *‘local’ structure*? That is, keep nearby point close, but don’t worry too much about preserving the pairwise distances of more distant points.



t-SNE: t-distributed Stochastic Neighbor Embeddings

The t-SNE algorithm attempts to minimize the following loss function:

$$\mathcal{L} = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

p_{ij} = affinity between points i and j in the **original** space

q_{ij} = affinity between points i and j in the **target** space

You can think of ‘affinity’ as roughly the inverse of distance. The p affinity is calculated using a Gaussian kernel normalized to sum to 1 across all points. For the q affinity, a t -distribution kernel is used.

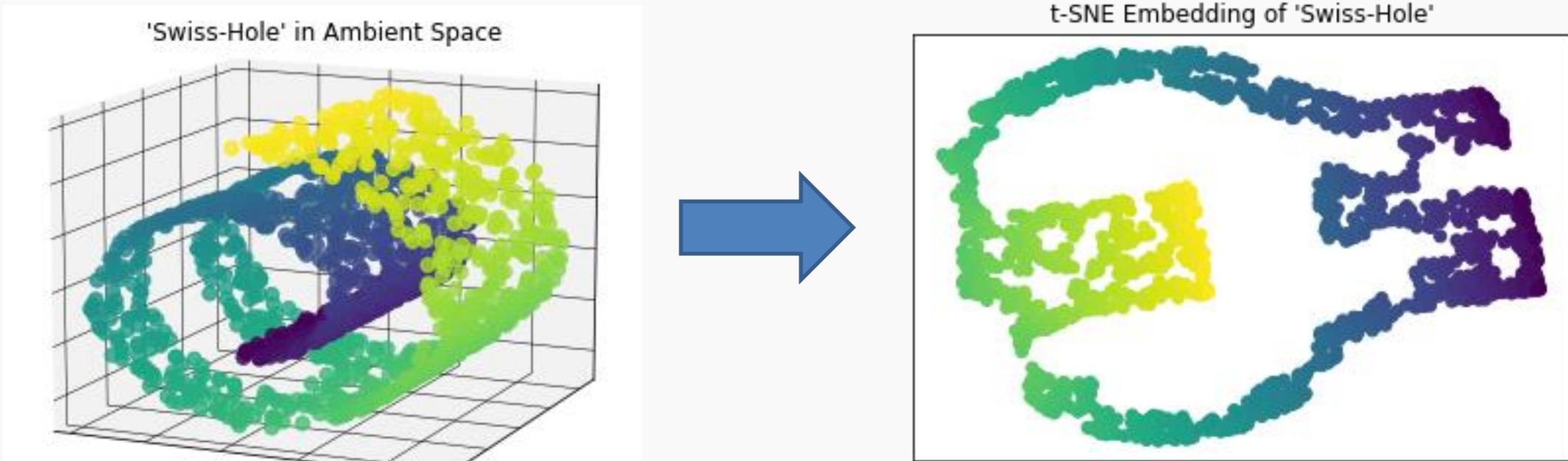
You can see that the loss suffers most when we map points that were originally close to points far away in the target space.



t-SNE ‘unrolls’ the Swiss-Hole

With t-SNE we've managed to reduce the dimensionality while preserving the *local* structure of the original data.

t-SNE is just one example of an algorithm that learns a low dimensional manifold that captures the *intrinsic* dimensionality of the original data.



Revisiting Fashion MNIST with t-SNE

The ‘swiss-role’ had an intrinsic dimensionality of 2 while being embedded in a 3-D ambient space.

Fashion MNIST data may have 784 predictors, but it is perhaps best to also think of this as the ambient space.

As an experiment, generate random vectors of 784 pixel values. How long until you happen to generate an image of a pair of pants? We’ll wait...

Images of clothes live on some lower dimensional manifold embedded within the space of all 28x28 grayscale images.



Assuming the manifold is linear is asking too much

The space of 28x28 grayscale images include not just images of clothes, but all possible 28x28 images! Not to mention the countless ‘noise’ images depicting nothing at all.

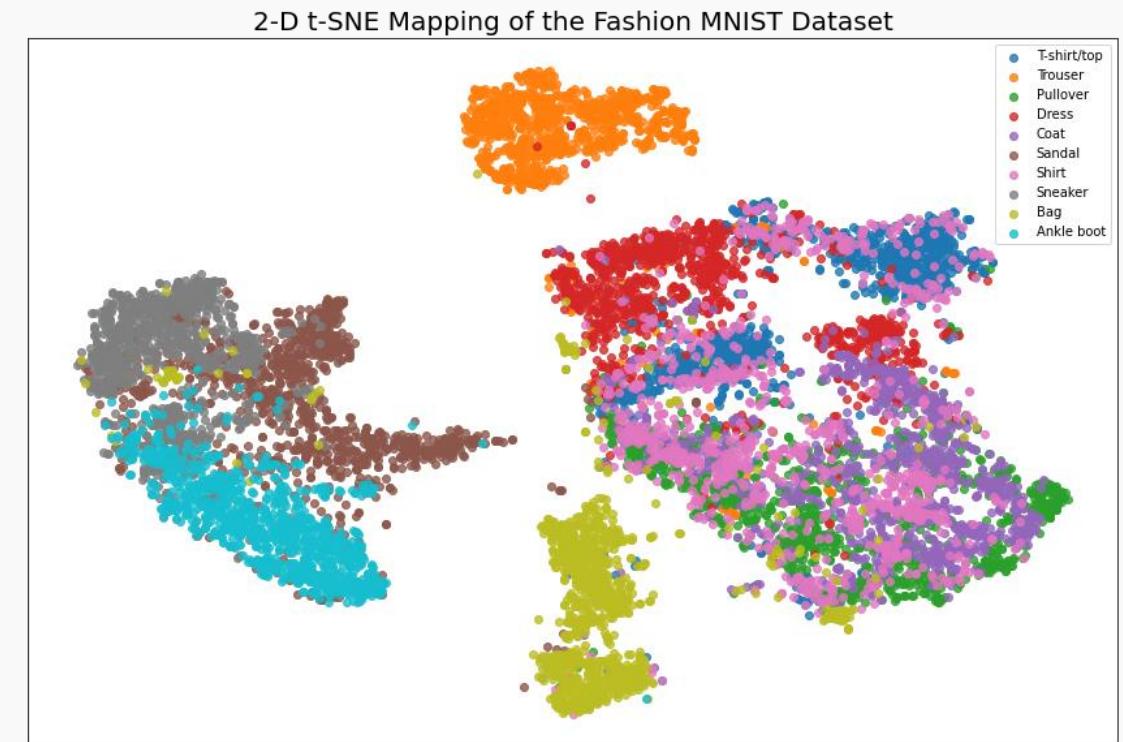
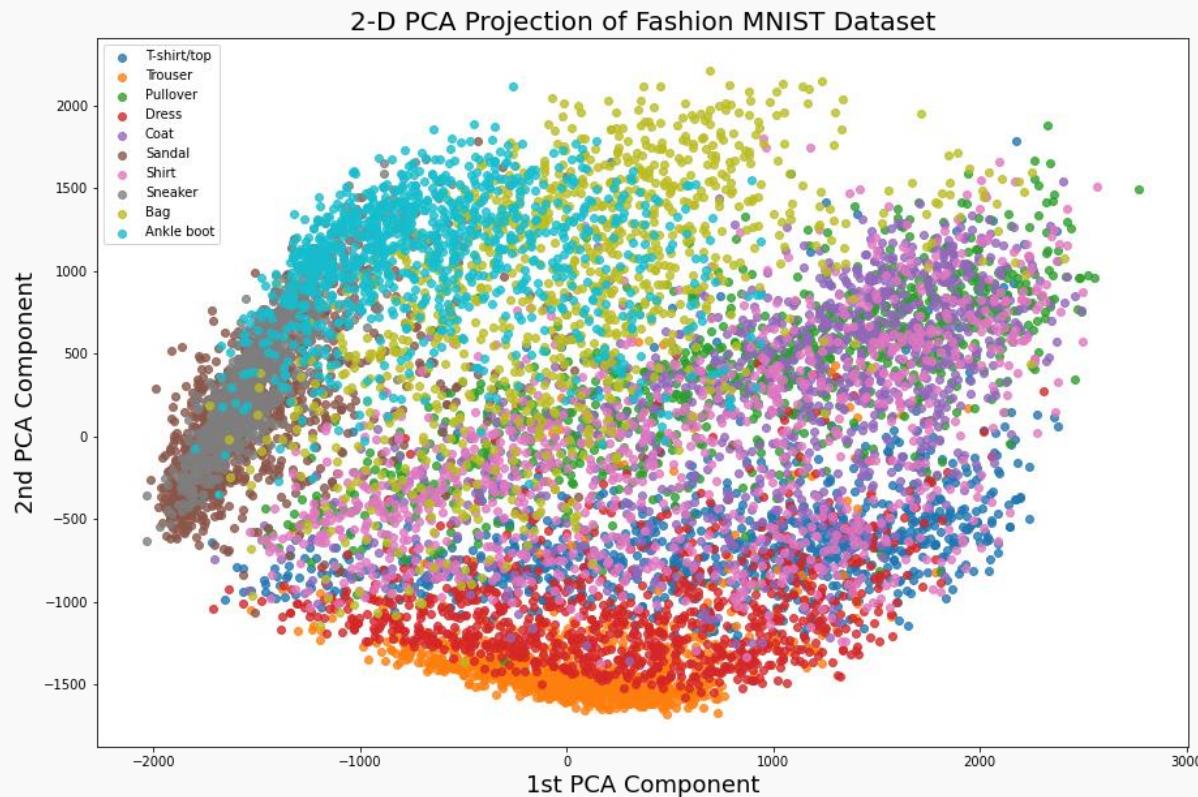
You can think of images of clothes as living on some lower dimensional manifold embedded within the ambient 784-D space.

PCA limits us to finding hyperplanes in the *ambient* linear space. We don’t have any reason to believe that the actual manifold is a linear subspace. *t*-SNE is more flexible.



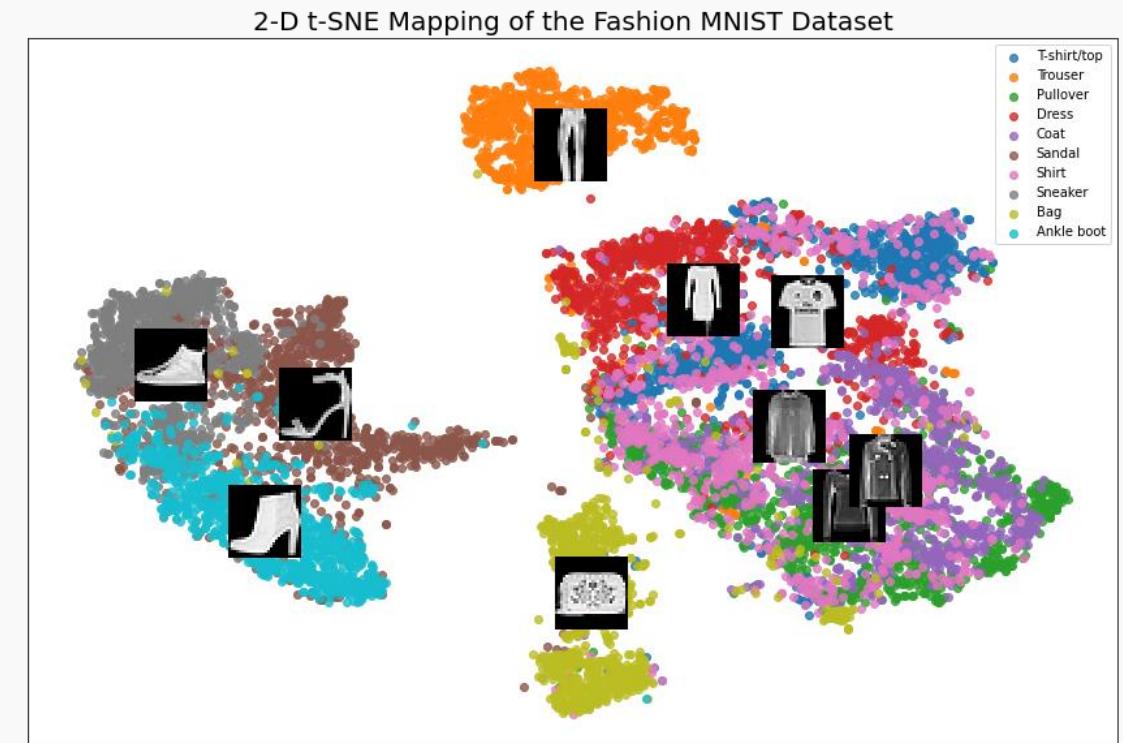
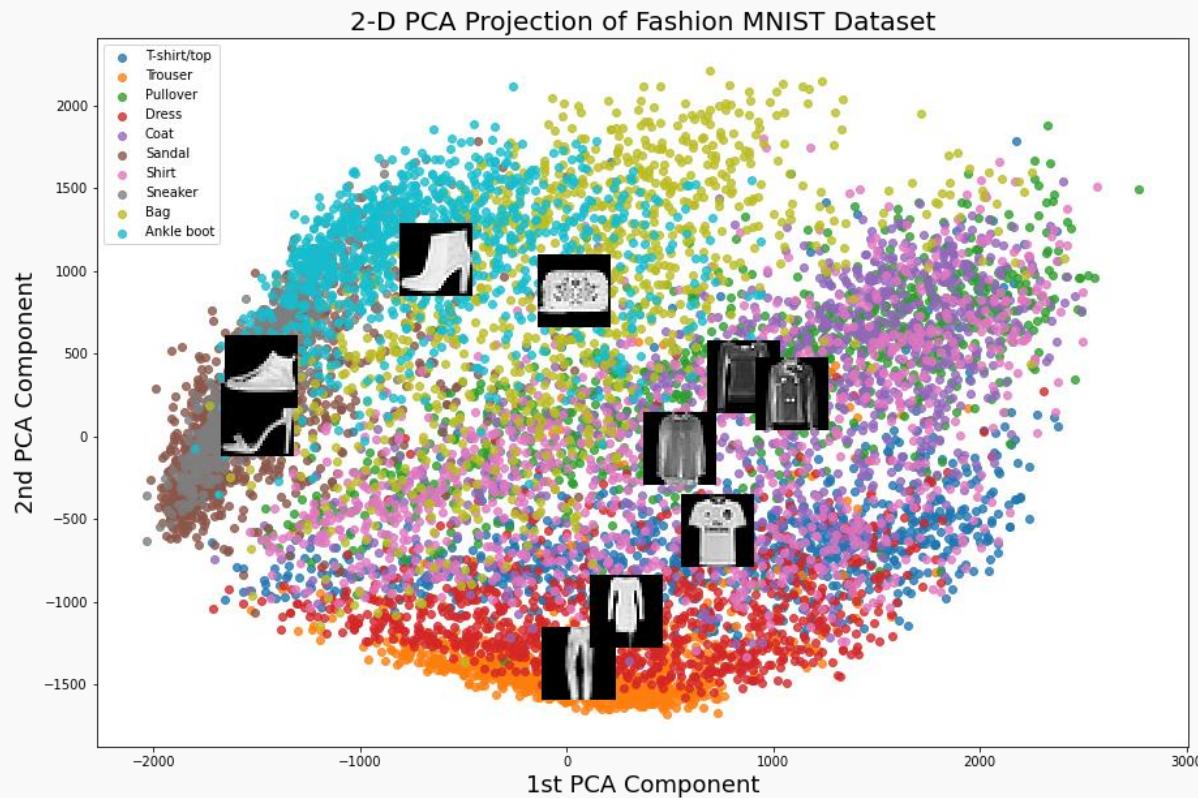
Fashion MNIST: PCA vs t-SNE

In persevering local structure, t-SNE ends up providing more separation.



Fashion MNIST: PCA vs t-SNE (with images)

Unique categories, like pants and bags, end up as little islands.



Challenges of t-SNE

- There is randomness in the algorithm so your results will vary (the ‘S’ is for ‘**stochastic**’).
- It is an iterative algorithm and can be **very slow**. Some preliminary dimensionality reduction is often used to speed things up. You can use PCA for this!
- It has **hyperparameters** to be tuned: number of iterations, learning rate, number of neighboring points to consider ('perplexity'), etc.
- Doesn't have anything like PCA's components and so is **hard to interpret**. This makes it primarily a tool for visualization.



Prediction Intervals



CS109A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai and Chris Gumb

Outline

Part A and B: Assessing the Accuracy of the Coefficient Estimates

Bootstrapping and confidence intervals

Part C: How well do we know \hat{f}

The confidence intervals of \hat{f}

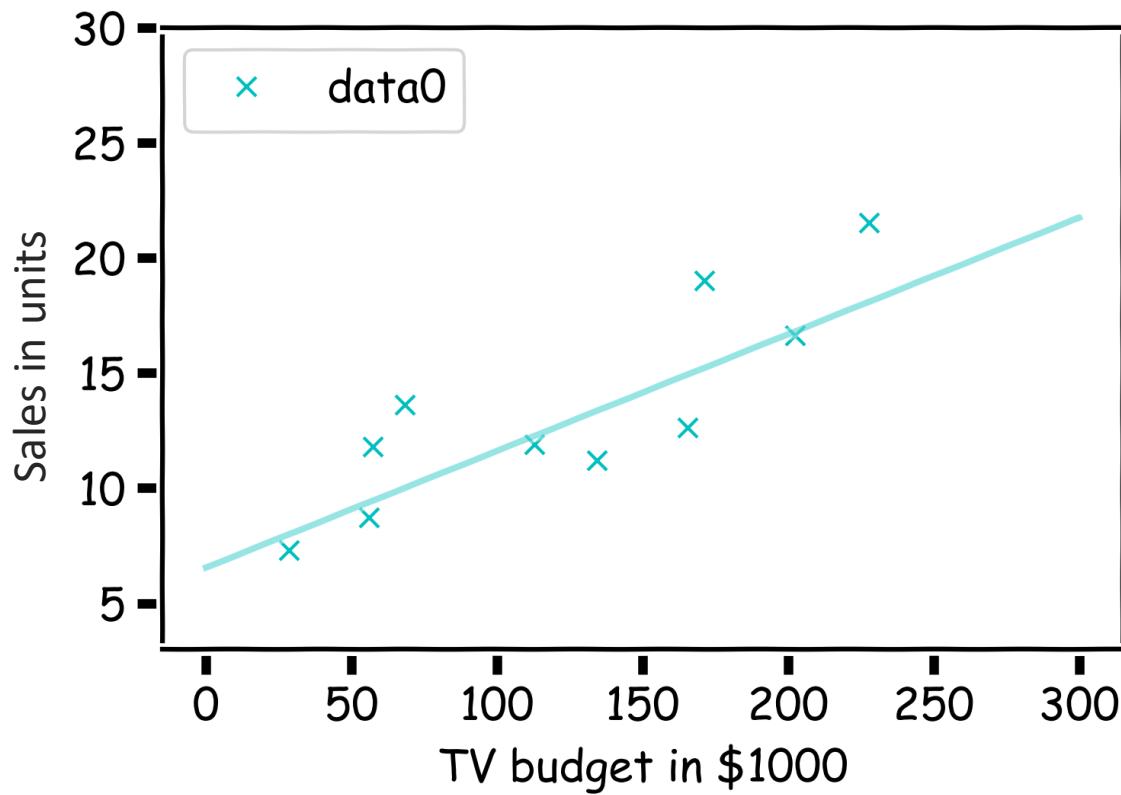
Part D: Evaluating Significance of Predictors

Does the outcome depend on the predictors?

Hypothesis testing

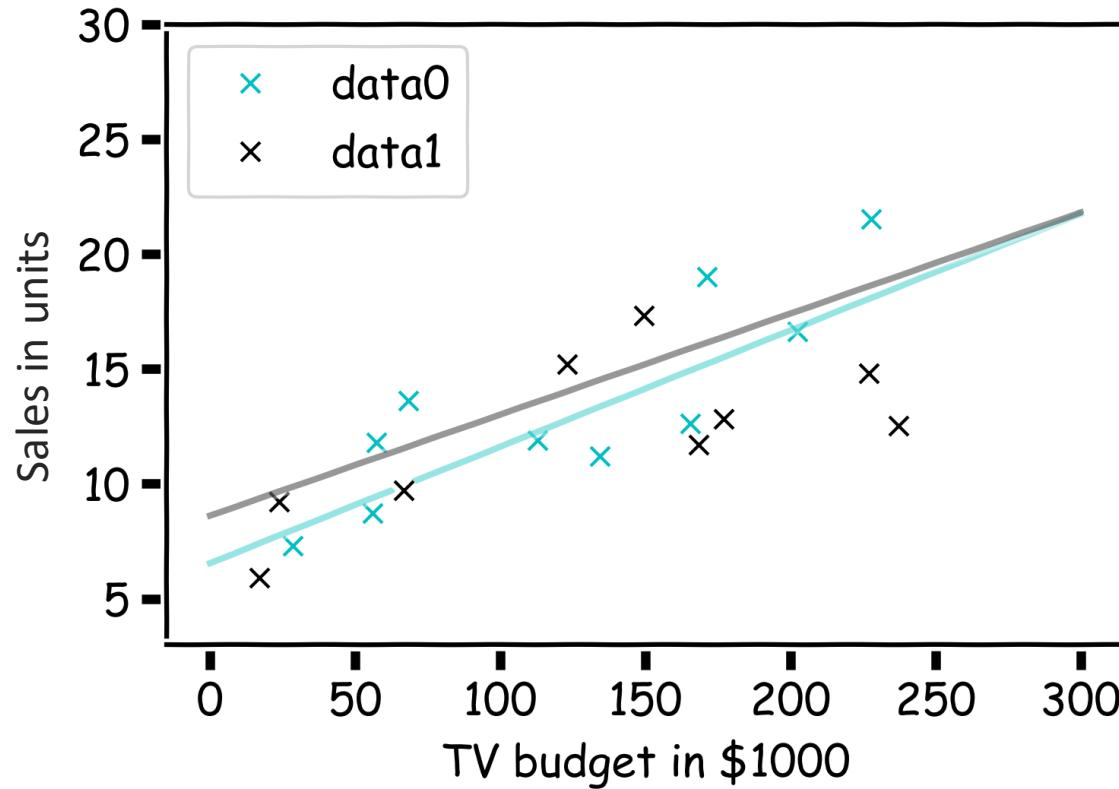
How well do we know \hat{f} ?

Our confidence in f is directly connected with our confidence in β s. For each bootstrap sample, we have one β , which we can use to determine the model, $f(x) = X\beta$.



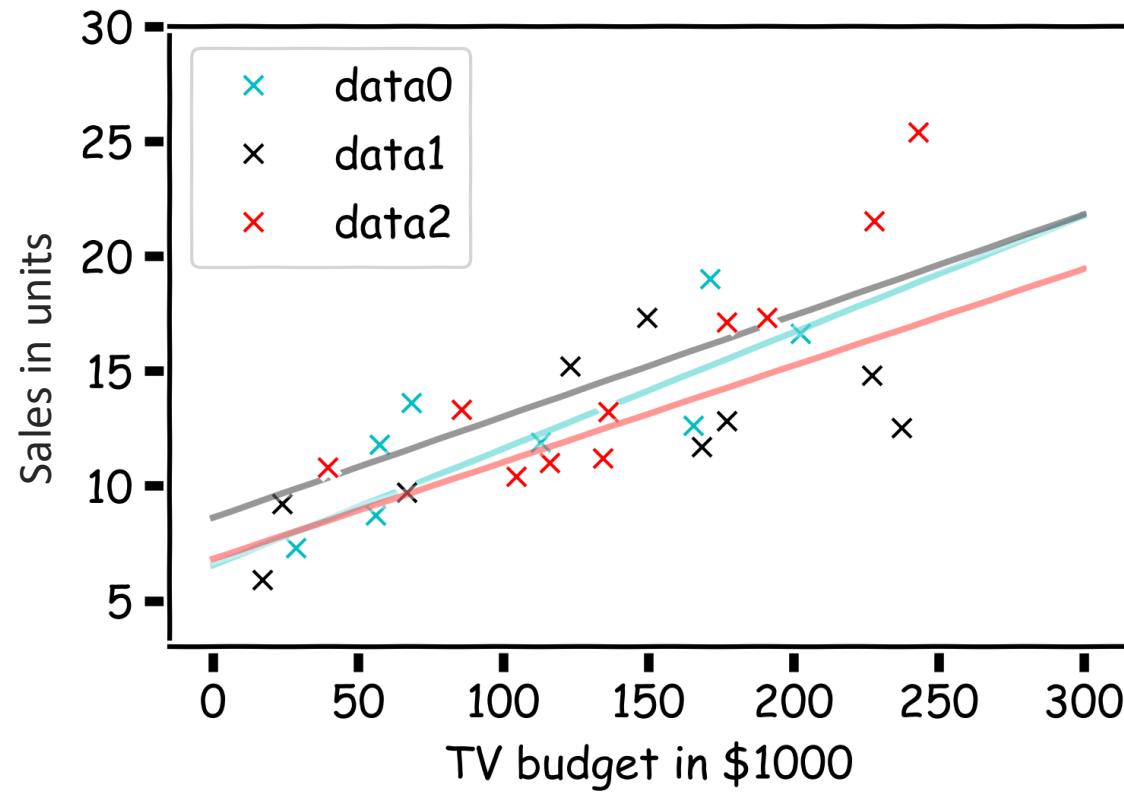
How well do we know \hat{f} ?

Here we show two different models' predictions given the fitted coefficients, fit on two separate bootstrapped sets of data.



How well do we know \hat{f} ?

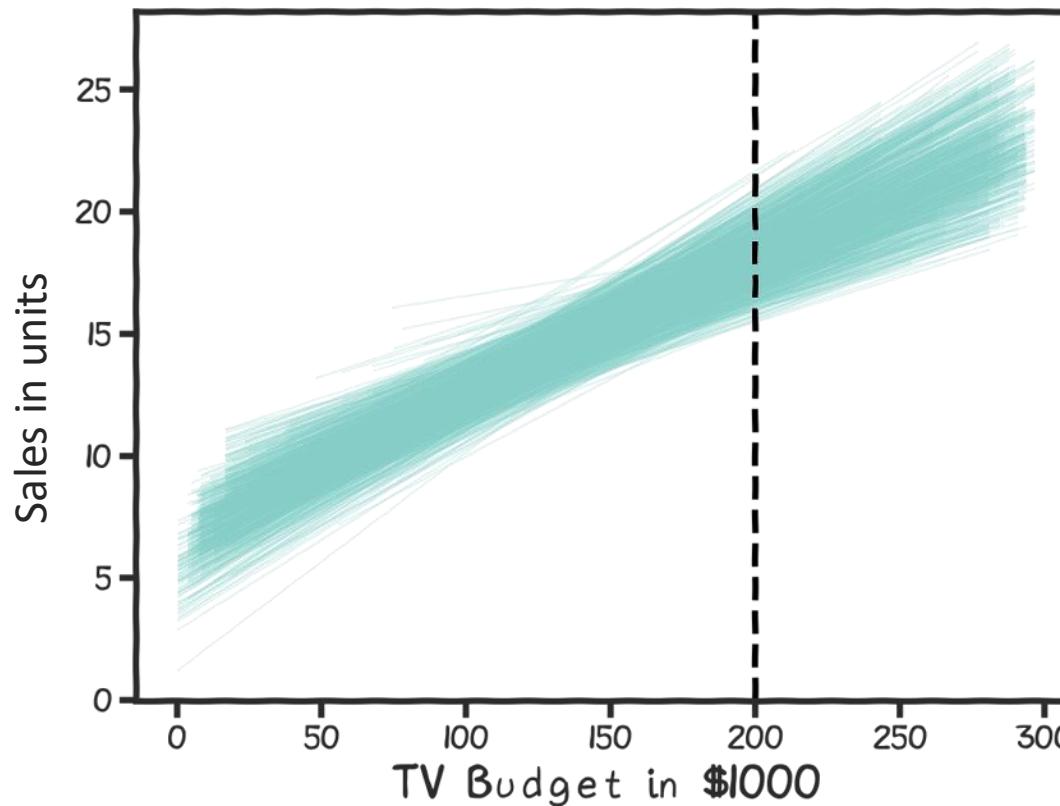
There is one such regression line for every bootstrapped sample.



How well do we know \hat{f} ?

Below we show all regression lines for a thousand of such bootstrapped samples.

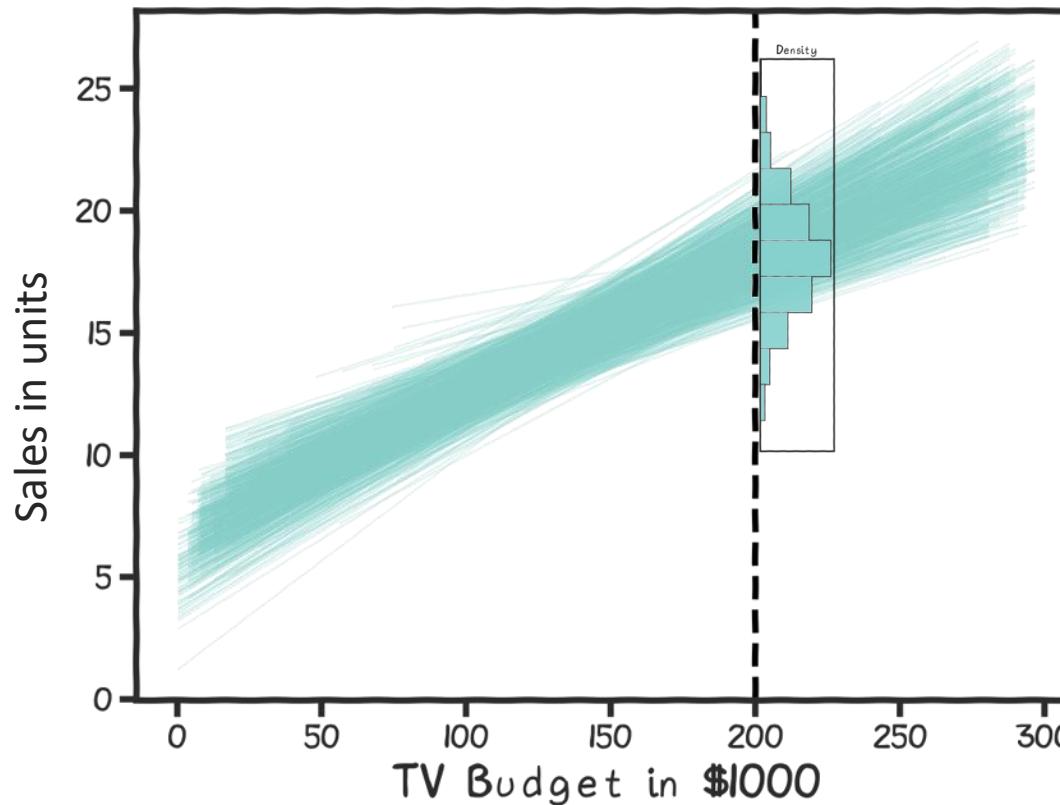
For a given x , we examine the distribution of f and determine the mean and standard deviation (or extract the desired quantiles).



How well do we know \hat{f} ?

Below we show all regression lines for a thousand of such bootstrapped samples.

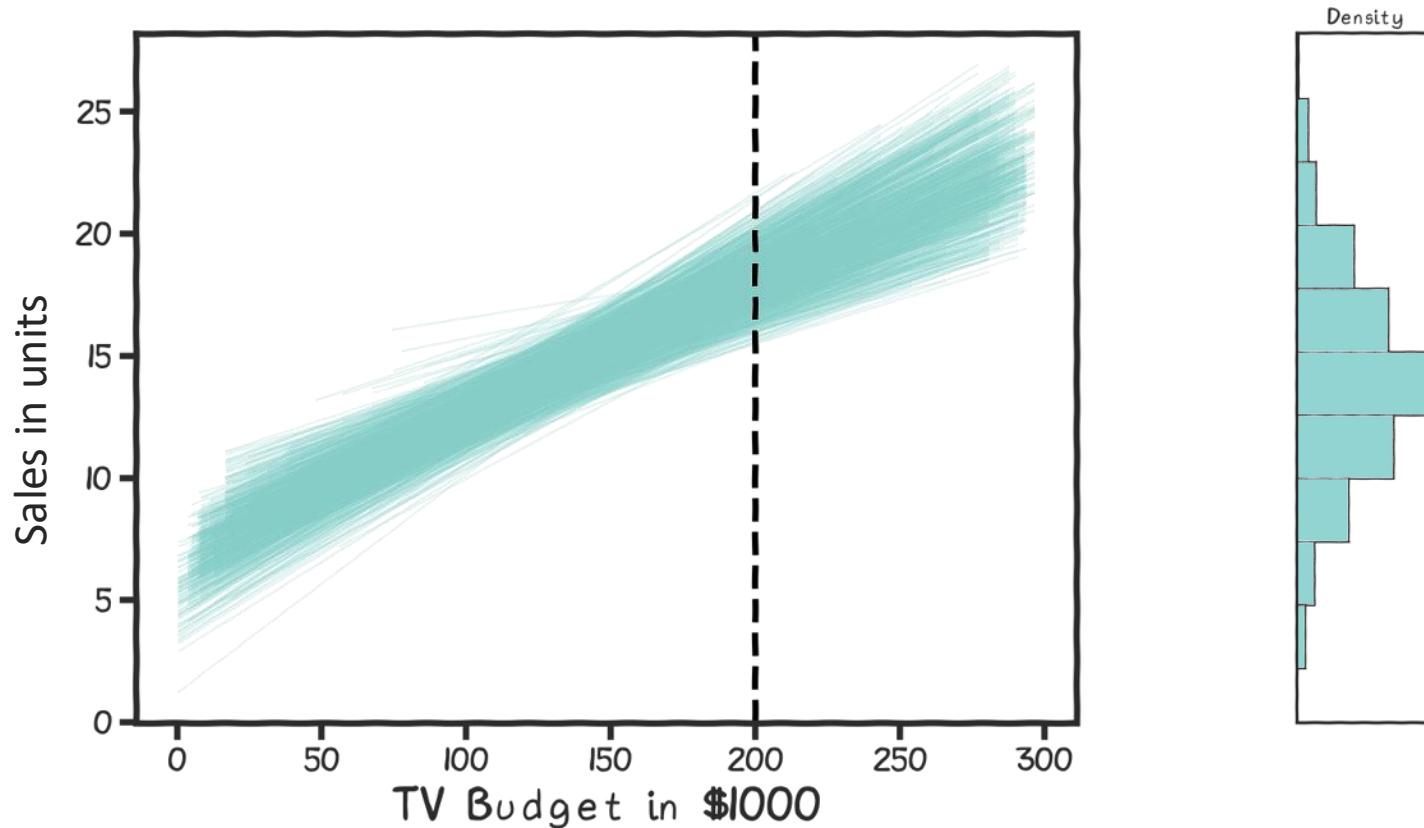
For a given x , we examine the distribution of f and determine the mean and standard deviation (or extract the desired quantiles).



How well do we know \hat{f} ?

Below we show all regression lines for a thousand of such bootstrapped samples.

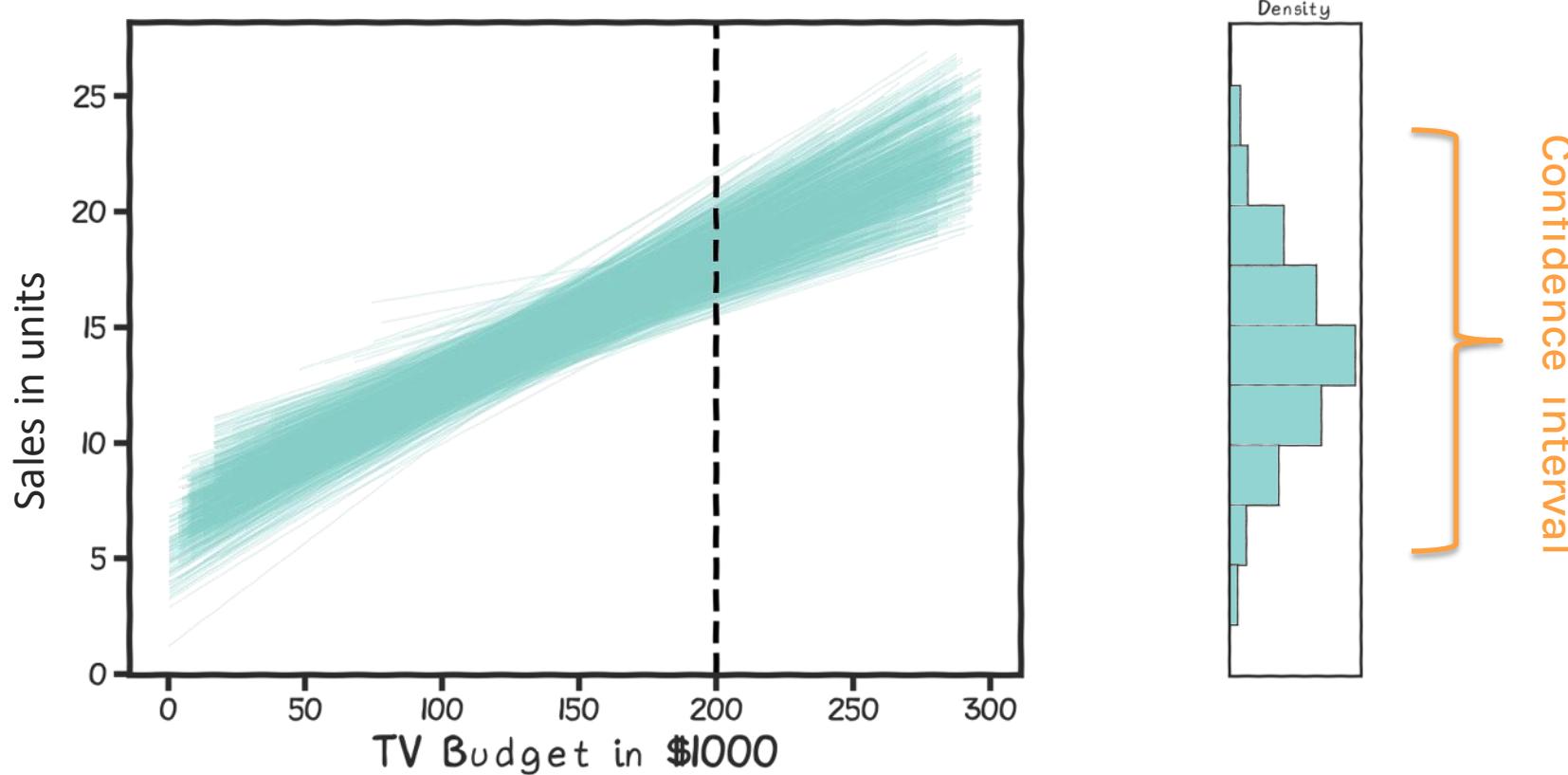
For a given x , we examine the distribution of f and determine the mean and standard deviation (or extract the desired quantiles).



How well do we know \hat{f} ?

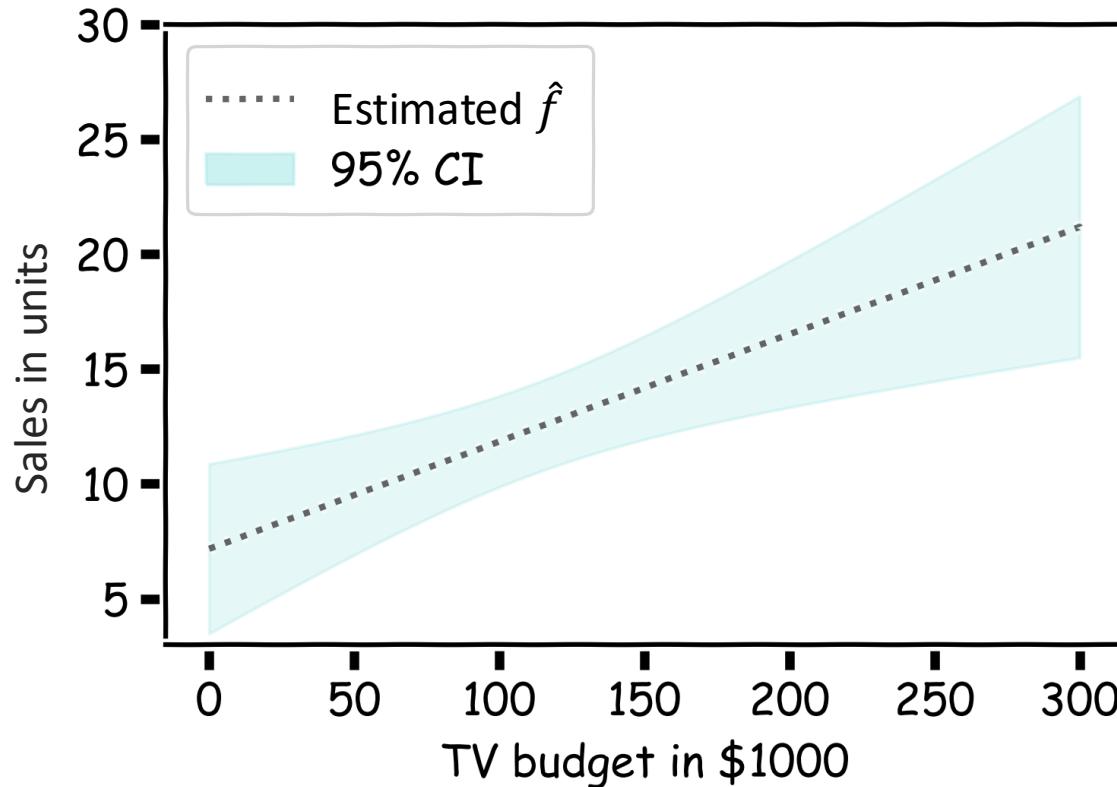
Below we show all regression lines for a thousand of such bootstrapped samples.

For a given x , we examine the distribution of f and determine the mean and standard deviation (or extract the desired quantiles).



How well do we know \hat{f} ?

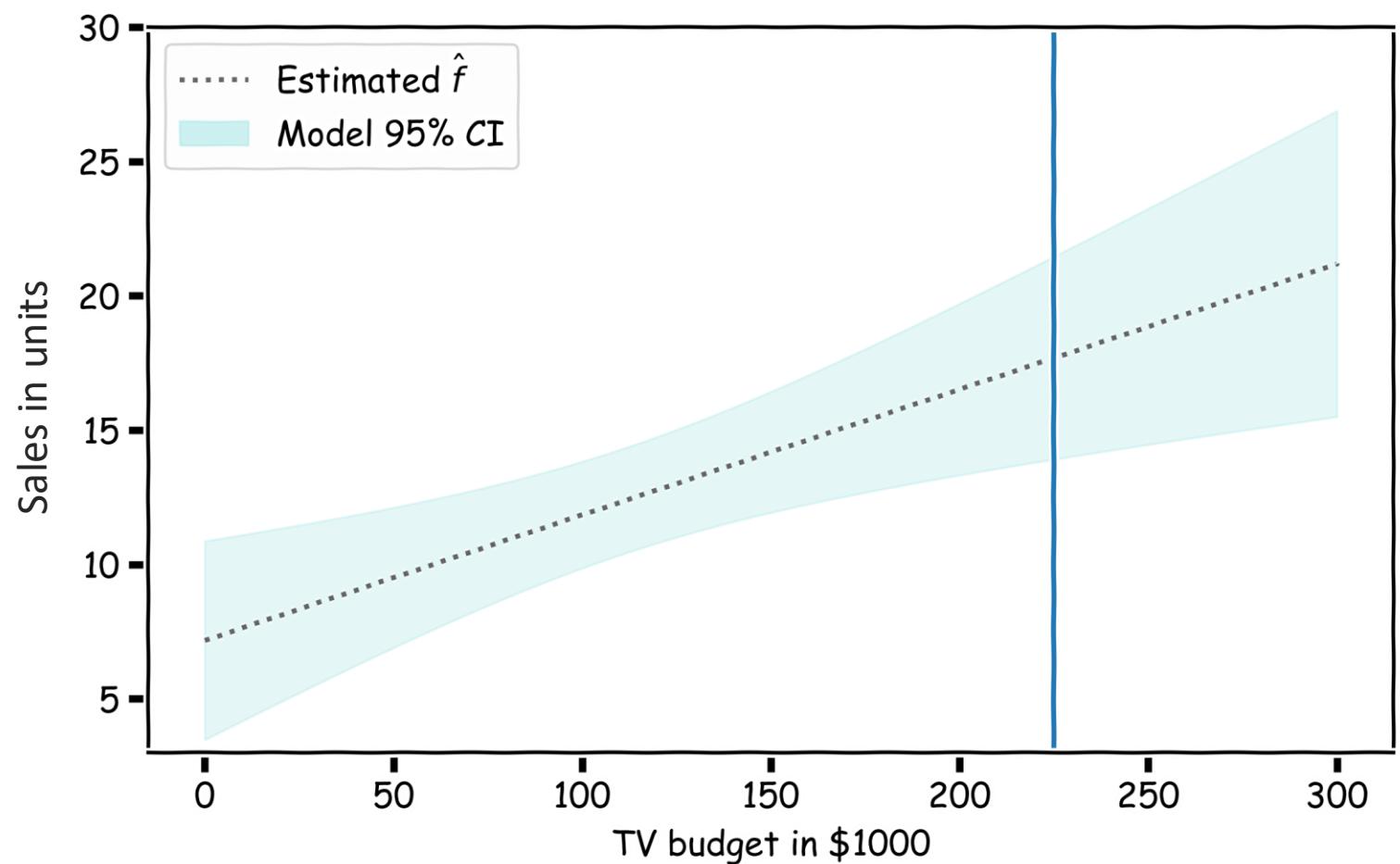
For every x , we calculate the mean of the models, $\widehat{\mu}_f$ (shown with dotted line) and the 95% CI of those models (shaded area).



Confidence in predicting \hat{y}

Even if we knew $f(x)$, the response value cannot be predicted perfectly because of the random error in the model (irreducible error).

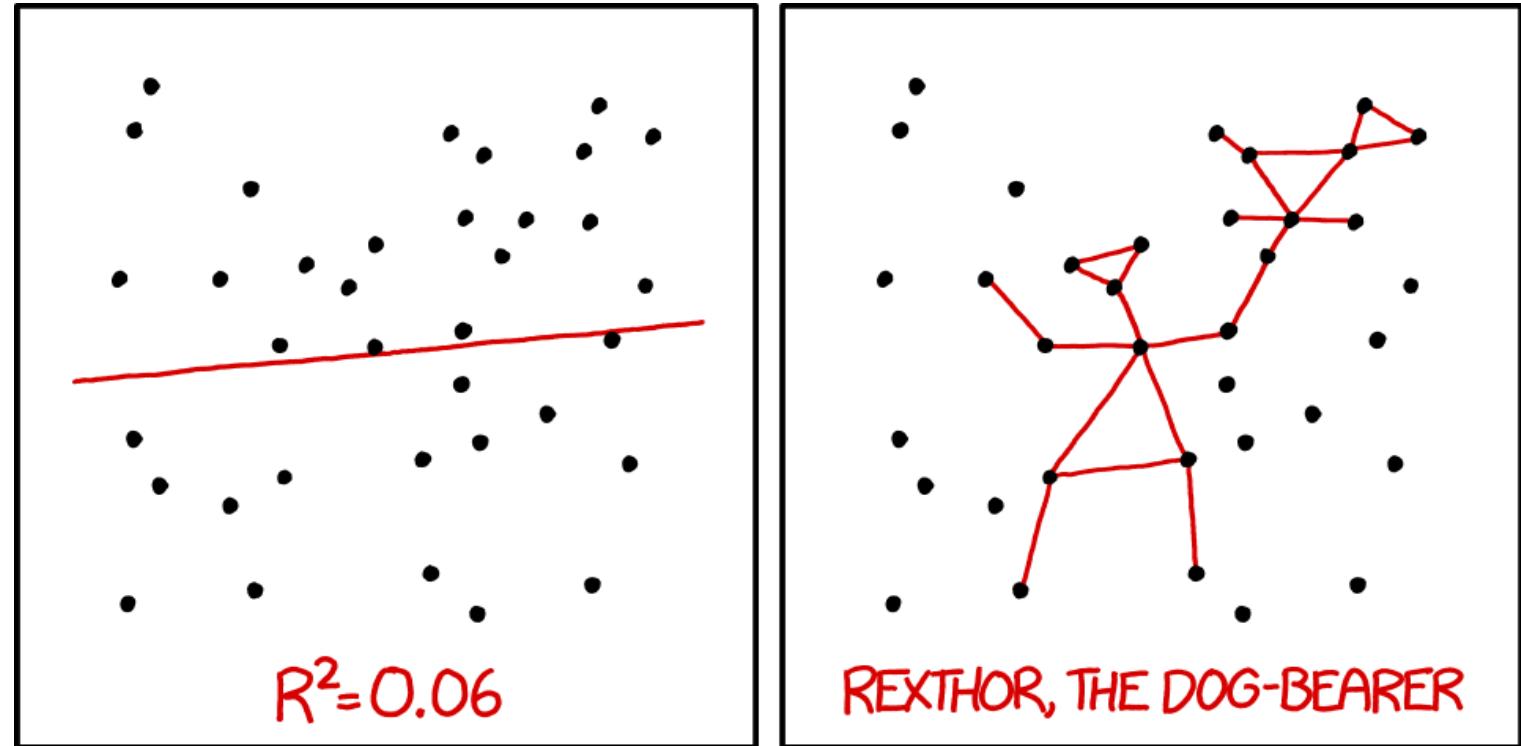
How much will Y vary from \hat{Y} ? We use **prediction intervals** to answer this question.



Confidence in predicting \hat{y}

Even if we knew $f(x)$, the response value cannot be predicted perfectly because of the random error in the model (irreducible error).

How much will Y vary from \hat{Y} ? We use **prediction intervals** to answer this question.

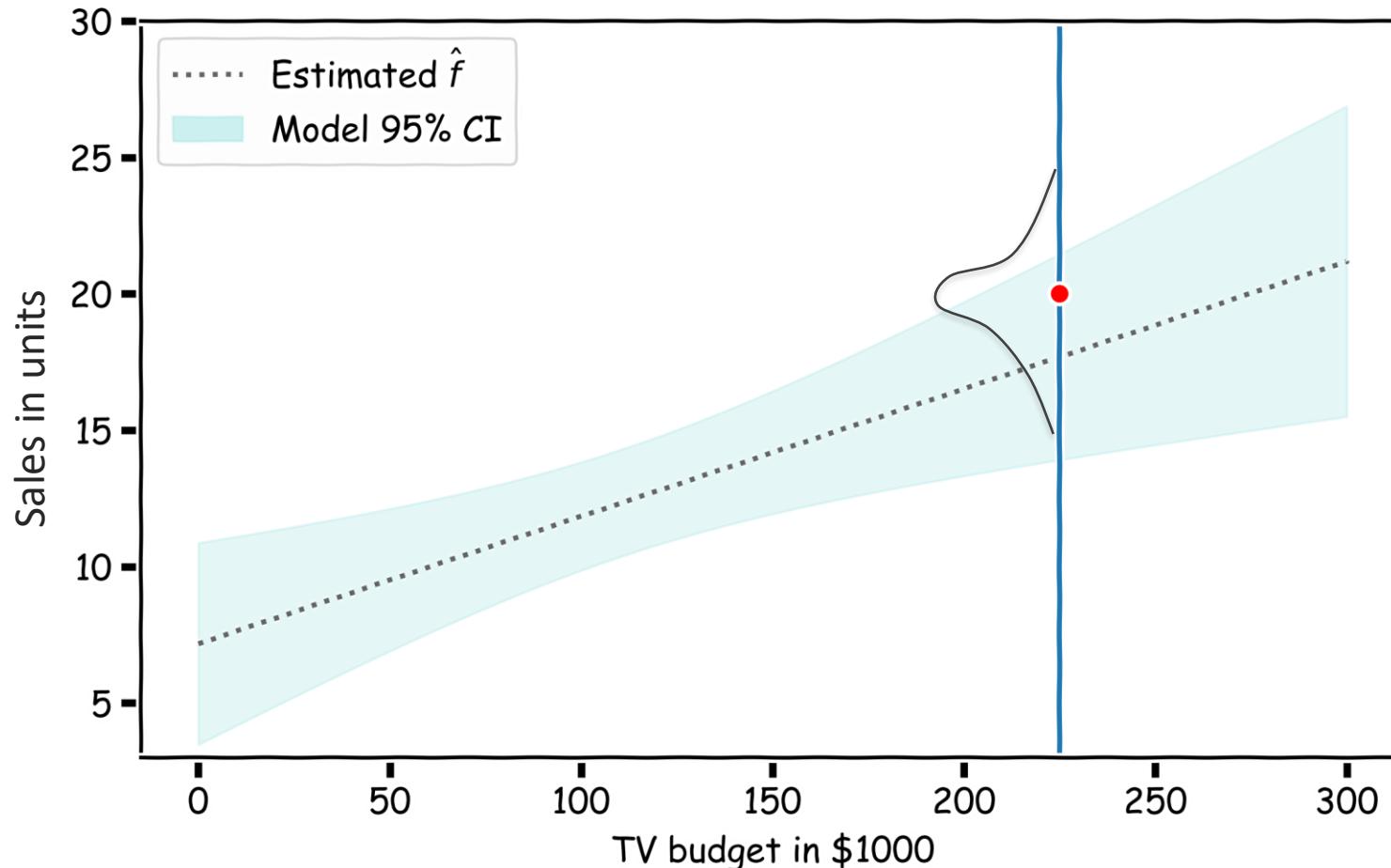


I DON'T TRUST LINEAR REGRESSIONS WHEN IT'S HARDER TO GUESS THE DIRECTION OF THE CORRELATION FROM THE SCATTER PLOT THAN TO FIND NEW CONSTELLATIONS ON IT.

Confidence in predicting \hat{y}

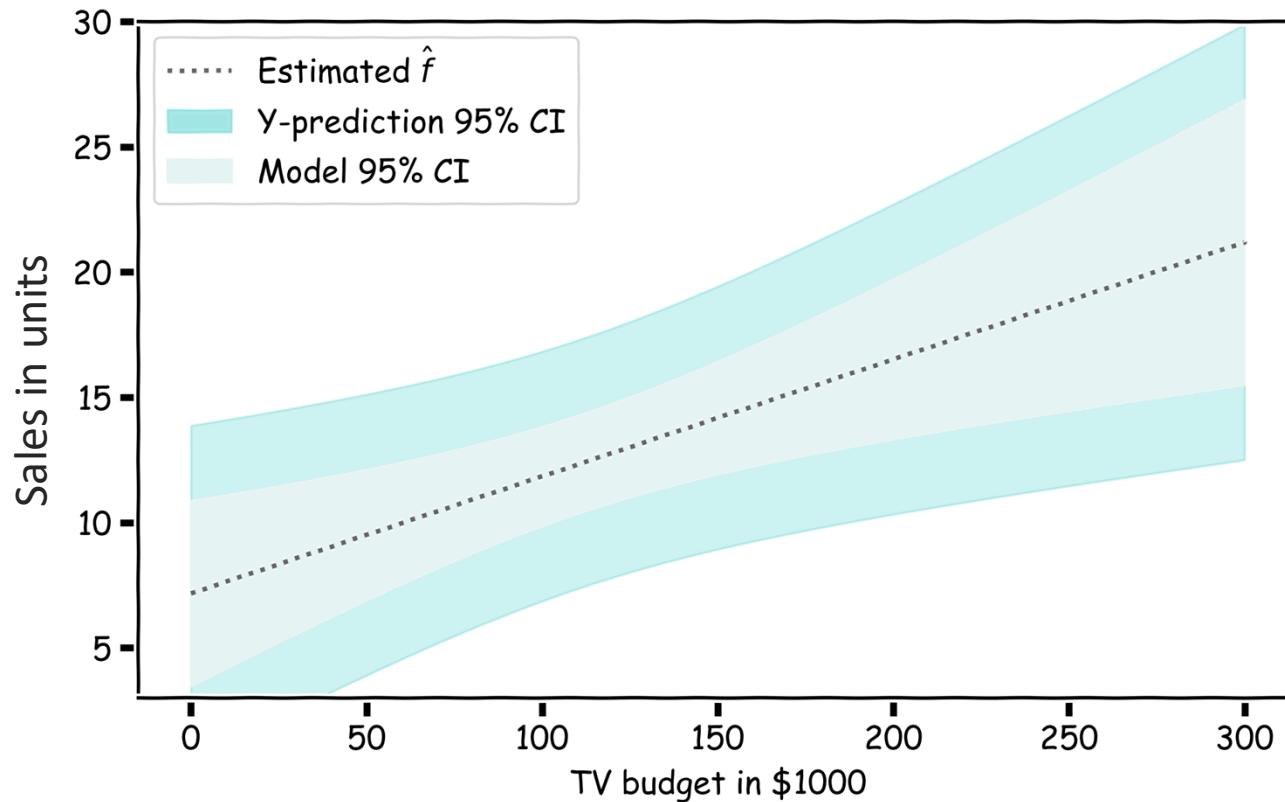
$f(x)$ is a random variable

- For a given x , we have a distribution of models $f(x)$
- For each of these $f(x)$, the prediction for $y \sim N(f(x), \sigma_\epsilon)$



Confidence in predicting \hat{y}

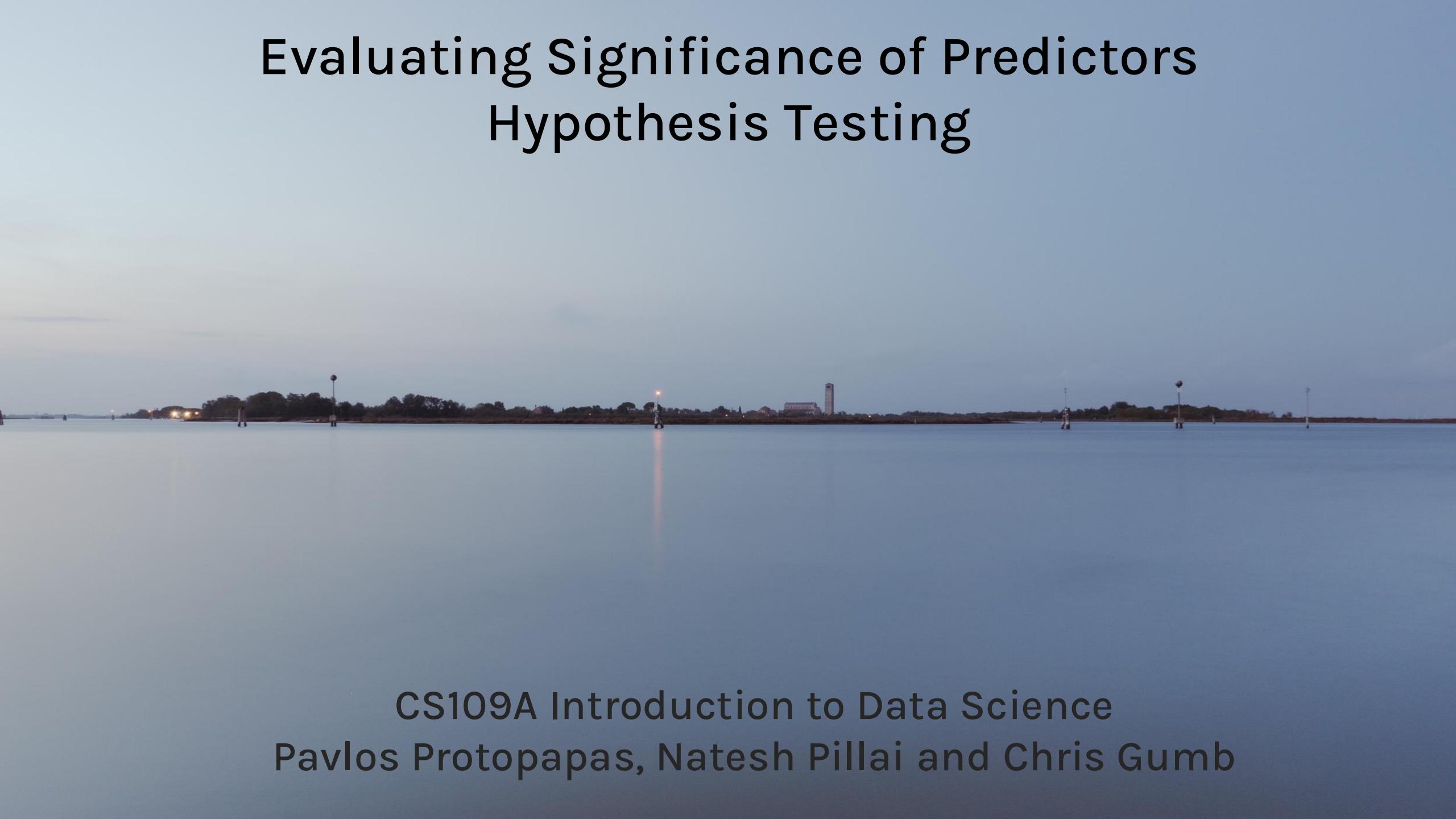
- For a given x , we have a distribution of models $f(x)$
- For each of these $f(x)$, the prediction for $y \sim N(f(x), \sigma_\epsilon)$
- The prediction confidence intervals are then ...



Thank you

Evaluating Significance of Predictors

Hypothesis Testing



CS109A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai and Chris Gumb

Outline

Part A and B: Assessing the Accuracy of the Coefficient Estimates

Bootstrapping and confidence intervals

Part C: Evaluating Significance of Predictors

Does the outcome depend on the predictors?

Hypothesis testing

Part D: How well do we know \hat{f}

The confidence intervals of \hat{f}

How Reliable are the Model Interpretations

Suppose our model for advertising is:

$$y = 1.01x + 0.005$$

where y is the sales in units (each unit is \$1000) and x is the TV budget.

Interpretation: For every dollar invested in advertising gets you 1.01 back in sales, which is a 1% net increase.

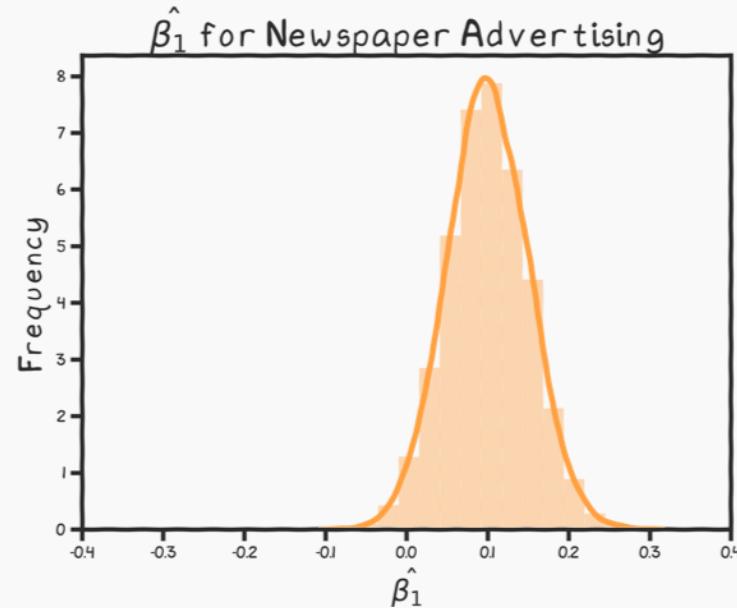
But how **certain** are we in our estimation of the coefficient 1.01?

Now you know how **certain** you are in your estimates, will you want to change your answer?

Feature Importance

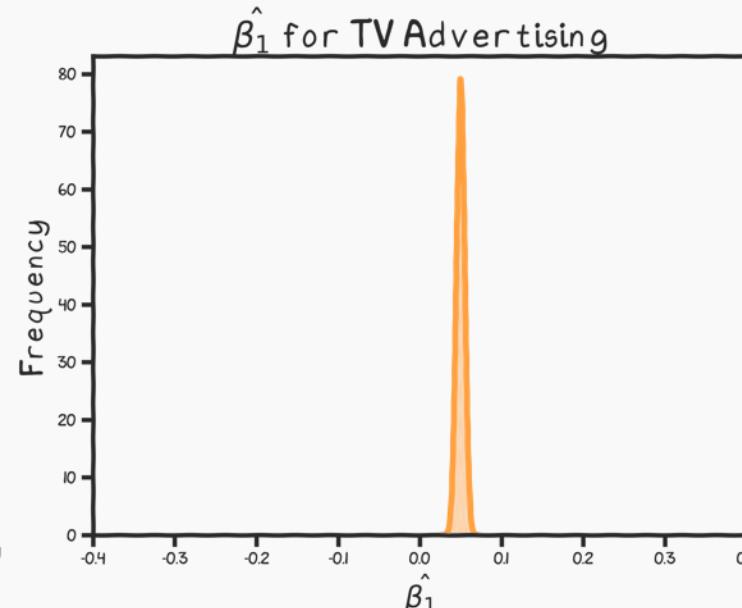
Now we know how to generate these distributions we are ready to answer ***two important questions:***

- A. Which predictors are most important?
- B. And which of them really affects the outcome?



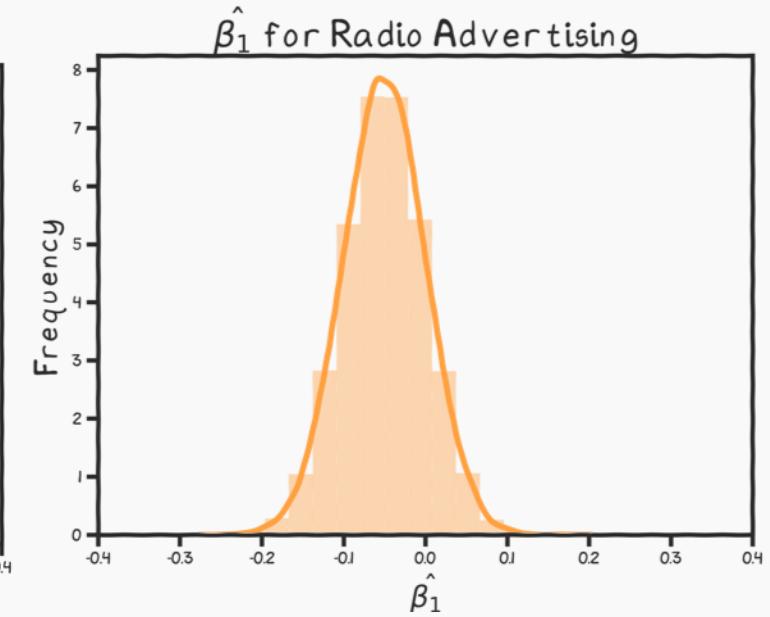
$$\mu_{\hat{\beta}_1} = 0.1$$

$$\sigma_{\hat{\beta}_1} = 0.05$$



$$\mu_{\hat{\beta}_1} = 0.05$$

$$\sigma_{\hat{\beta}_1} = 0.005$$

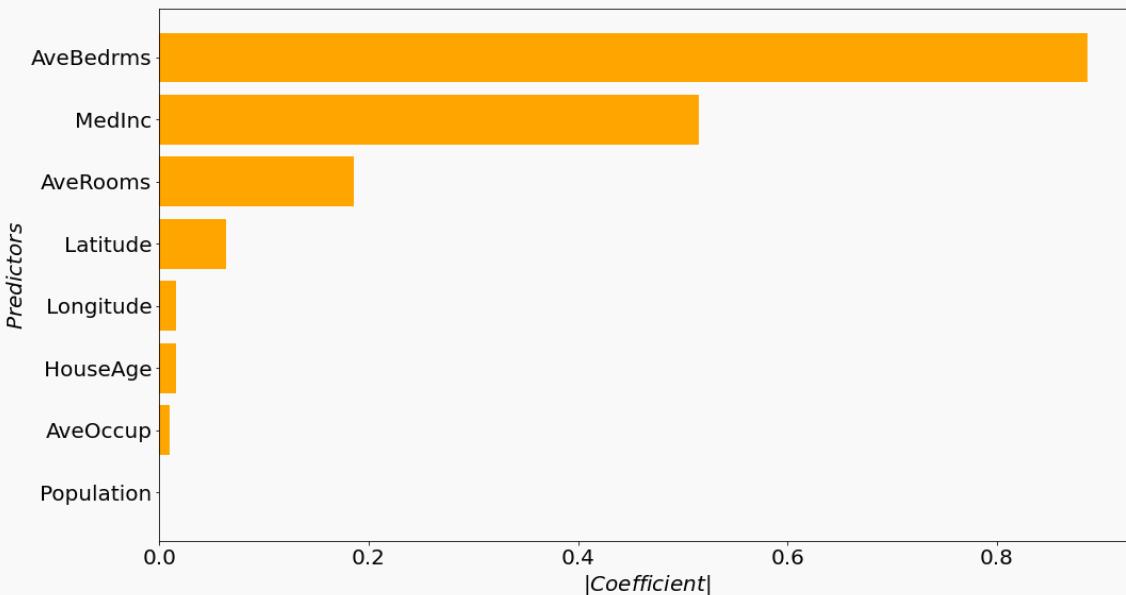


$$\mu_{\hat{\beta}_1} = -0.05$$

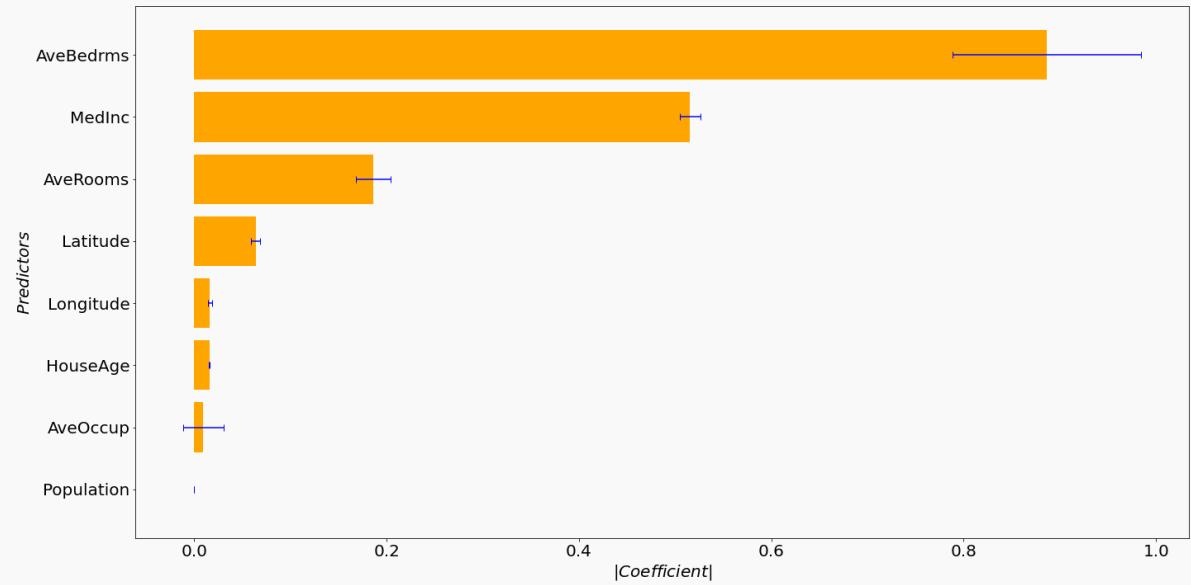
$$\sigma_{\hat{\beta}_1} = 0.1$$

Feature Importance

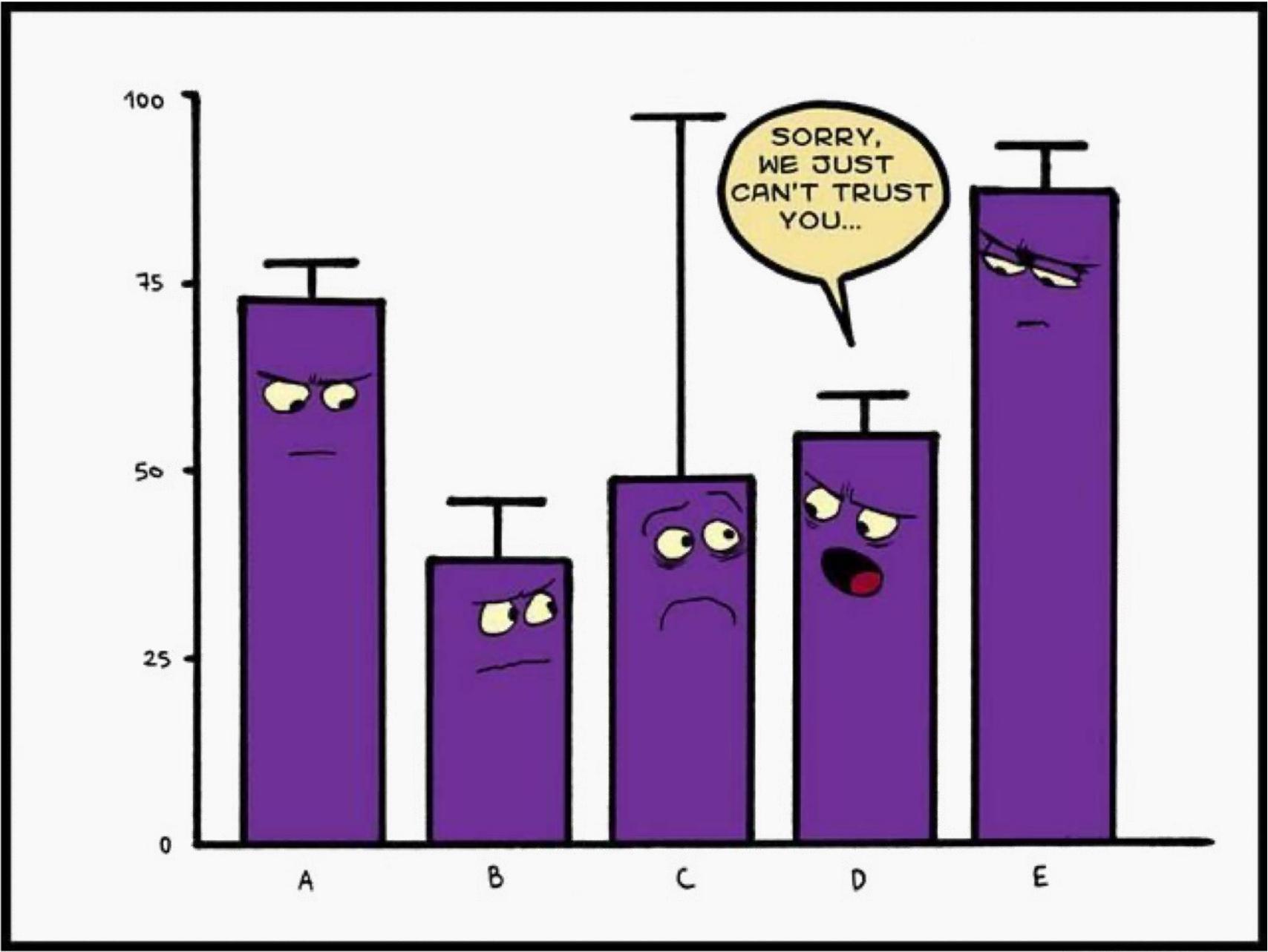
The example below is from [California housing data](#). The coefficients below are from a model that predicts prices given house size, age, crime, etc.



Feature importance based on the
absolute value of the coefficients.



Feature importance based on the absolute mean value of the coefficients over multiple bootstraps including uncertainties.

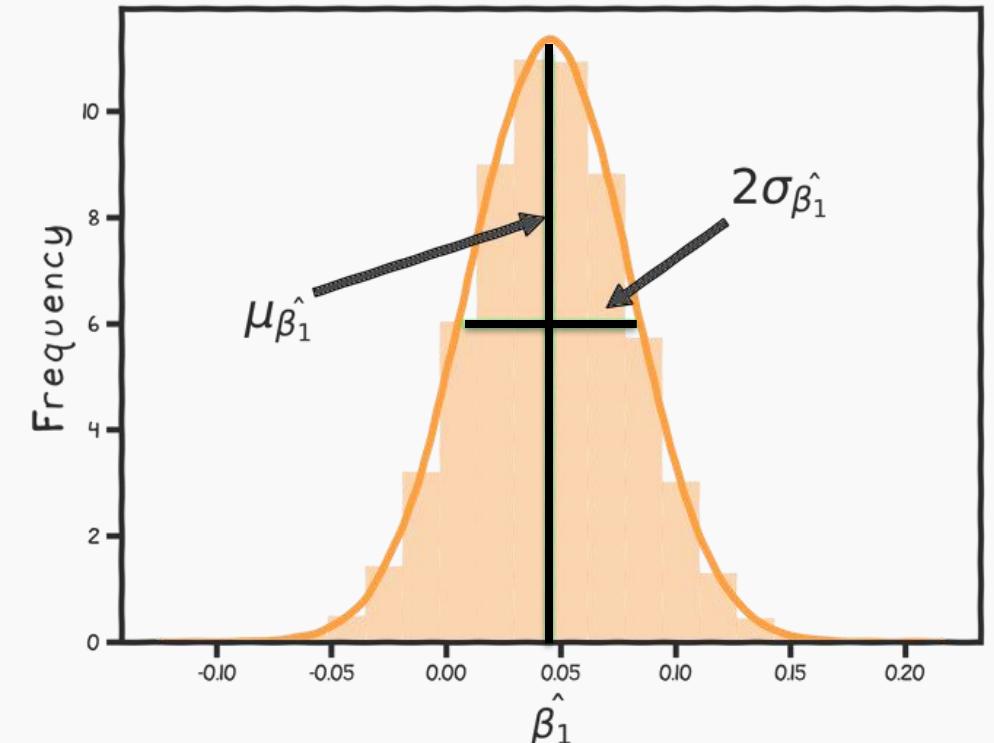


Feature Importance

To incorporate the coefficients' uncertainty, we need to determine whether the estimates of β 's are sufficiently **far from zero**.

To do so, we define a new **metric**, which we call \hat{t} – **test** statistic which measures the distance from zero in units of standard deviation:

$$\hat{t}\text{-test} = \frac{\mu_{\hat{\beta}_1}}{\sigma_{\hat{\beta}_1}}$$



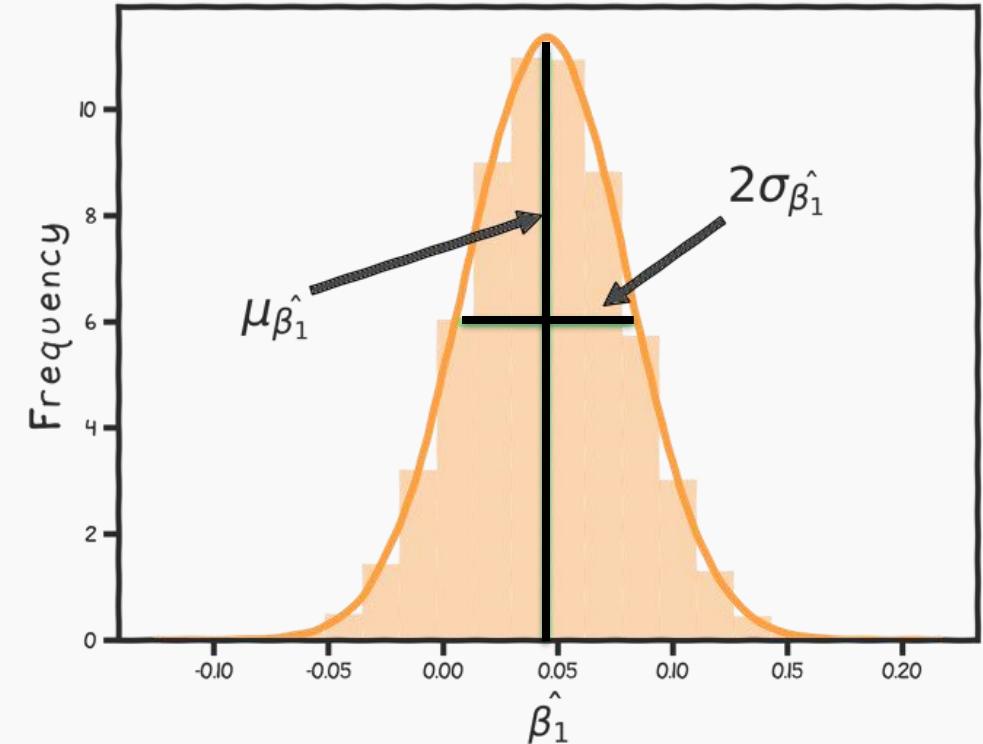
Feature Importance

$$\hat{t}\text{-test} = \frac{\mu_{\hat{\beta}_1}}{\sigma_{\hat{\beta}_1}}$$

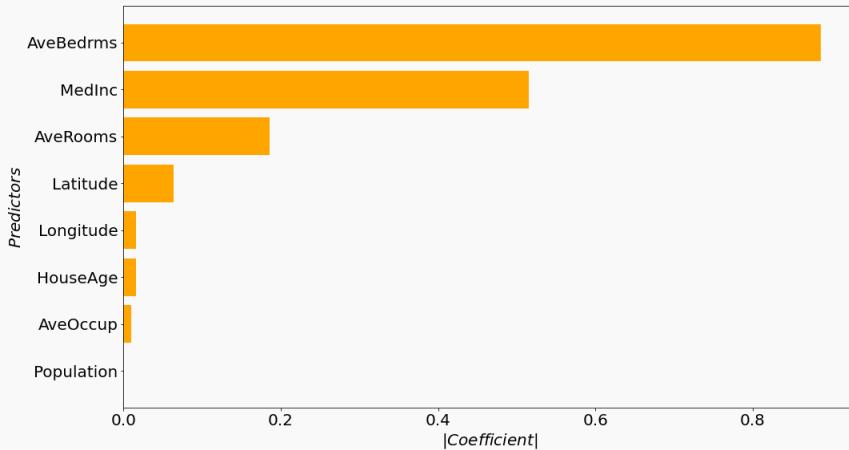
$\hat{t}\text{-test}$ is a scaled version of the usual t-test:

$$t\text{-test} = \frac{\mu_{\hat{\beta}_1}}{\sigma_{\hat{\beta}_1}/\sqrt{n}} = \sqrt{n} \hat{t}\text{-test}$$

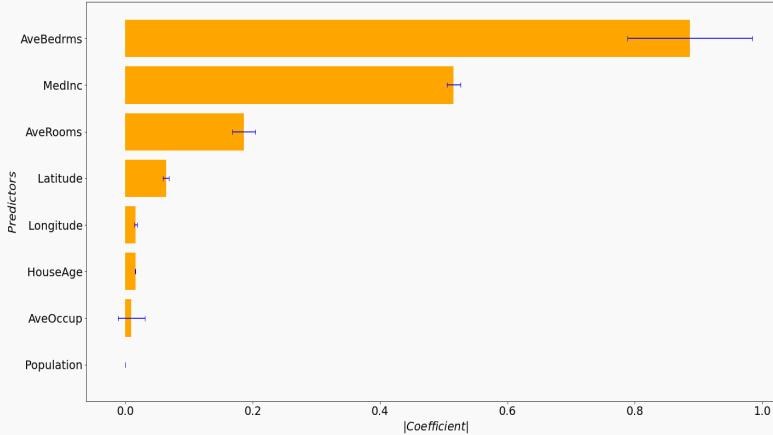
n is the number of bootstraps.



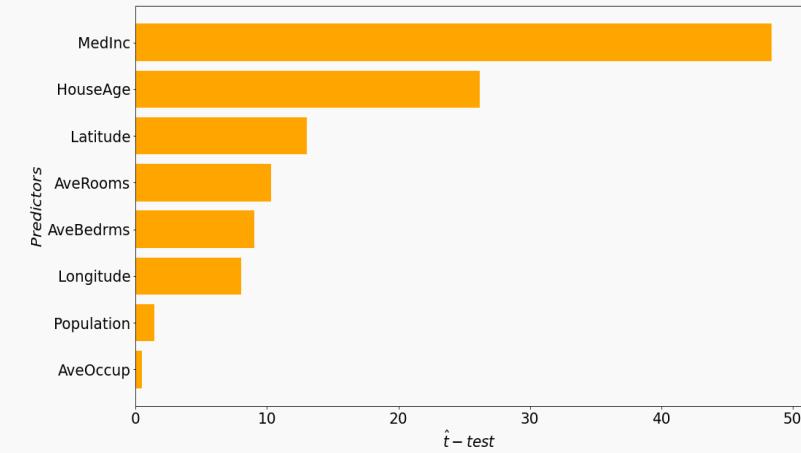
Feature Importance



The **absolute value** of the coefficients.



The absolute value of the coefficients over multiple **bootstraps** and includes the **uncertainty** of the coefficients.



The \hat{t} -test. Notice the rank of the importance has changed.

Feature Importance

Because a predictor is ranked as the most important, it does not necessarily mean that the **outcome depends on that predictor.**

How do we assess if there is a true relationship between outcome and predictors?

As with R^2 score, we should compare its significance ($\hat{t} - test$) to:



Feature Importance

What do we
mean by random
data?

We want to compare the $\hat{t} - test$ of the predictors from our model with $\hat{t} - test$ values calculated using random data.

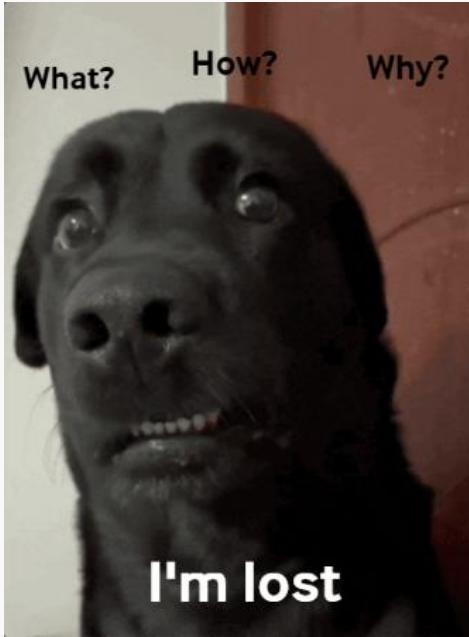
Feature Importance

We want to compare the $\hat{t} - test$ of the predictors from our model with $\hat{t} - test$ values calculated using random data.

```
np.random.norm(0,1, n)
```

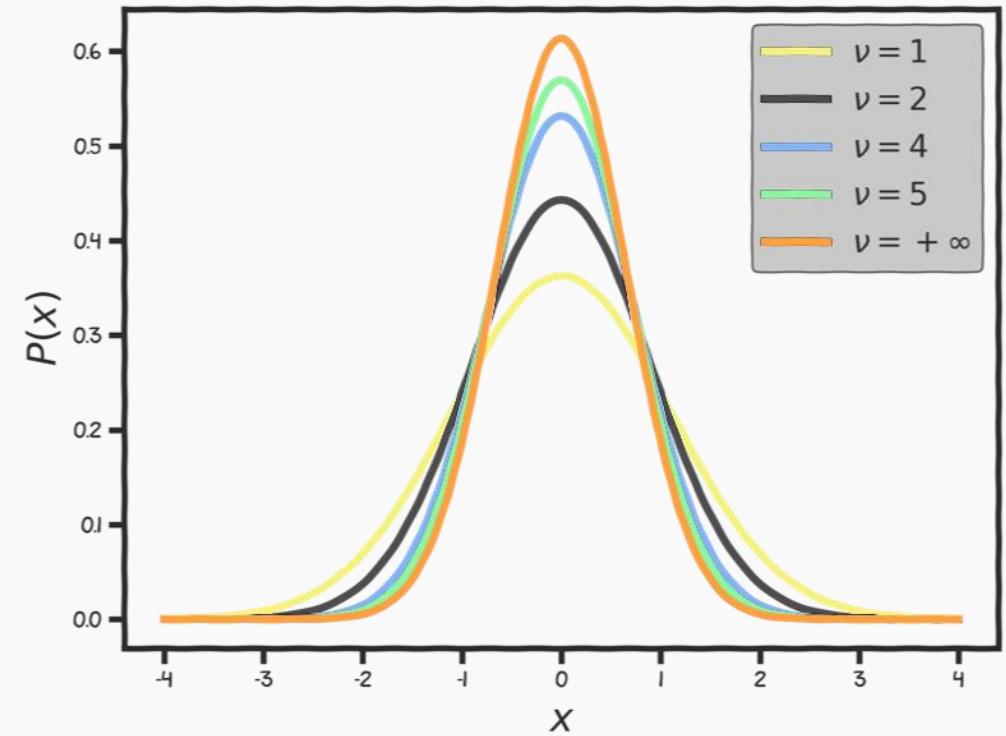
Feature Importance

1. For m random datasets fit m models.
2. Generate distributions for all predictors and calculate the means and standard errors ($\mu_{\hat{\beta}}, \widehat{SE}_{\hat{\beta}_1}$).
3. Calculate the $\hat{t}_{test_random} = \frac{\mu_{\hat{\beta}_1}}{\widehat{SE}_{\hat{\beta}_1}}$.
4. Repeat steps 1-3 and create a histogram for all the $\hat{t} - test - random$.



Feature Importance

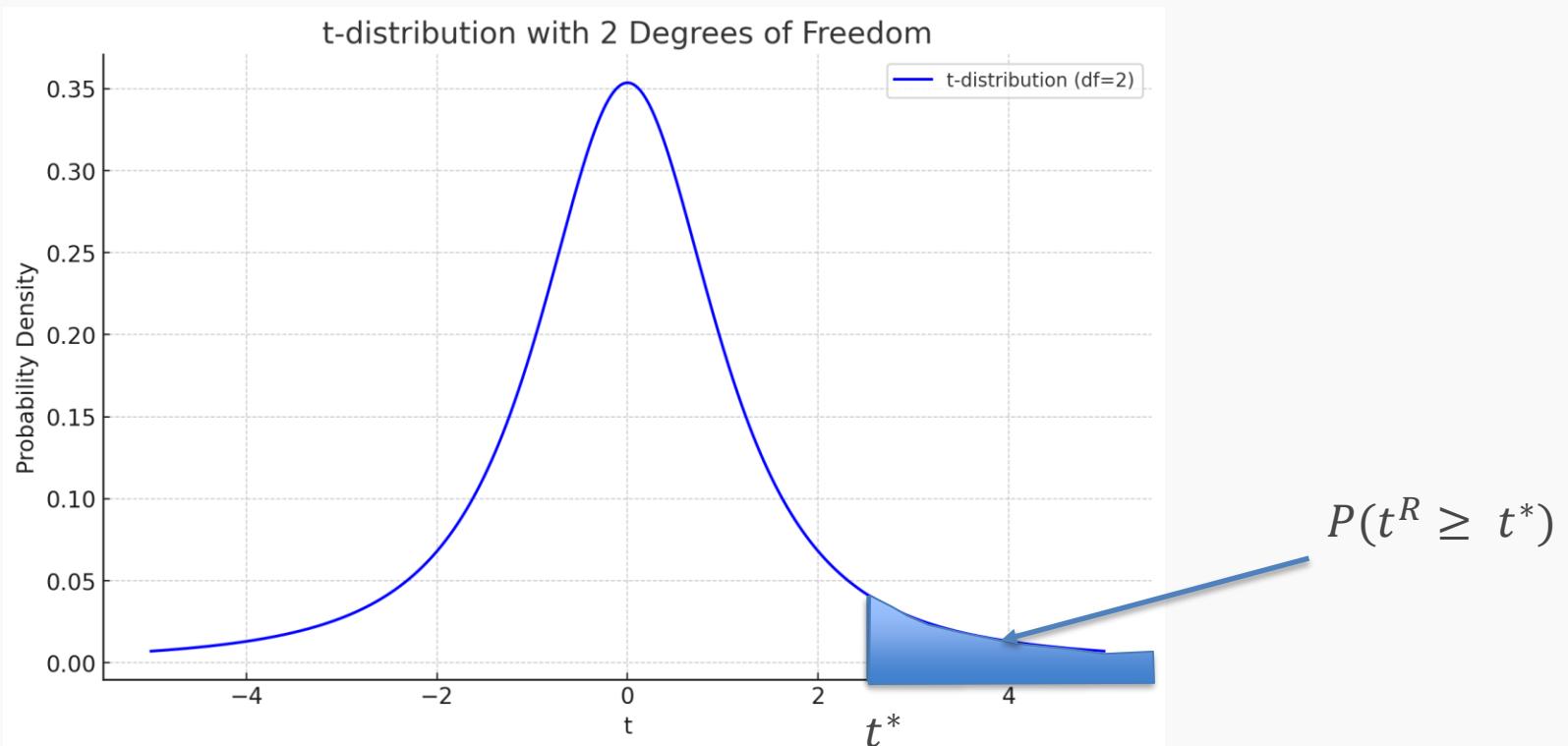
It turns out we do not have to do this, because this is a known distribution called student-t distribution.



Student-t distribution, where v is the degrees of freedom (number of data points minus number of predictors) = $n - (p + 1)$.

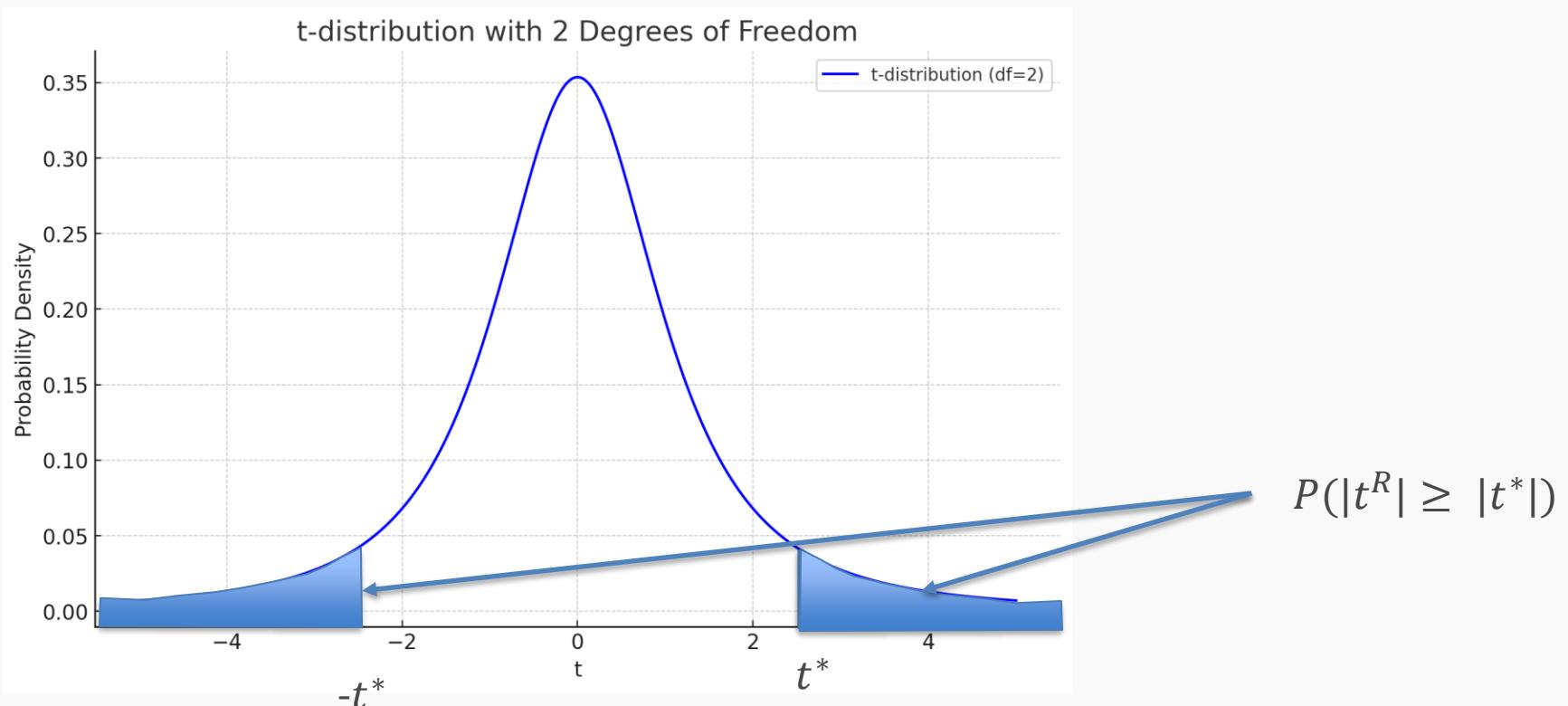
P-value

To compare the t -test values of the predictors from our model, t^* , with the t -tests calculated using random data, t^R , we estimate the probability of observing $t^R \geq t^*$.



P-value

Actually, we need compare $|t^*|$, with the t-tests calculated using random data, $|t^R|$, we estimate the probability of observing $|t^R| \geq |t^*|$.



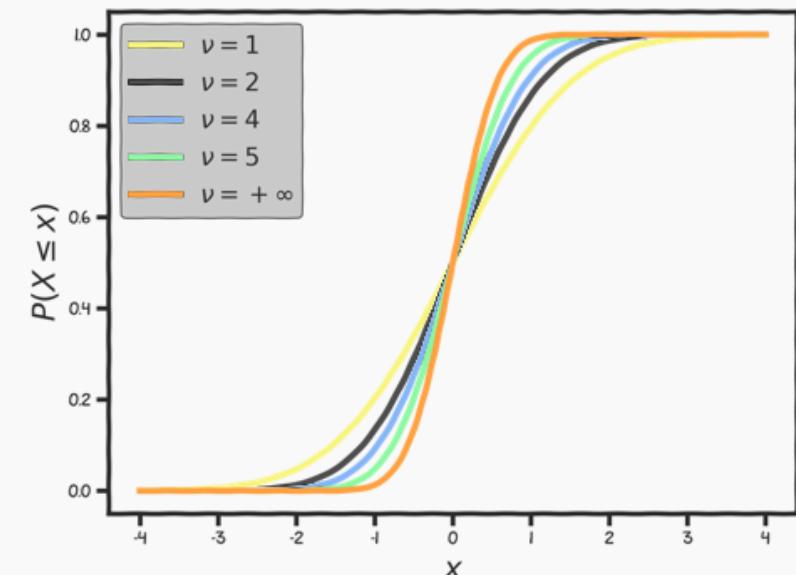
P-value

We call this probability the p-value:

$$p\text{-value} = P(|t^R| \geq |t^*|)$$

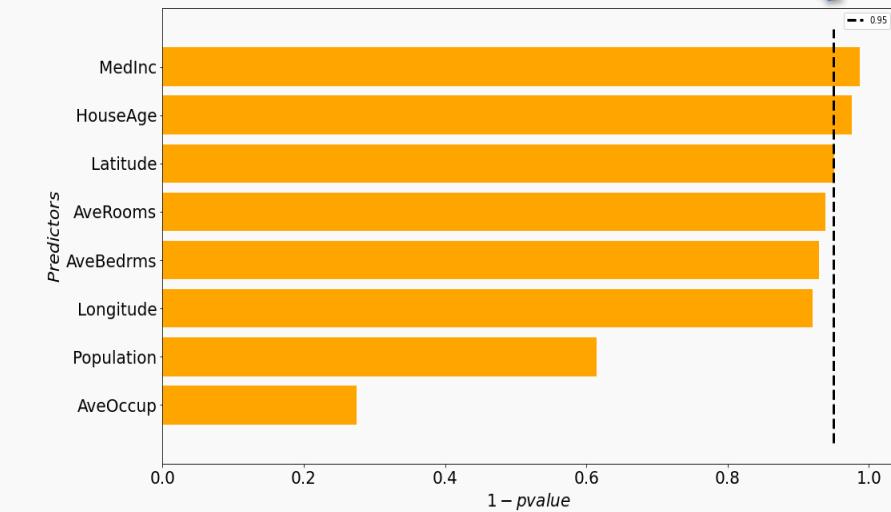
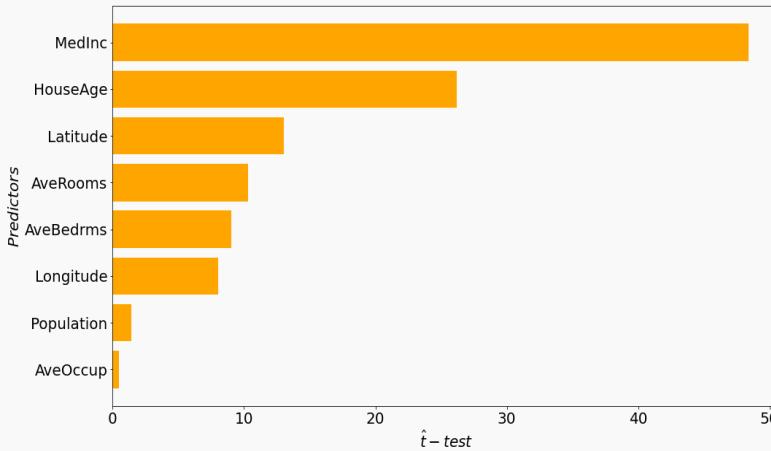
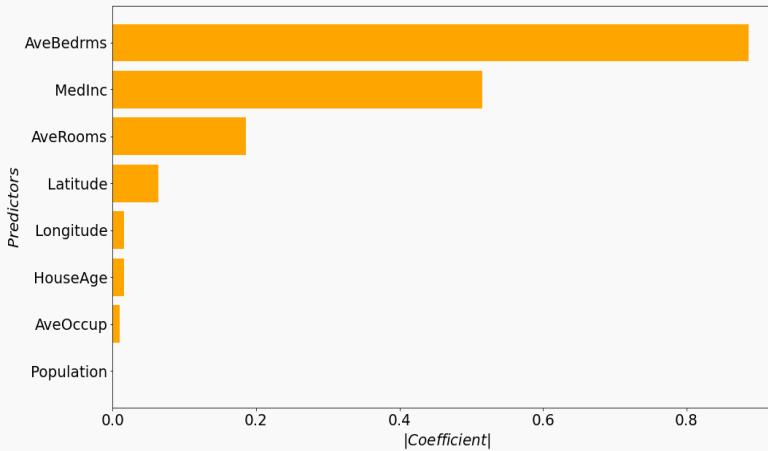
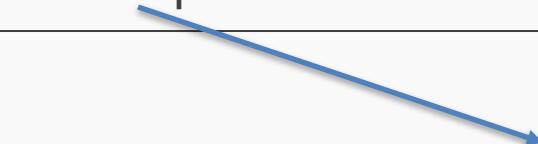
Small p-value indicates that it is unlikely to observe such a substantial association between the predictor and the response due to chance. It is common to use p-value<0.05 as the threshold for significance.

To calculate the p-value we use the cumulative distribution function (CDF) of the student-t. stats model a python library has a build-in function stats.t.cdf () which can be used to calculate this.



Feature Importance

Any predictor with a $1 - p$ value higher than this is considered important.



The absolute value of the coefficients over multiple **bootstraps** and includes the coefficients' uncertainty.

The \hat{t} -test. Notice the rank of the importance has changed.

Using the the **p-value** we also have which predictors are important. Note here we use $1-p$.

Hypothesis Testing

Hypothesis testing is a formal process through which we evaluate the validity of a statistical hypothesis by considering evidence **for** or **against** the hypothesis gathered by **random sampling** of the data.

1. State the hypotheses, typically a **null hypothesis**, H_0 and an **alternative hypothesis**, H_1 , that is the negation of the former.
2. Choose a type of analysis, i.e. how to use sample data to evaluate the null hypothesis. Typically, this involves choosing a single test statistic.
3. **Sample** data and compute the test statistic.
4. Use the value of the test statistic to either **reject** or **not reject** the null hypothesis.

Hypothesis Testing

1. State Hypothesis:

Null hypothesis:

H_0 : There is no relation between X_j and Y , ($\beta_j = 0$).

The alternative:

H_a : There is some relation between X_j and Y , ($\beta_j \neq 0$).

2. Choose test statistics

$$\hat{t} - test = \frac{\mu_{\hat{\beta}_1}}{\widehat{SE}_{\hat{\beta}_1}}$$

Hypothesis Testing

3. Sample:

Using bootstrap we can estimate $\hat{\beta}_1$'s, and $\mu_{\hat{\beta}_1}$ and $\widehat{SE}_{\hat{\beta}_1}$ and the $\hat{t} - test$.

4. Reject or not reject the hypothesis:

We compute **p-value**, the probability of observing any value equal to $|t|$ or larger, from random data.

If p-value < p-value-threshold we **reject the null**.

Not Done Yet



Permutation Tests: a side note

Should you use a bootstrap approach to perform a hypothesis test?

While this is tempting, this is **not advisable**. Why?

It is a technical issue: the bootstrap approach is prone to inflating Type I error: you conclude there is an association when there really is not one.

In order to preserve the stated Type I error (presumably at 5%), you should instead perform a permutation test: another resampling method.

In a permutation test, you resample the data assuming the null hypothesis is true. This can most easily be done by shuffling the response variable while keeping the columns of the predictors as-is.

Bootstrapping and Confidence Intervals



CS109A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai and Chris Gumbel

Outline

Part A and B: Assessing the Accuracy of the Coefficient Estimates

Bootstrapping and confidence intervals

Part C: Evaluating Significance of Predictors

Does the outcome depend on the predictors?

Hypothesis testing

Part D: How well do we know \hat{f}

The confidence intervals of \hat{f}

Lack of Active Imagination

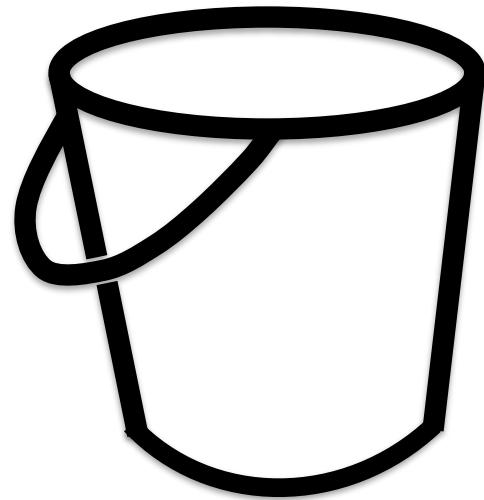
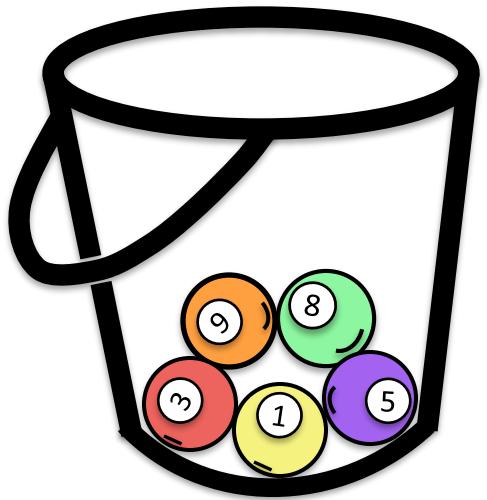
In the lack of active imagination, parallel universes and the likes, we need an alternative way of producing fake data set that resemble the parallel universes.

Bootstrapping is the practice of **sampling** from the observed data (X, Y) in estimating statistical properties.

*Note: this is not to create synthetic data to add to the actual observed data. It is to mimic the alternative universes mentioned previously.

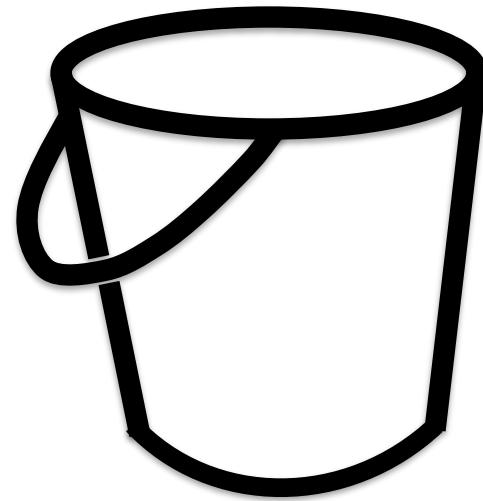
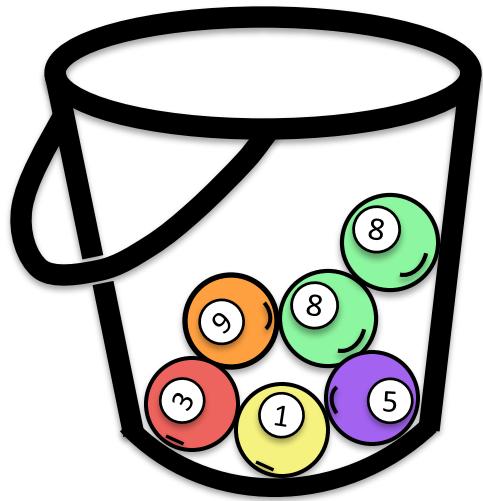
Bootstrap

Imagine we have 5 billiard balls in a bucket.



Bootstrap

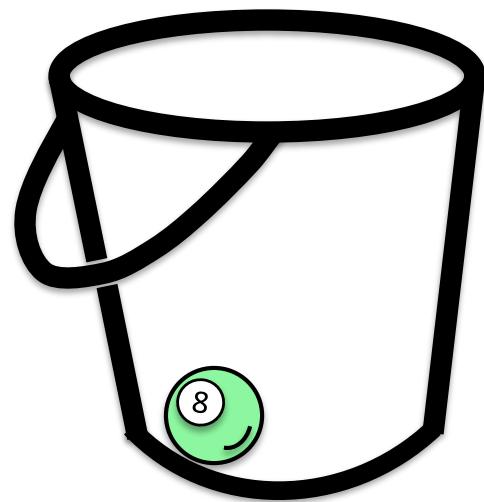
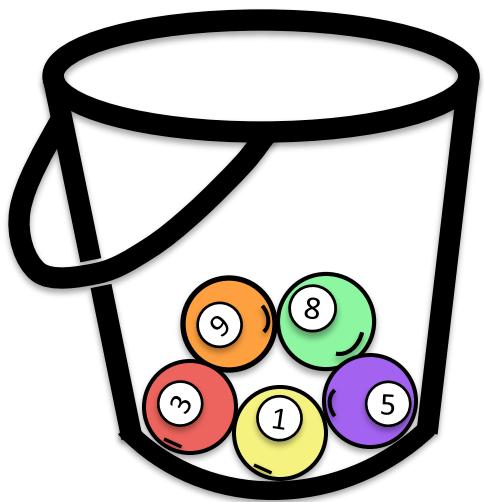
We first pick **randomly** a ball and **replicate it**.



This is called **sampling with replacement**.

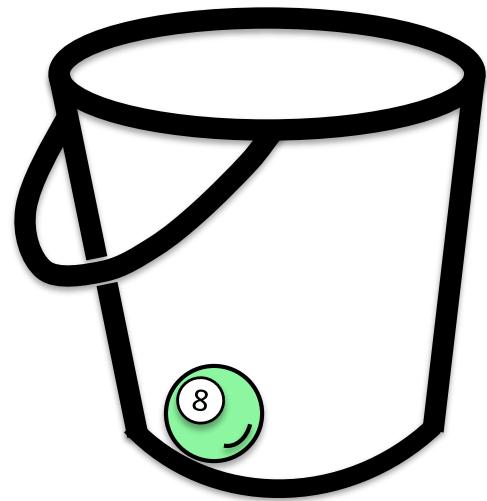
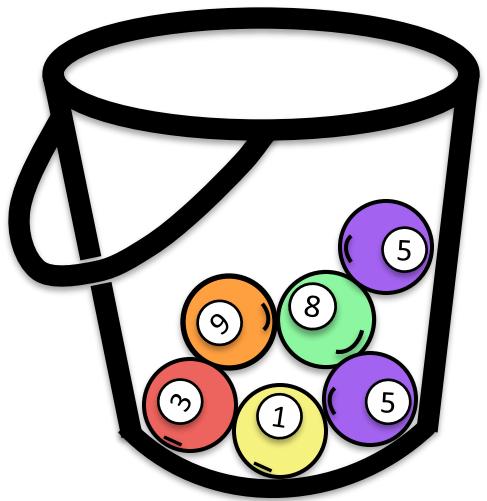
Bootstrap

We move the replicated ball to another bucket.



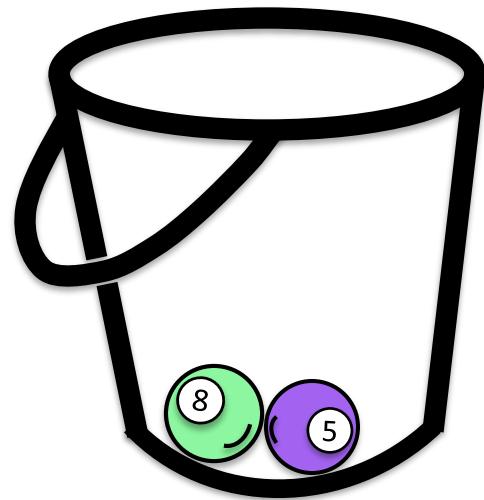
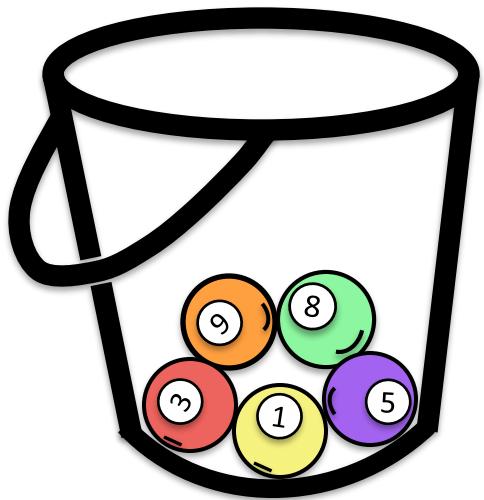
Bootstrap

We then randomly pick another ball and again we replicate it.



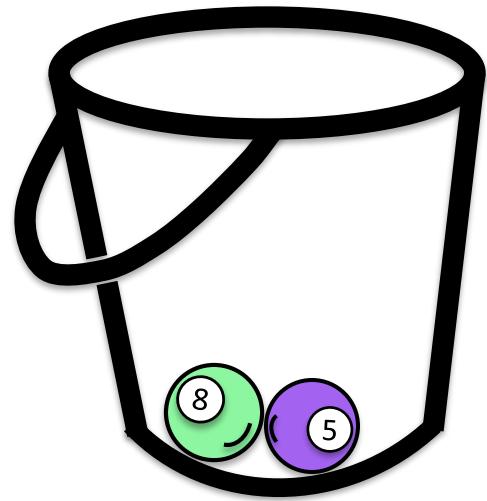
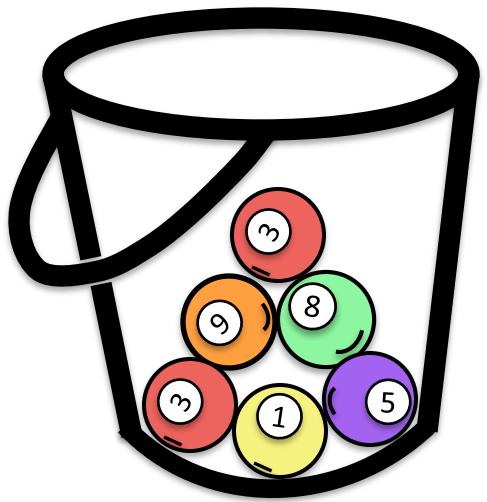
Bootstrap

As before, we move the replicated ball to the other bucket.



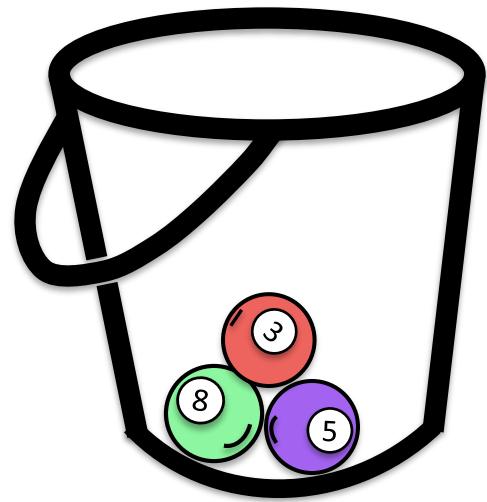
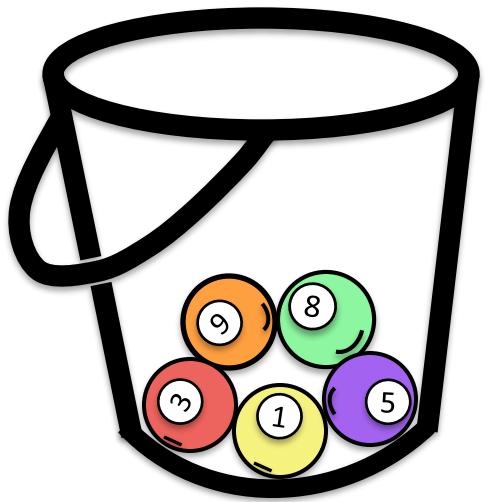
Bootstrap

We repeat this process.



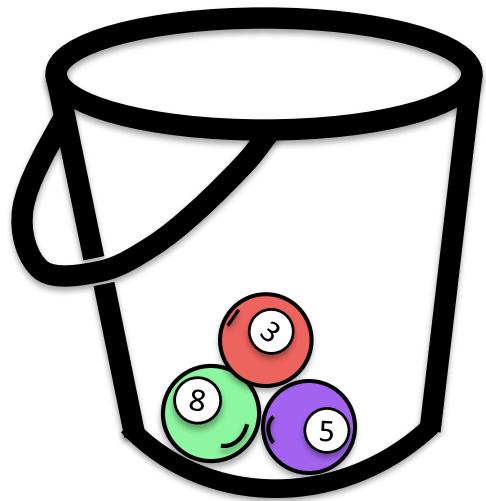
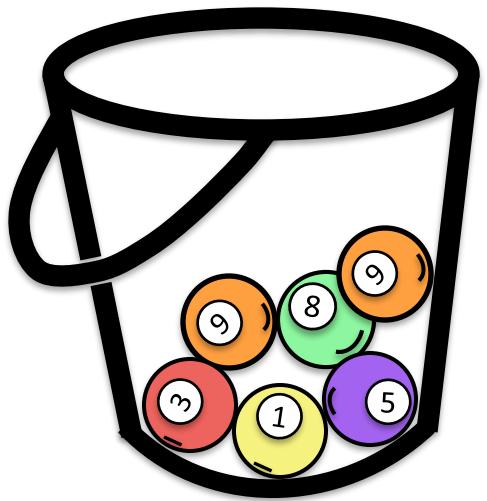
Bootstrap

We repeat this process.



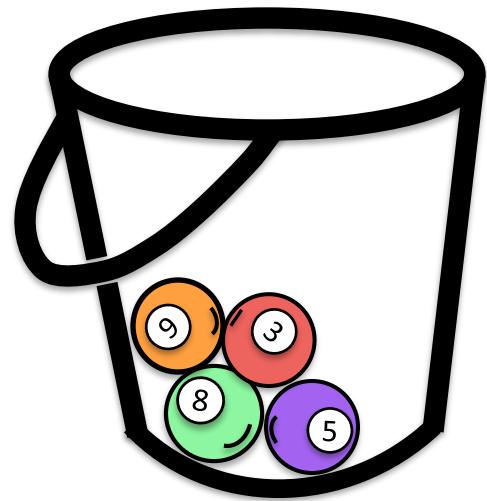
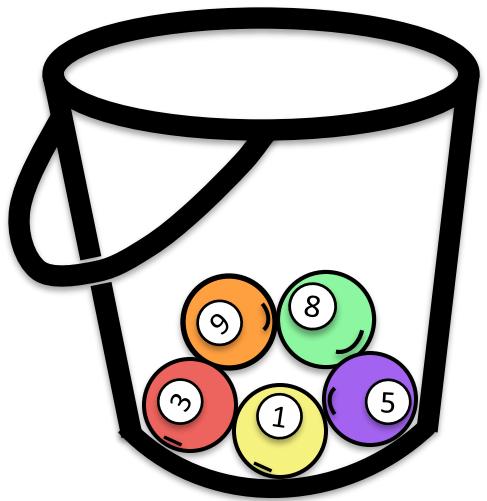
Bootstrap

We repeat this process.



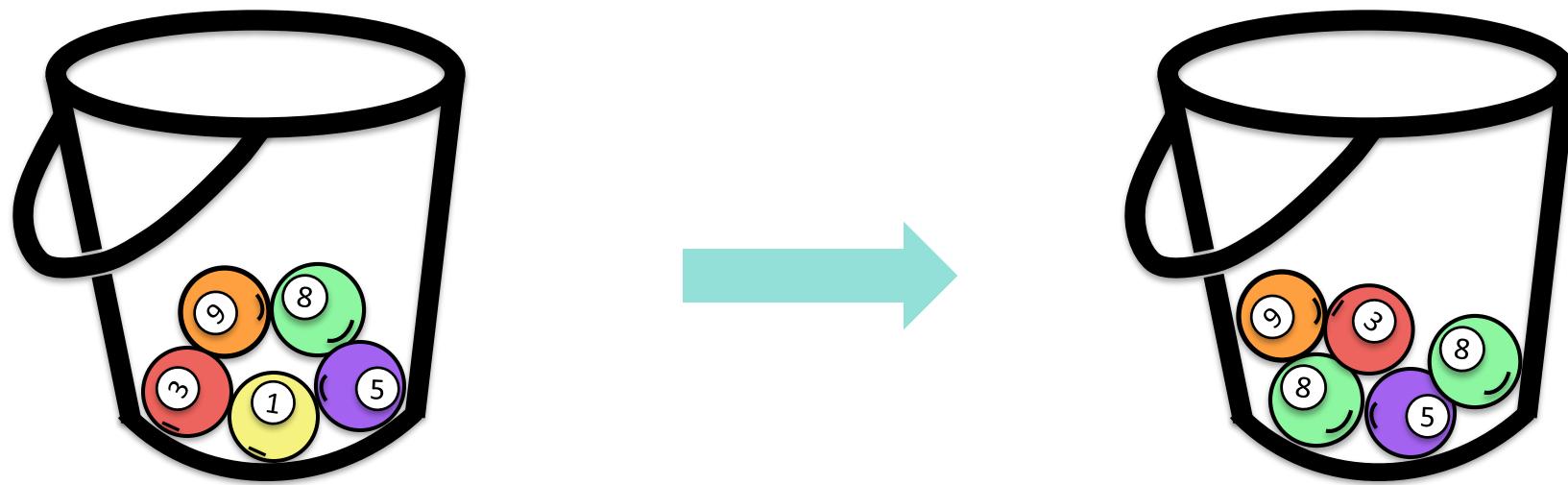
Bootstrap

We repeat this process.



Bootstrap

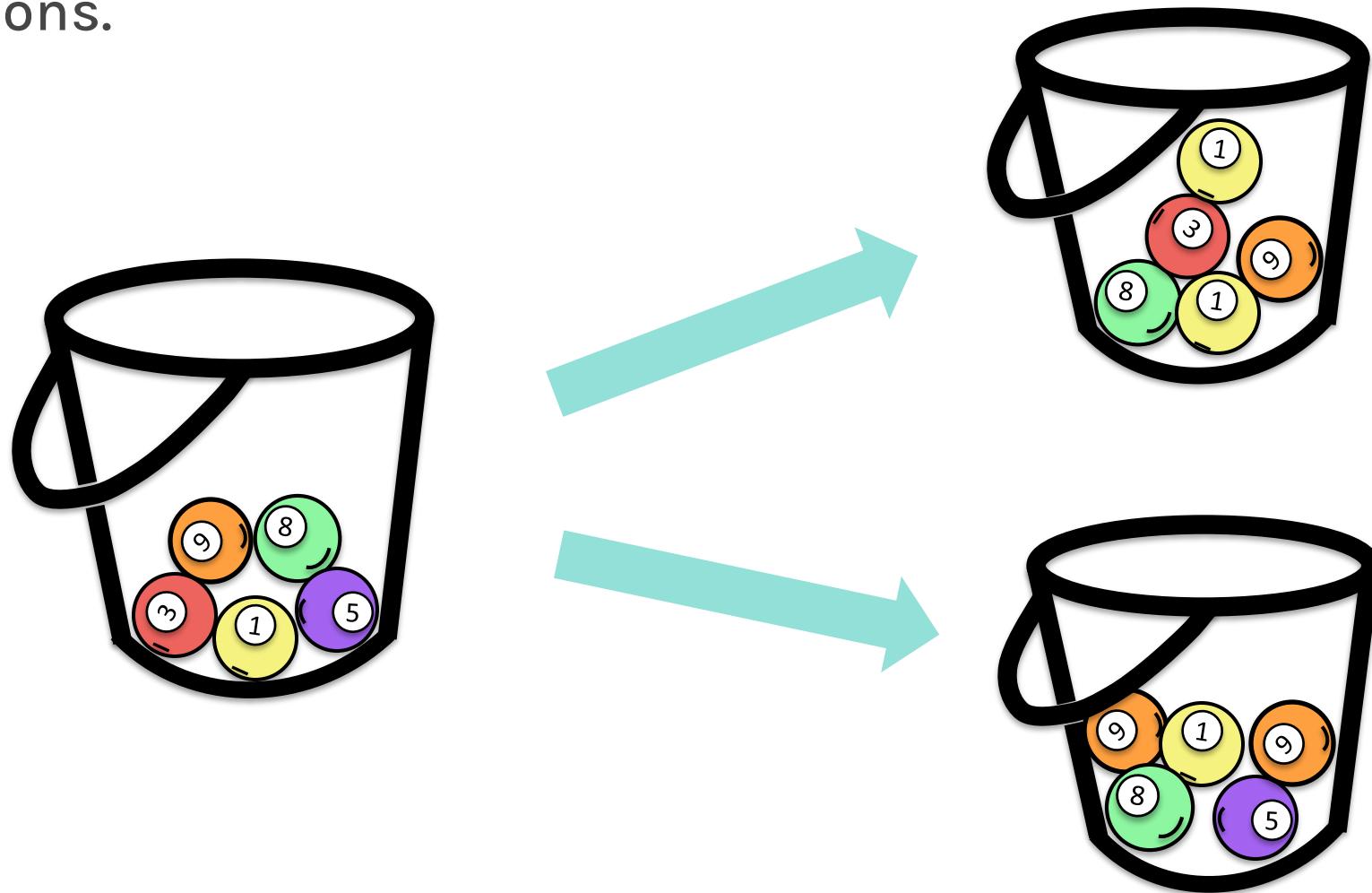
We continue until the “other” bucket has **the same number of balls** as the original one.



This new bucket represents a new parallel universe

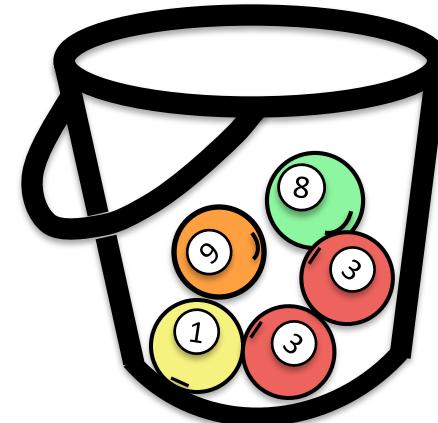
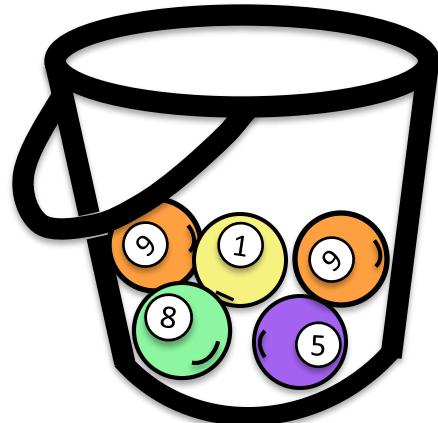
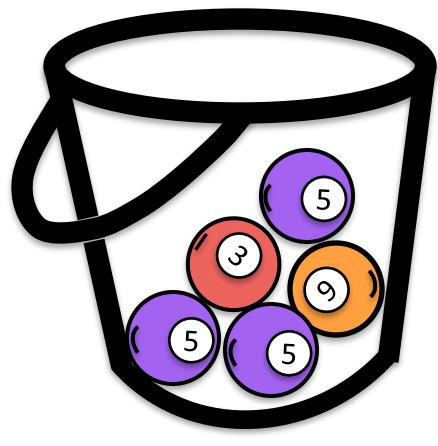
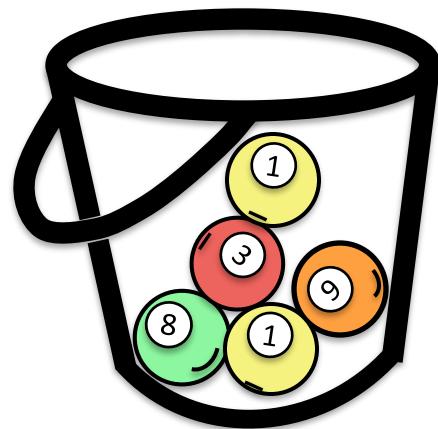
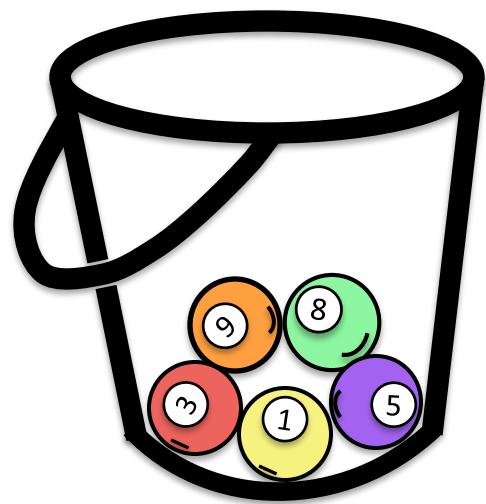
Bootstrap

We repeat the same process and acquire another set of bootstrapped observations.



Bootstrap

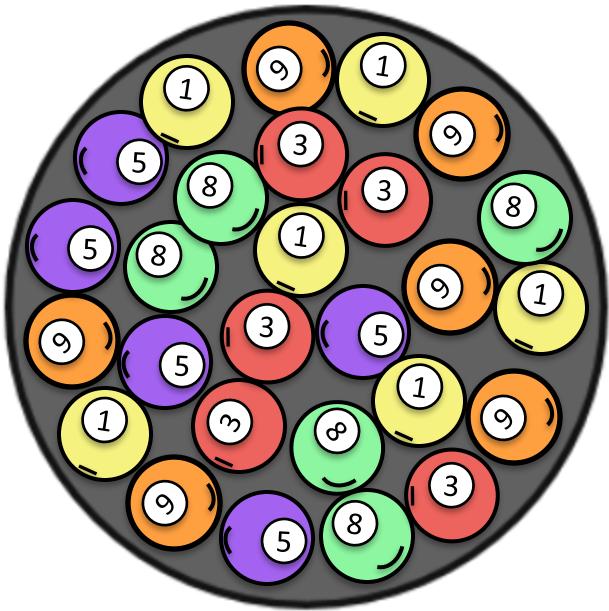
We repeat the same process and acquire another set of bootstrapped observations.



Bootstrap

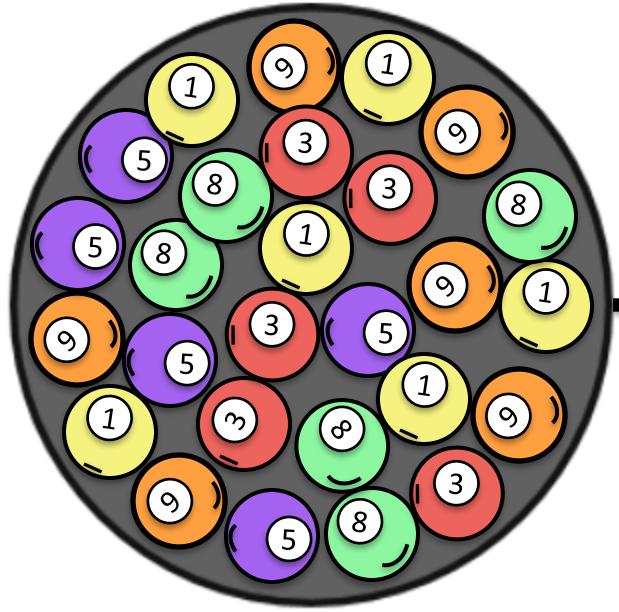
DATASET

Size N

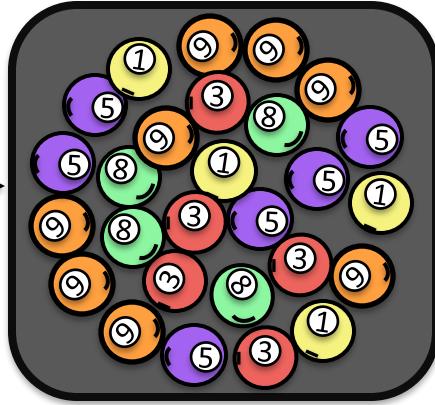


Bootstrap

DATASET
Size N



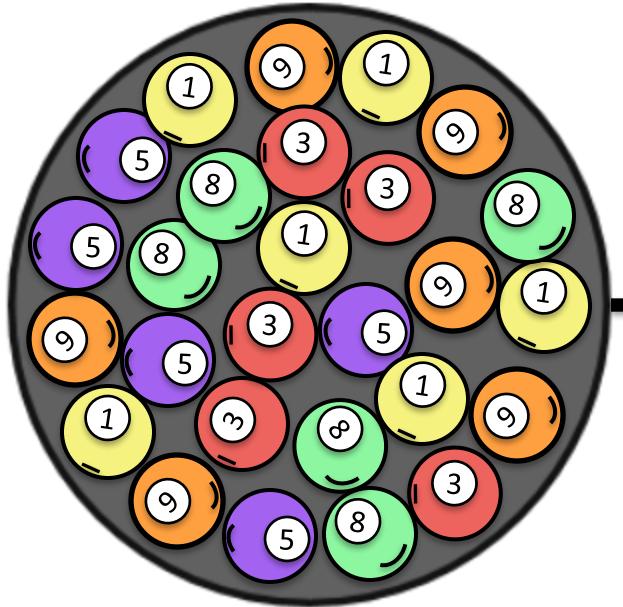
Sample with replacement



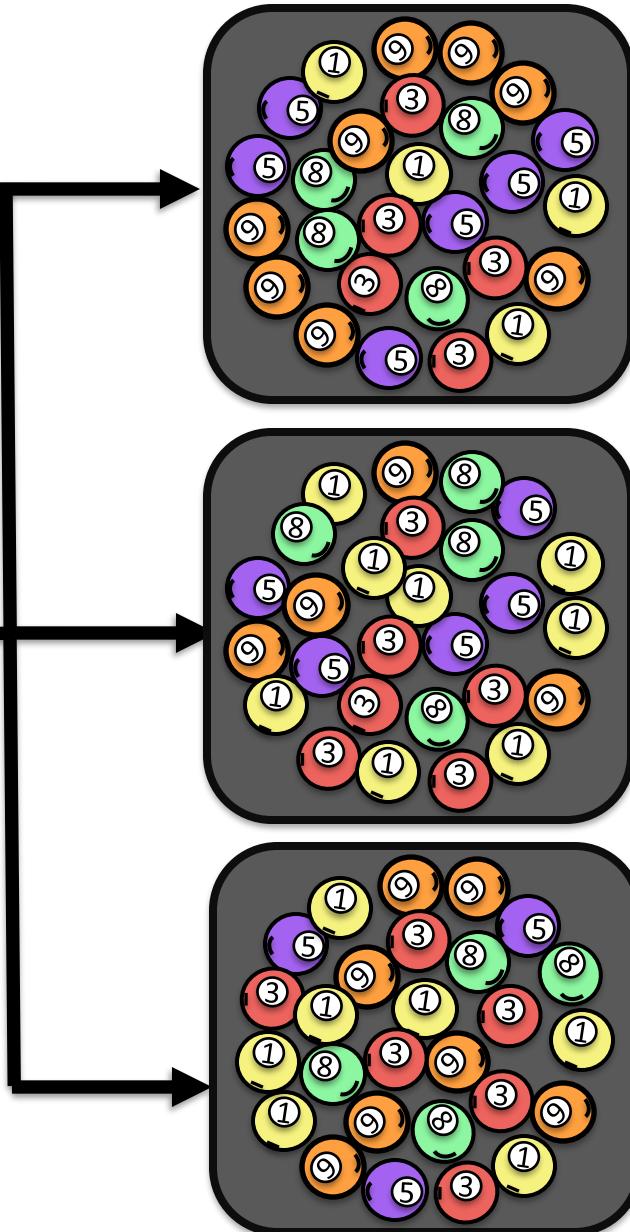
Sample 1
Size N

Bootstrap

DATASET
Size N



Sample with replacement



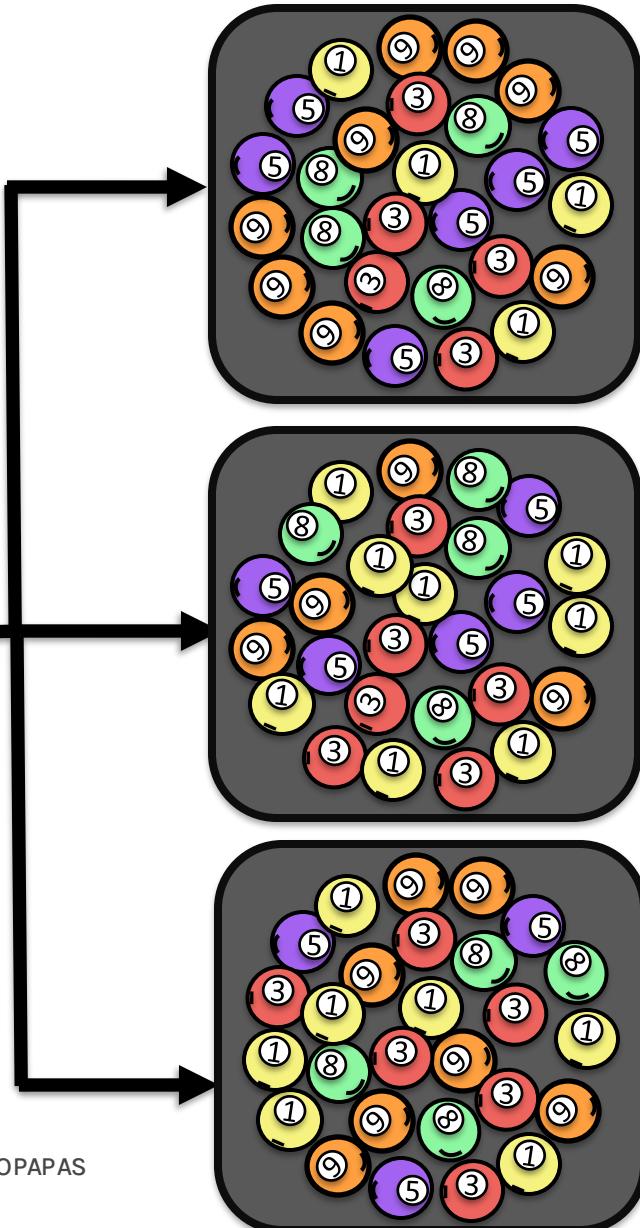
Sample 1
Size N

Sample 2
Size N

Sample 3
Size N

Bootstrap

Sample with replacement



Sample 1

Size N

Train

$$\text{Model 1: } \hat{y} = \hat{\beta}_0^{(1)} + \hat{\beta}_1^{(1)}x$$

Sample 2

Size N

Train

$$\text{Model 2: } \hat{y} = \hat{\beta}_0^{(2)} + \hat{\beta}_1^{(2)}x$$

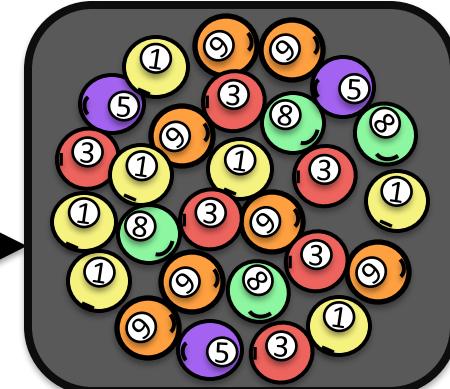
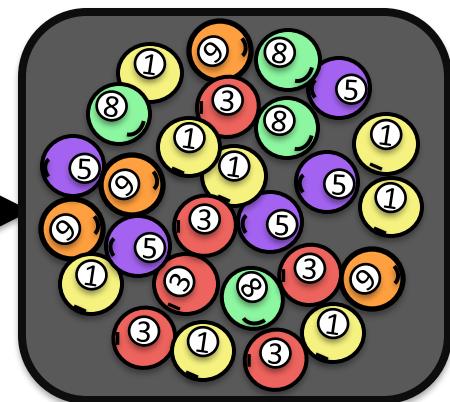
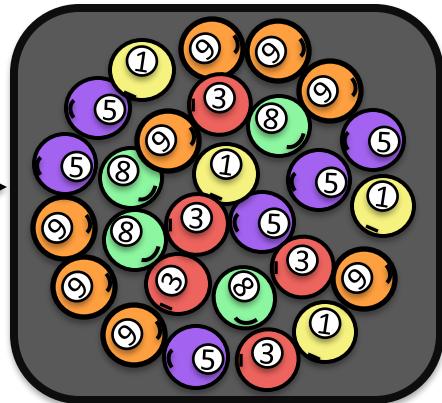
Sample 3

Size N

Train

$$\text{Model s: } \hat{y} = \hat{\beta}_0^{(s)} + \hat{\beta}_1^{(s)}x$$

Bootstrap



Sample 1
Size N

Train →

$$\text{Model 1: } \hat{y} = \hat{\beta}_0^{(1)} + \hat{\beta}_1^{(1)}x$$

Sample 2
Size N

Train →

$$\text{Model 2: } \hat{y} = \hat{\beta}_0^{(2)} + \hat{\beta}_1^{(2)}x$$

Sample 3
Size N

Train →

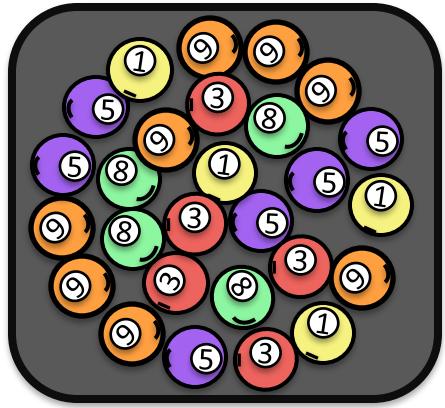
$$\text{Model s: } \hat{y} = \hat{\beta}_0^{(s)} + \hat{\beta}_1^{(s)}x$$

Combine models

$$\mu_{\hat{\beta}} = \frac{1}{s} \sum_{i=1}^s \hat{\beta}^{(i)}$$

$$\sigma_{\hat{\beta}} = \sqrt{\frac{1}{s-1} \sum_{i=1}^s (\hat{\beta}^{(i)} - \bar{\beta})^2}$$

In summary, for each “Parallel Universe”...



Train

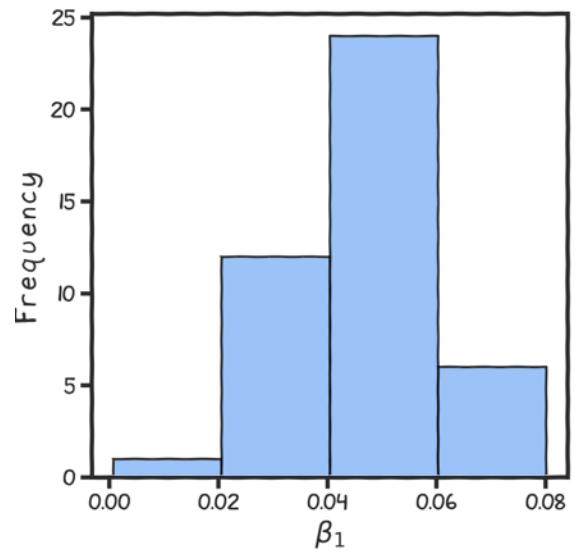
$$\text{Model } i: \hat{y} = \hat{\beta}_0^{(i)} + \hat{\beta}_1^{(i)}x$$



Combine
all models

$$\mu_{\hat{\beta}} = \frac{1}{s} \sum_{i=1}^s \hat{\beta}^{(i)}$$

$$\sigma_{\hat{\beta}} = \sqrt{\frac{1}{s-1} \sum_{i=1}^s (\hat{\beta}^{(i)} - \bar{\beta})^2}$$



s models

Bootstrapping for Estimating Sampling Error

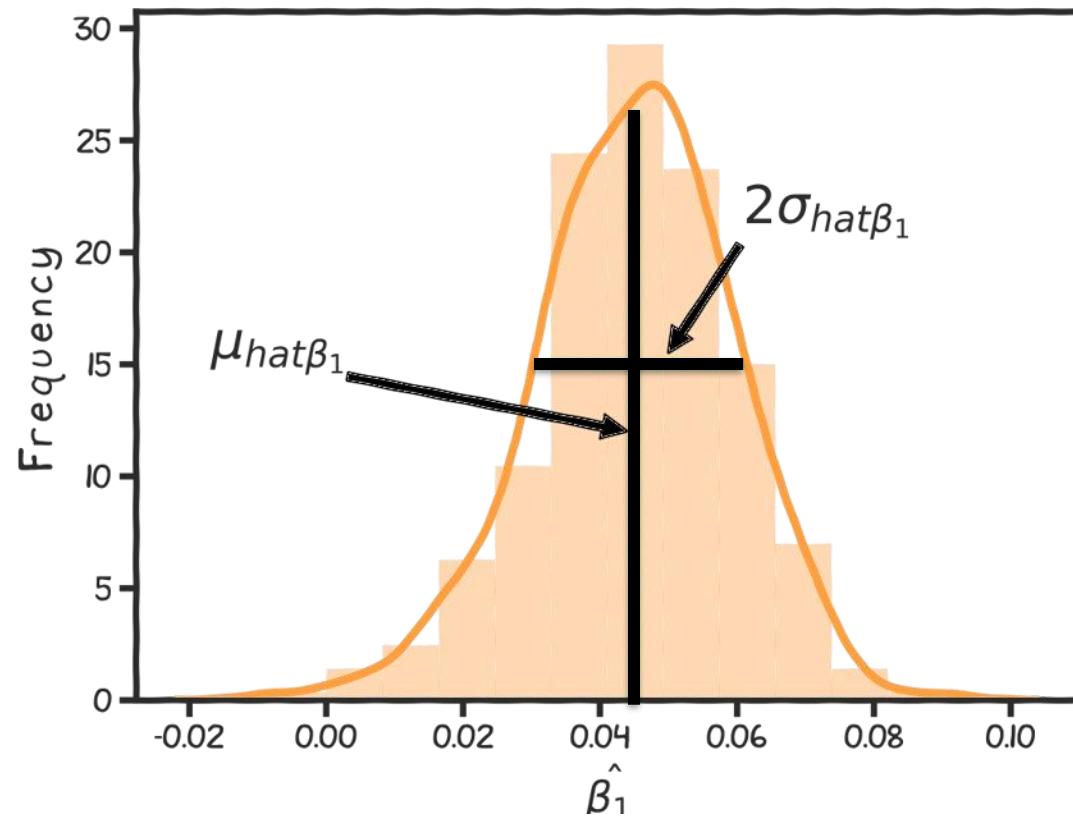
Definition

Bootstrapping is the practice of estimating properties of an estimator by measuring those properties by sampling from the observed data.

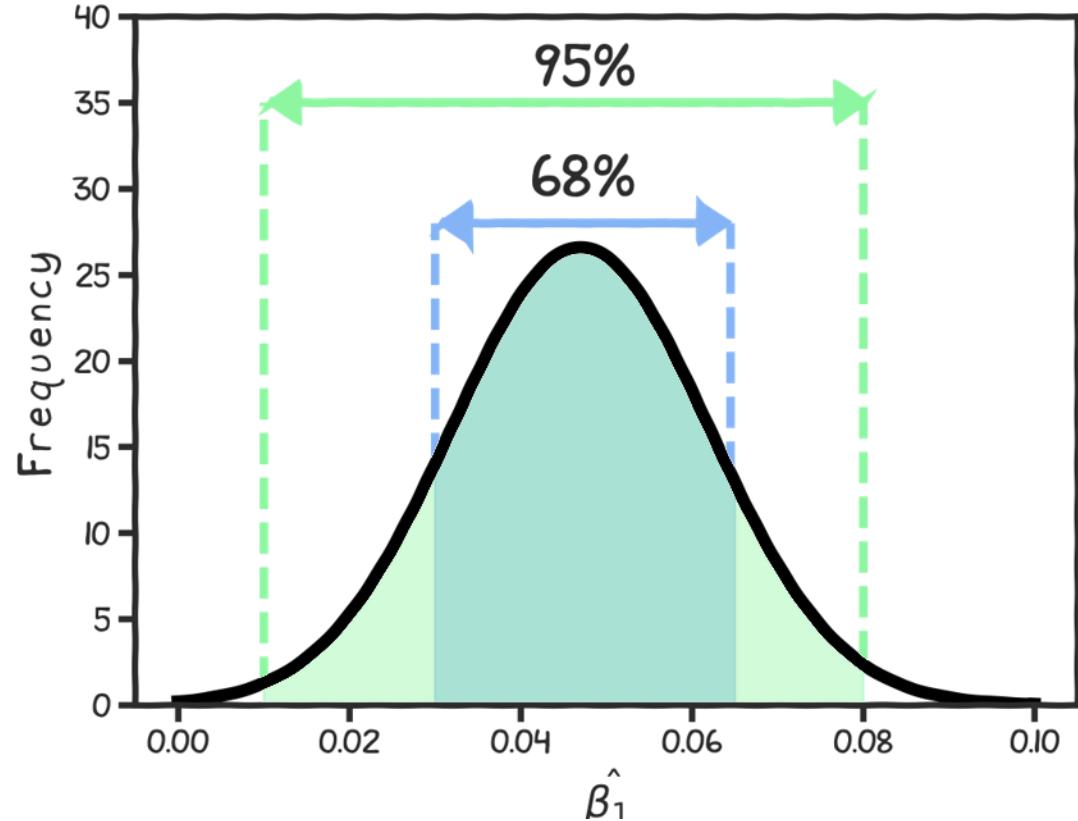
For example, we can compute $\hat{\beta}_0$ and $\hat{\beta}_1$ multiple times by randomly sampling from our data set. We then use the variance of our multiple estimates to approximate the true variance of $\hat{\beta}_0$ and $\hat{\beta}_1$.

Confidence intervals for the predictors estimates (cont)

We can empirically estimate the standard deviations $\hat{\sigma}_{\hat{\beta}}$ which are called the **standard errors**, $SE(\hat{\beta}_0), SE(\hat{\beta}_1)$ through bootstrapping.



Confidence intervals for the predictor estimates (cont.)



The standard errors give us a sense of our uncertainty over our estimates.

Typically, we express this uncertainty as a **95% confidence interval**, which is the range of values such that the **true value** of β_1 is contained in this interval with 95% percent probability.

Standard Errors based on probability theory

Alternatively: If we assume normality, then:

And if we know the variance σ_ϵ^2 of the noise ϵ , we can compute $SE(\hat{\beta}_0), SE(\hat{\beta}_1)$ analytically using the formulae below (no need to bootstrap):

$$SE(\hat{\beta}_0) = \sigma_\epsilon \sqrt{\frac{1}{n} + \frac{\bar{x}^2}{\sum_i (x_i - \bar{x})^2}}$$

$$SE(\hat{\beta}_1) = \frac{\sigma_\epsilon}{\sqrt{\sum_i (x_i - \bar{x})^2}}$$

Where n is the number of observations.

\bar{x} is the mean value of the predictor.

$$CI_{\hat{\beta}}(95\%) = [\hat{\beta} - 2SE(\hat{\beta}), \hat{\beta} + 2SE(\hat{\beta})]$$

Standard Errors

In practice, we do not know the value of σ_ϵ since we do not know the exact distribution of the noise ϵ .

However, if we make the following **assumptions**:

- the errors $\epsilon_i = y_i - \hat{y}_i$ and $\epsilon_j = y_j - \hat{y}_j$ are uncorrelated, for $i \neq j$,
- each ϵ_i has a mean 0 and variance σ_ϵ^2 ,

then, we can empirically estimate σ^2 , from the data and our regression line:

$$\sigma_\epsilon = \sqrt{\frac{n \cdot MSE}{n - 2}} = \sqrt{\sum \frac{(\hat{f}(x) - y_i)^2}{n - 2}}$$

Remember: $y_i = f(x_i) + \epsilon_i \Rightarrow \epsilon_i = y_i - f(x_i)$

Inference in Linear Regression

Uncertainty in estimating the linear regression coefficients

CS109A Introduction to Data Science

Pavlos Protopapas, Natesh Pillai and Chris Gumb

Summary so far

- Statistical model
- k-nearest neighbors (kNN)
- Model fitness and model comparison (MSE)
- Goodness of fit (R²)
- Linear Regression, multi-linear regression and polynomial regression
- Model selection using validation and cross validation
- One-hot encoding for categorical variables
- Overfitting
- Ridge and Lasso regression
- Probability in regression/MLE

Comparison of Models

We have seen already 3 models. Choosing the right model isn't about minimizing the test error. We also want to understand and get insights from our models.

	Has $f(x)$ parametric	Easy to interpret
Linear Regression	Yes	Yes
Polynomial Regression	Yes	No
K-Nearest Neighbors	No	Yes

Having an explicit functional form of $f(x)$ makes it easy to store.

Interpretation is important to evaluating the model and understanding what the data tells us

Take home message

By taking a probabilistic approach to linear regression and assuming the residuals are normally distributed, we see that **maximizing the likelihood** for this model is equivalent to **minimizing mean squared error** around the line!

So, if we believe our residuals are normally distributed, then minimizing mean square error is a natural choice.

Outline

Part A and B: Assessing the Accuracy of the Coefficient Estimates

Bootstrapping and confidence intervals

Part C: Evaluating Significance of Predictors

Does the outcome depend on the predictors?

Hypothesis testing

Part D: How well do we know \hat{f}

The confidence intervals of \hat{f}

Outline

Part A and B: Assessing the Accuracy of the Coefficient Estimates

Bootstrapping and confidence intervals

Part C: Evaluating Significance of Predictors

Does the outcome depend on the predictors?

Hypothesis testing

Part D: How well do we know \hat{f}

The confidence intervals of \hat{f}

How reliable are the model interpretation

Suppose our model for advertising is:

$$y = 1.01x + 0.005$$

where y is the sales in 1000 units and each unit sells for \$1, x is the TV budget in \$1000.

Interpretation: for every dollar invested in advertising gets you 1.01 back in sales, which is 1% net increase.

How reliable are the model interpretation

$$y = 1.01x + 0.005$$

But how certain are we in our estimation of the coefficient 1.01? Why aren't we certain?

In order to assess these questions, we need to get a sense of the variability of our estimate(s)...they won't be 100% on target. That way we can build a range of plausible values of the true β_1 around our estimate $\hat{\beta}_1$. This is called a.....

Confidence Interval

There are many ways to build a confidence interval. We will see two options in today's class (the two most common approaches):

1. Using Bootstrap resamples
2. Using formulas based on probability theory

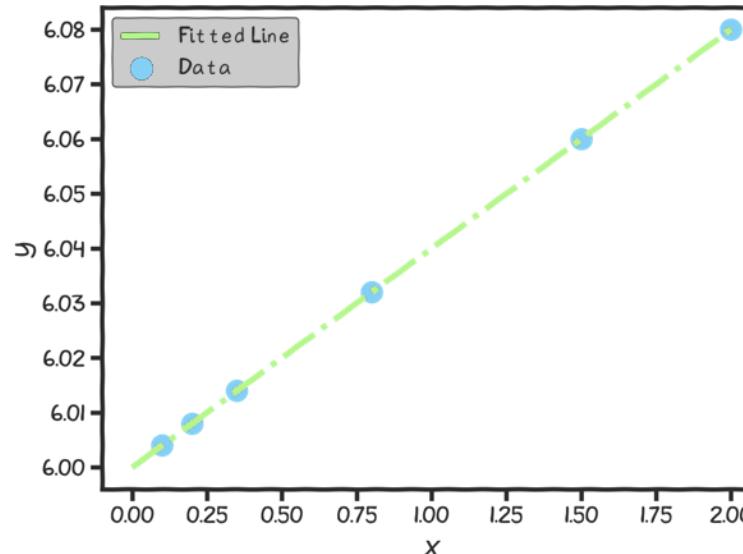
Confidence intervals for the predictors estimates

We interpret the ϵ term in our observation

$$y = f(x) + \epsilon$$

to be noise introduced by random variations in natural systems or imprecisions of our scientific instruments and everything else.

If we knew the exact form of $f(x)$, for example, $f(x) = \beta_0 + \beta_1 x$, and there was no noise in the data , then estimating the $\hat{\beta}$'s would have been exact (so is 1.01 worth it?).



Confidence intervals for the predictors estimates (cont.)

However, two things happen, which result in mistrust of the values of $\hat{\beta}$'s :

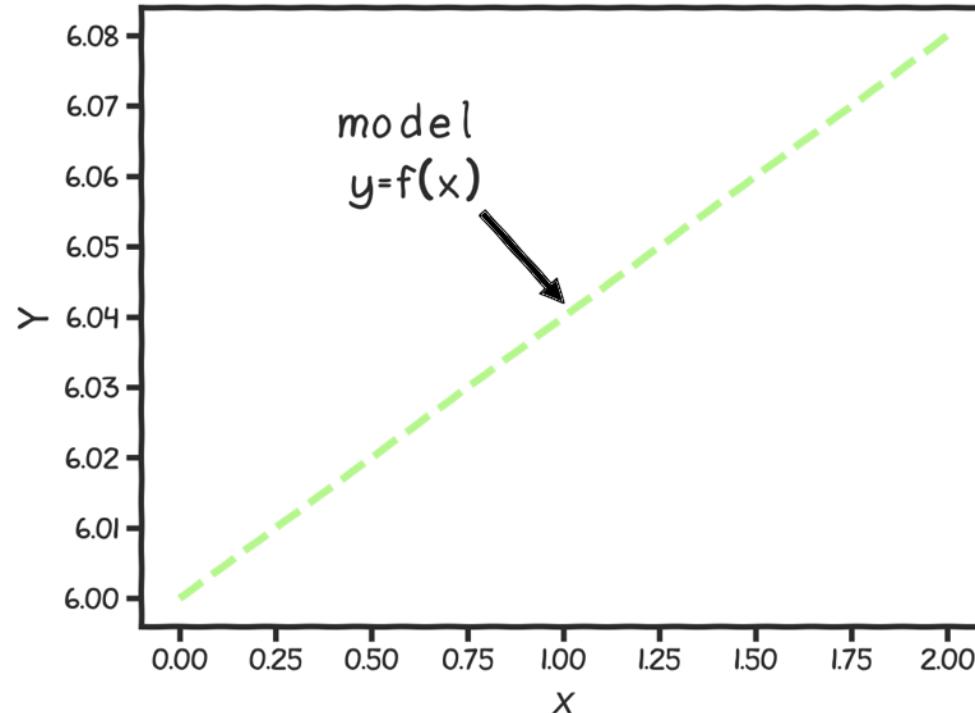
- observational error is always there - this is called **aleatoric** error, or **irreducible** error.
- we do not know the exact form of $f(x)$ - this is called **misspecification** error, and it is part of the **epistemic** error

We will put everything into catch-it-all term ε .

Because of ε , **every time** we measure the response y for a fix value of x , we will obtain a **different** observation, and hence a different estimate of $\hat{\beta}$'s.

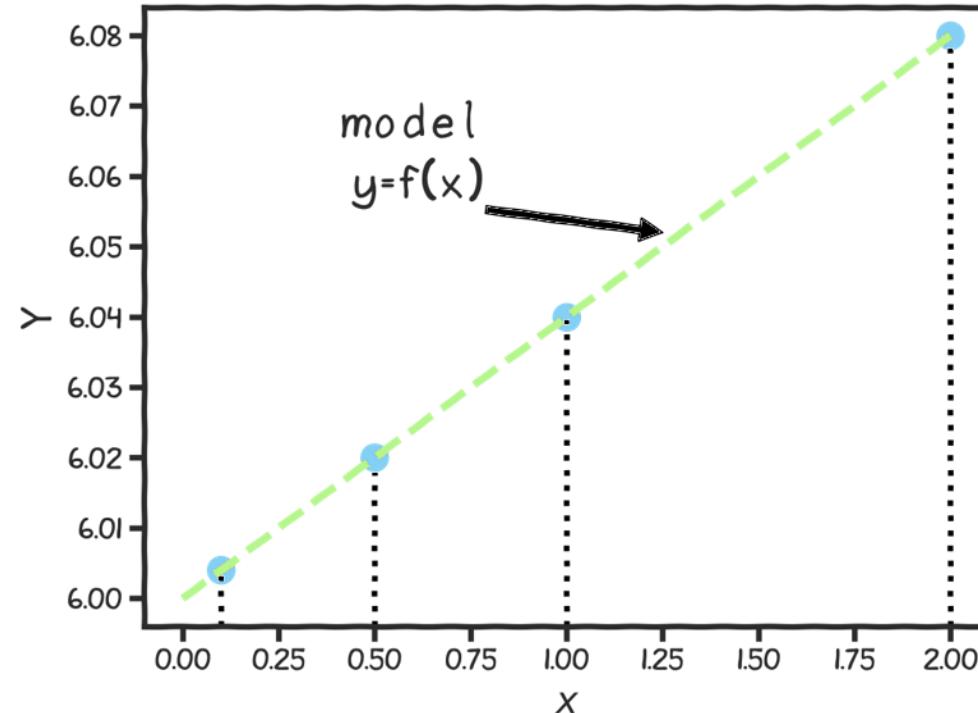
Confidence intervals for the predictors estimates (cont.)

Start with a model $f(X)$, the correct relationship between input and outcome.



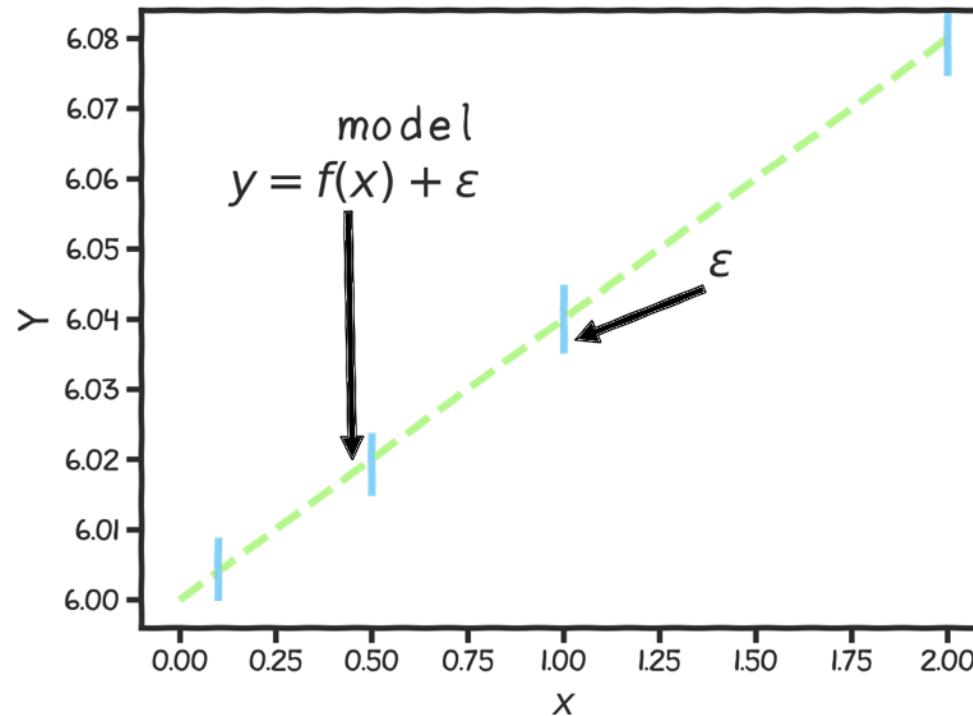
Confidence intervals for the predictors estimates (cont.)

For some values of X^* , $Y^* = f(X^*)$



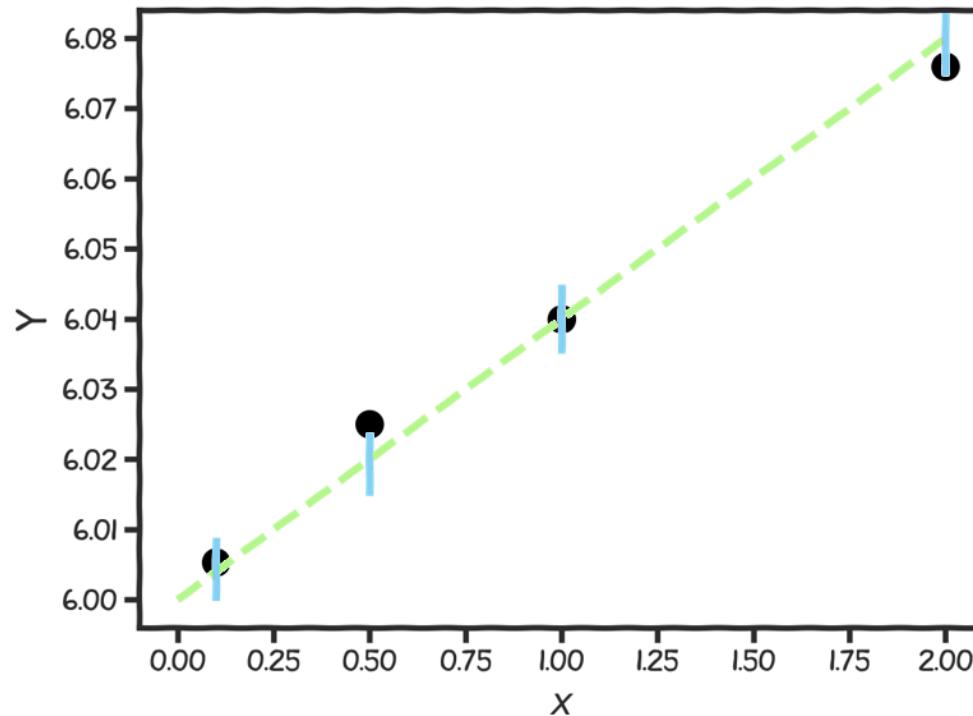
Confidence intervals for the predictors estimates (cont.)

But due to error, every time we measure the response Y for a fixed value of X^* we will obtain a different observation.



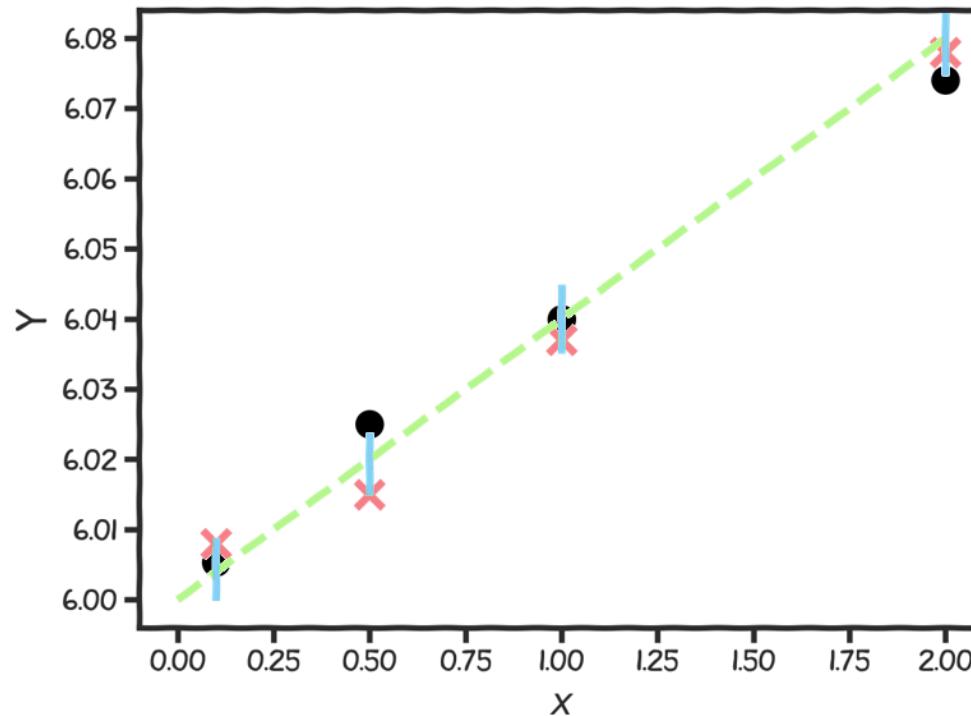
Confidence intervals for the predictors estimates (cont.)

One set of observations, “one realization” yields one set of Y s
(Circles: ●).



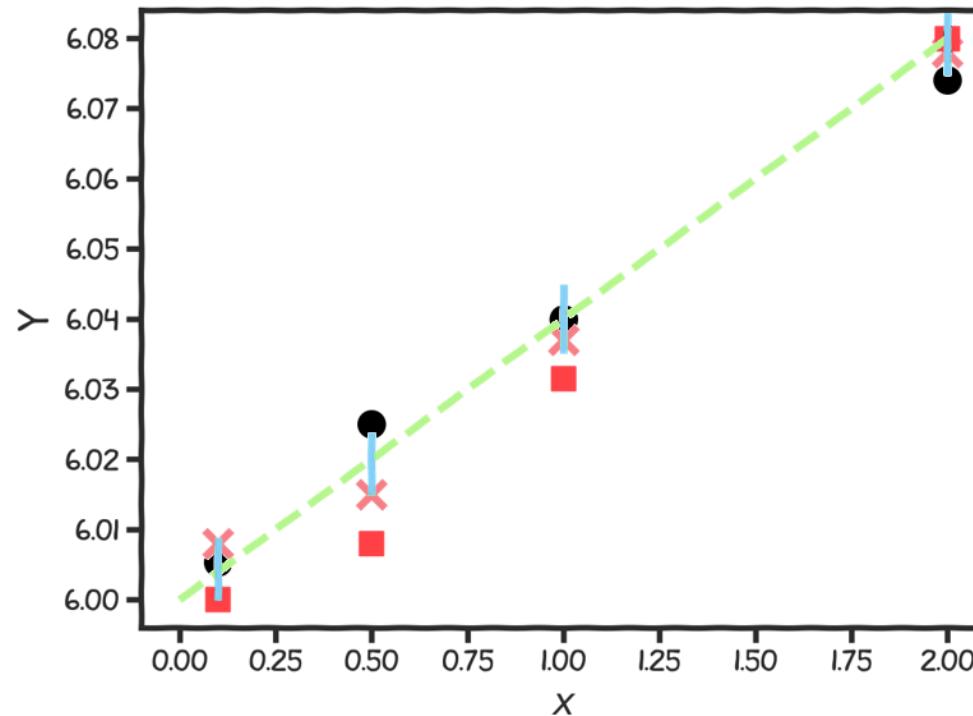
Confidence intervals for the predictors estimates (cont.)

Another set of observations, “another realization” yields another set of Y 's (Crosses: \times).



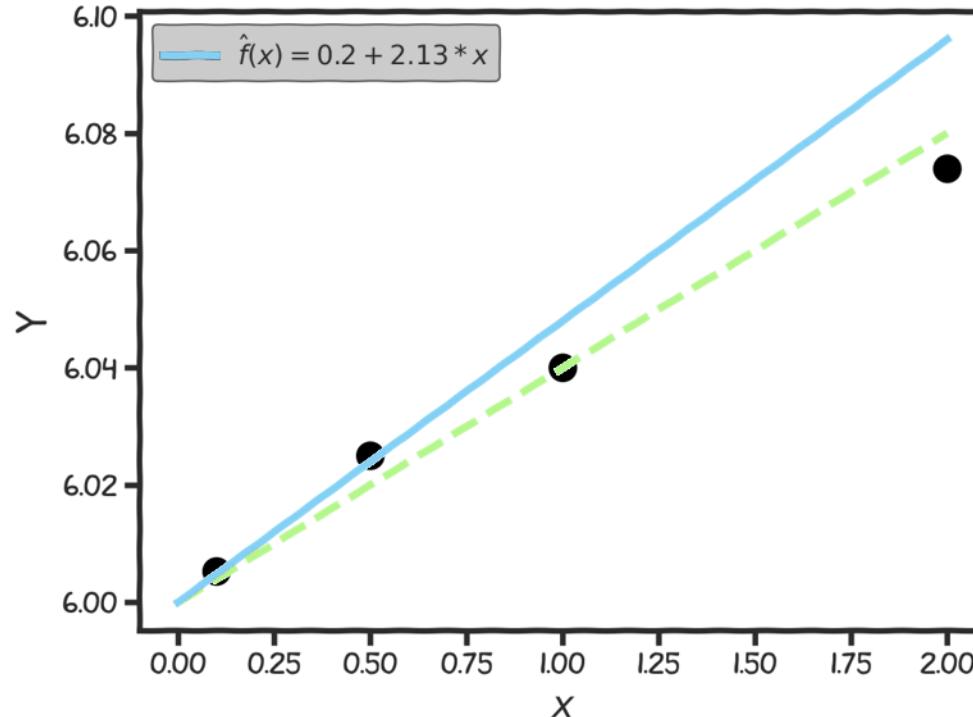
Confidence intervals for the predictors estimates (cont.)

Another set of observations, “another realization”, another set of Y 's (Squares: ■).



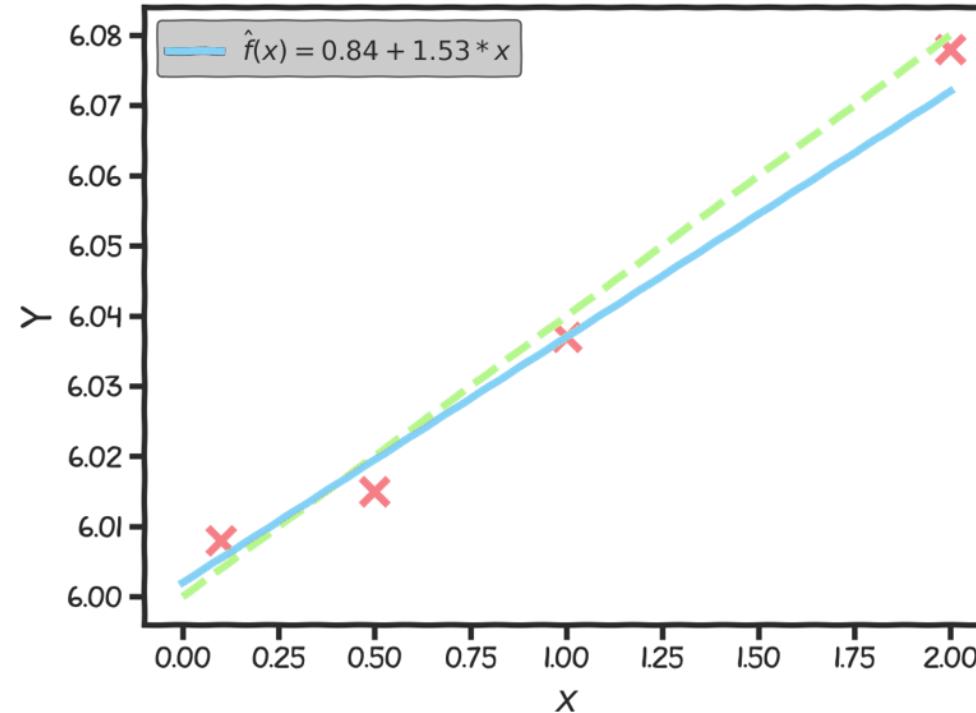
Confidence intervals for the predictors estimates (cont.)

For each one of those “realizations”, we fit a model and estimate $\hat{\beta}_0$ and $\hat{\beta}_1$.



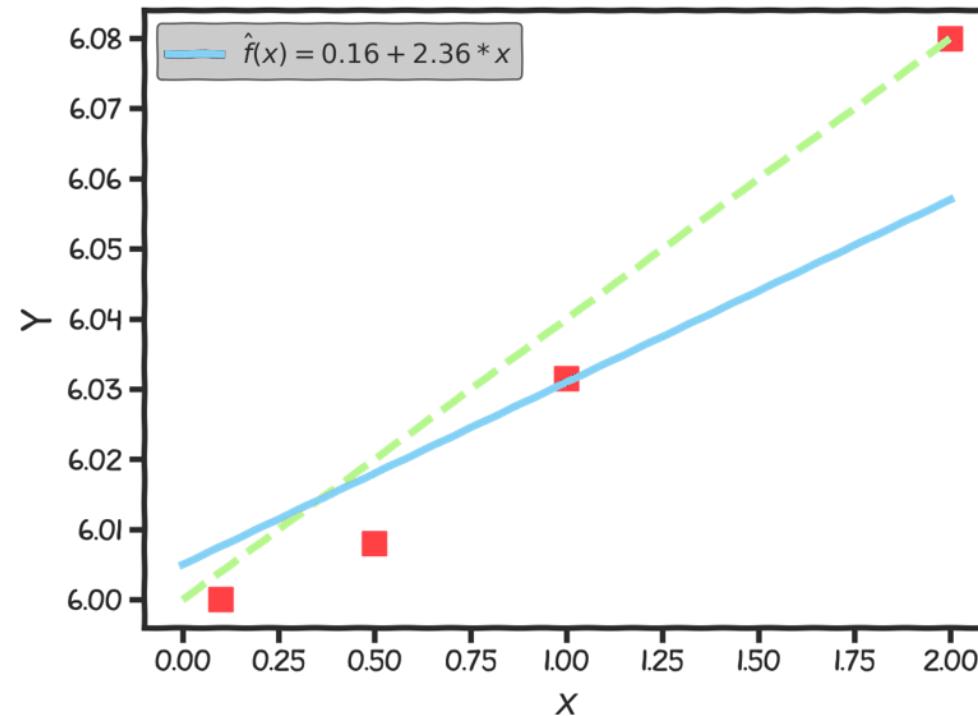
Confidence intervals for the predictors estimates (cont.)

For another “realization”, we fit another model and get different values of $\hat{\beta}_0$ and $\hat{\beta}_1$.



Confidence intervals for the predictors estimates (cont.)

For another “realization”, we fit another model and get different values of $\hat{\beta}_0$ and $\hat{\beta}_1$.

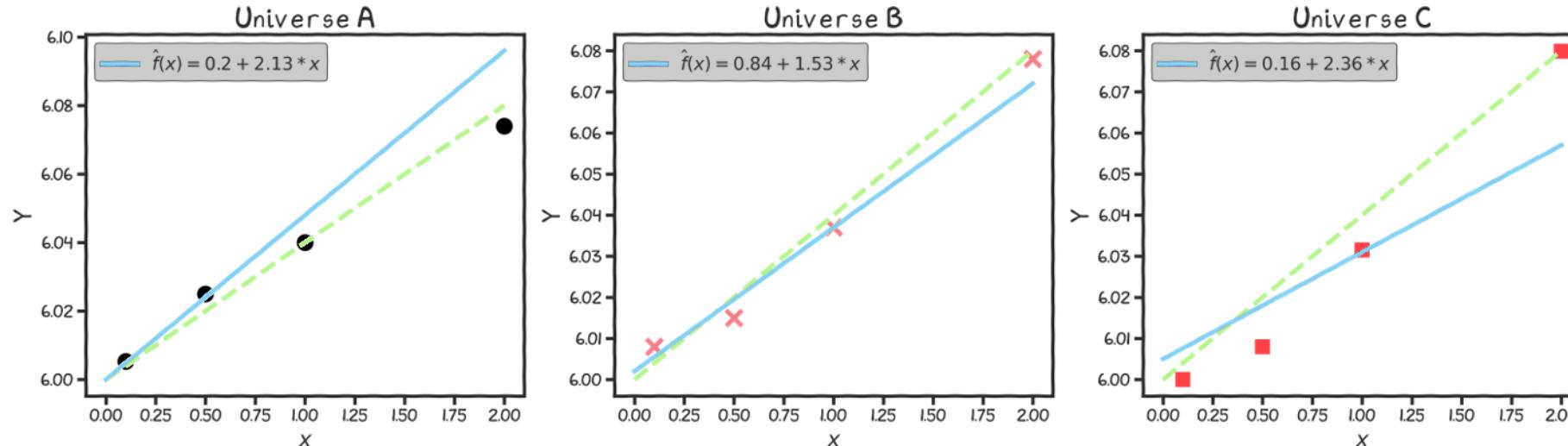


Confidence intervals for the predictors estimates (cont.)

So, if we have one set of measurements of $\{X, Y\}$, our estimates of $\hat{\beta}_0$ and $\hat{\beta}_1$ are just for that particular realization.

Question: If this is just one realization of reality, how do we know the truth? How do we deal with this conundrum?

Imagine (magic realism) we have parallel universes, and we repeat this experiment on each of the other universes.

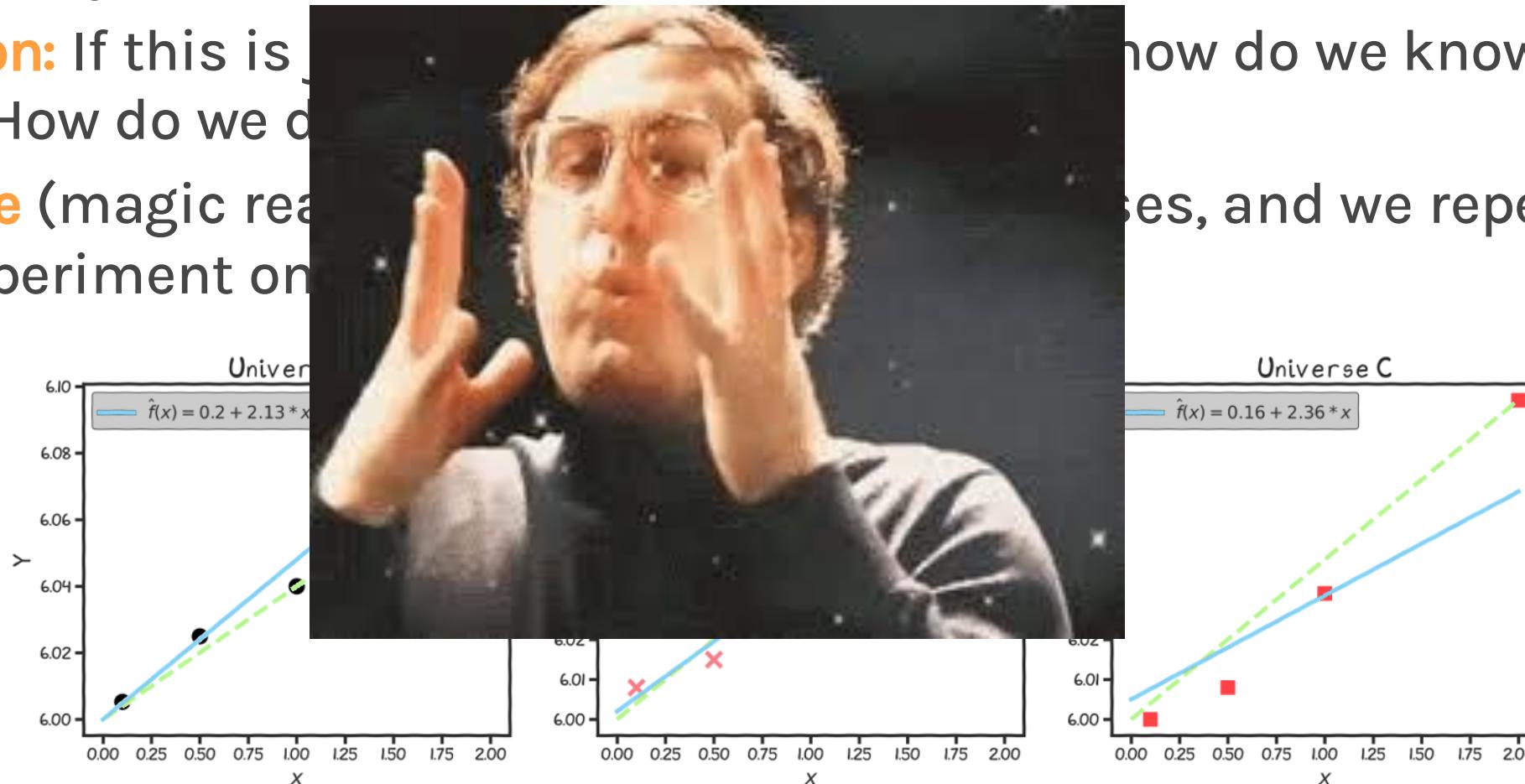


Confidence intervals for the predictors estimates (cont.)

So, if we have one set of measurements of $\{X, Y\}$, our estimates of $\hat{\beta}_0$ and $\hat{\beta}_1$ are just for that particular realization.

Question: If this is just one realization, how do we know the truth? How do we do this?

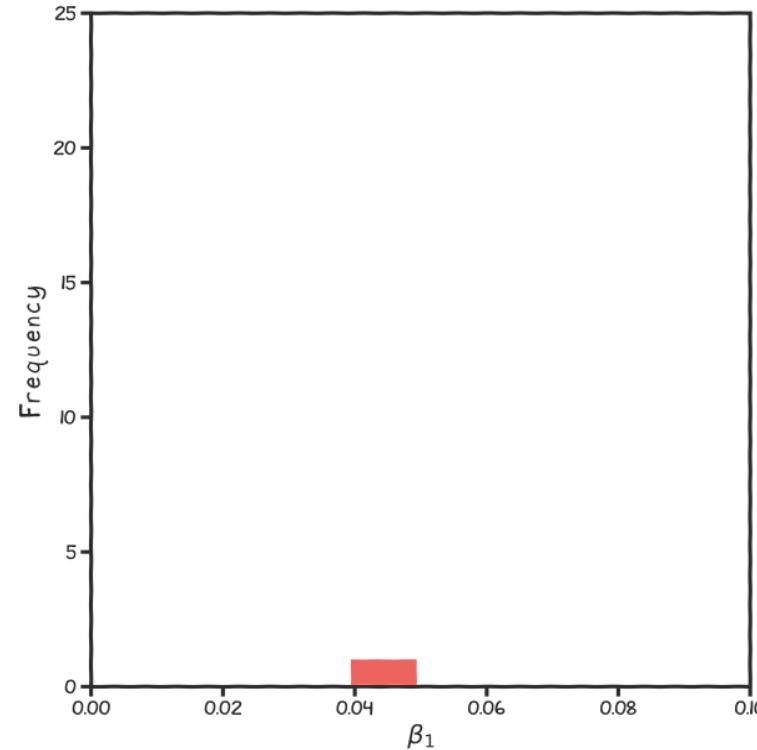
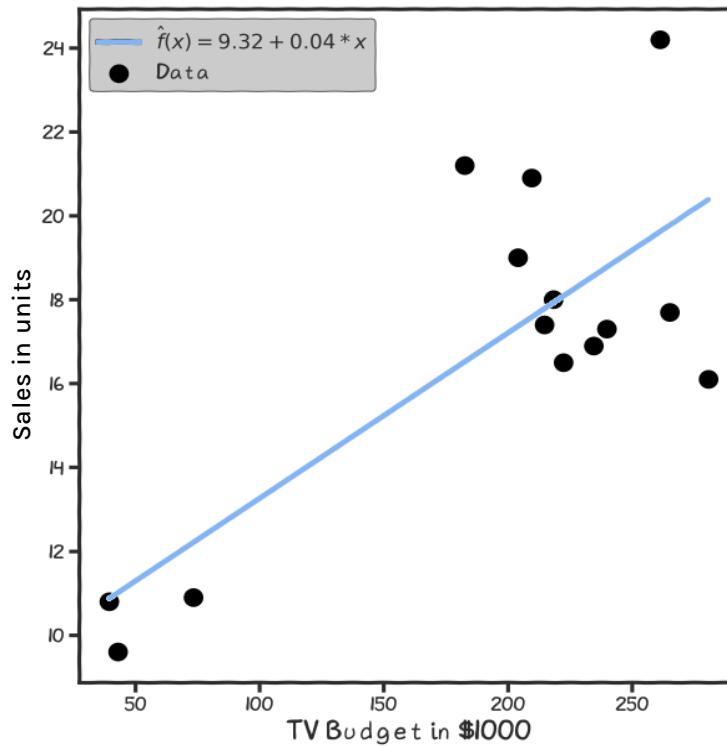
Imagine (magic reader) we do this many times, and we repeat this experiment on



Confidence intervals for the predictors estimates (cont.)

In our magical realisms, we can now sample multiple times.

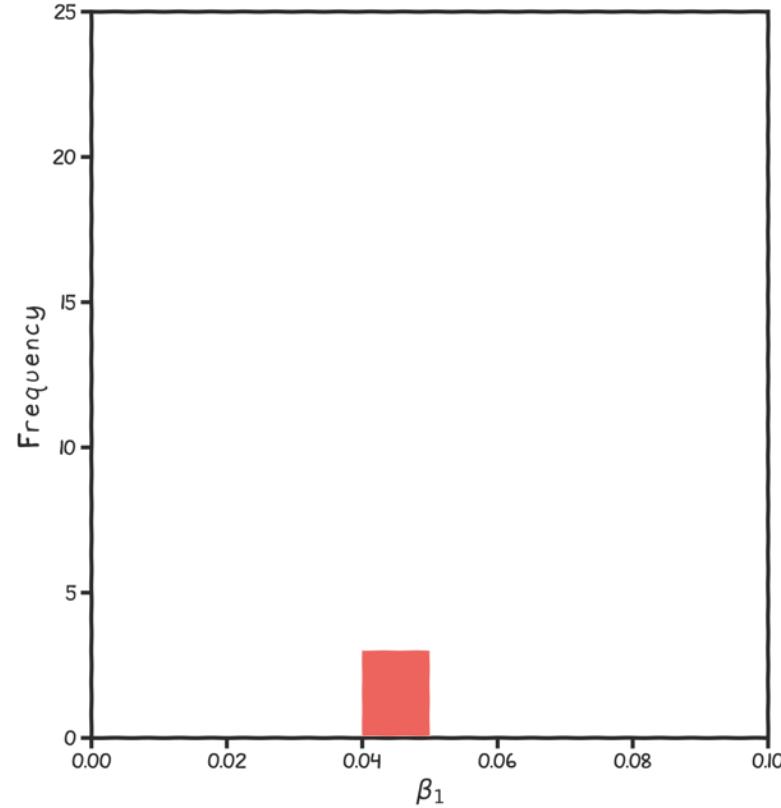
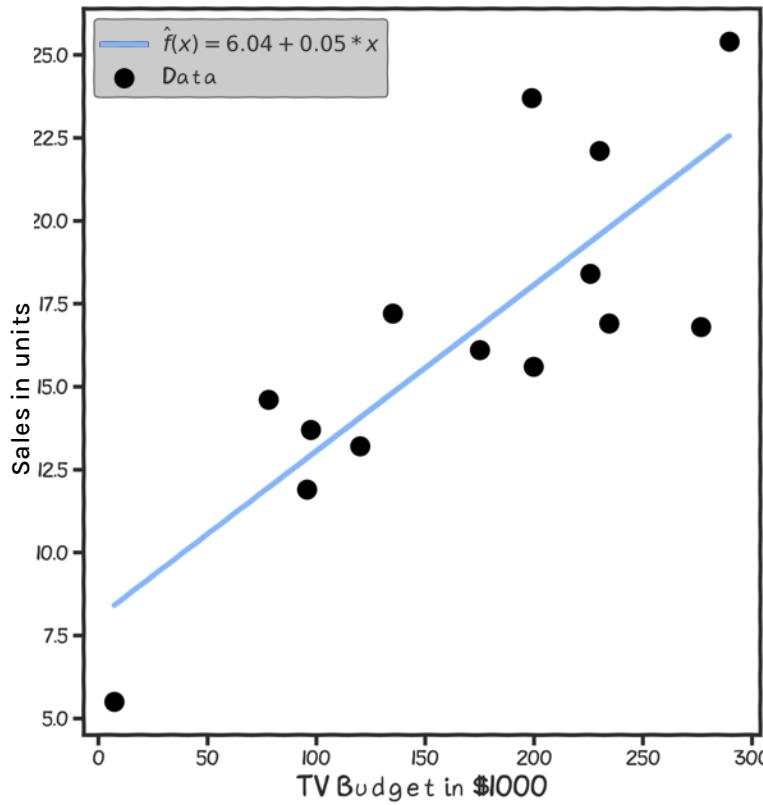
One universe, one sample, one set of estimates for $\hat{\beta}_0, \hat{\beta}_1$



There will be an equivalent plot for $\hat{\beta}_0$ which we don't show here for simplicity

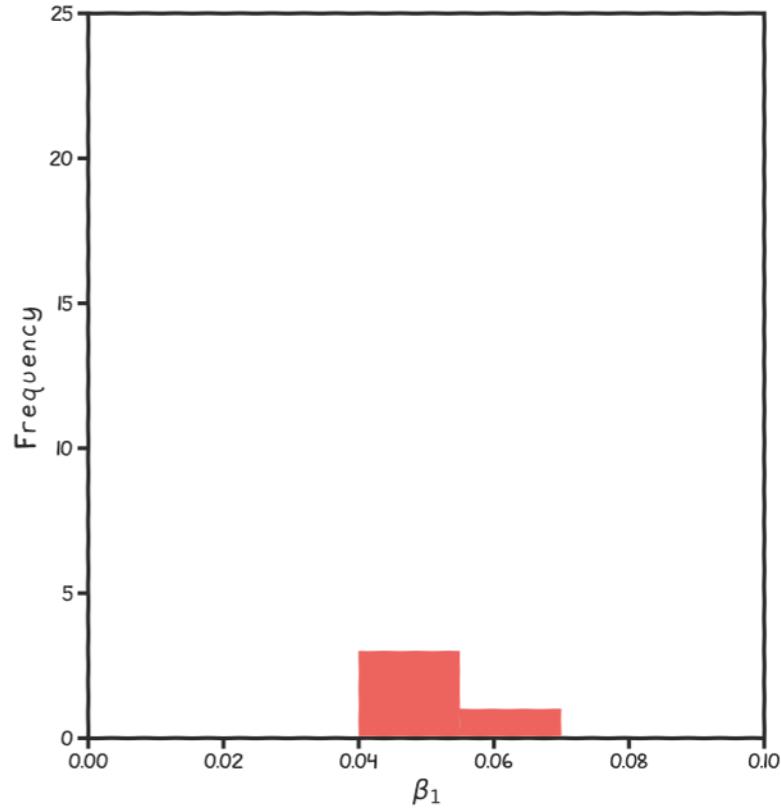
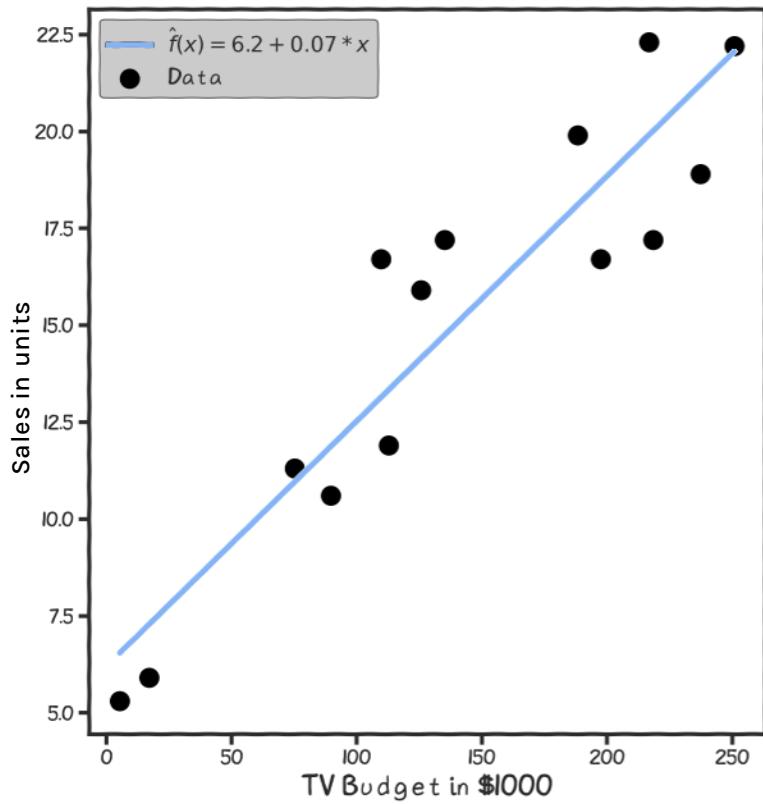
Confidence intervals for the predictors estimates (cont.)

Another sample, another estimate of $\hat{\beta}_0, \hat{\beta}_1$



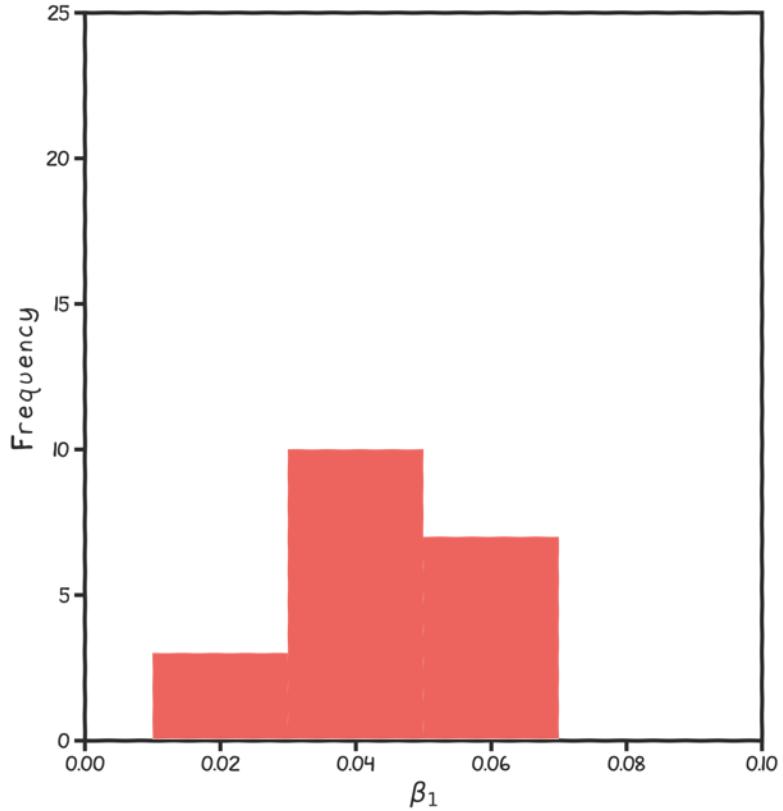
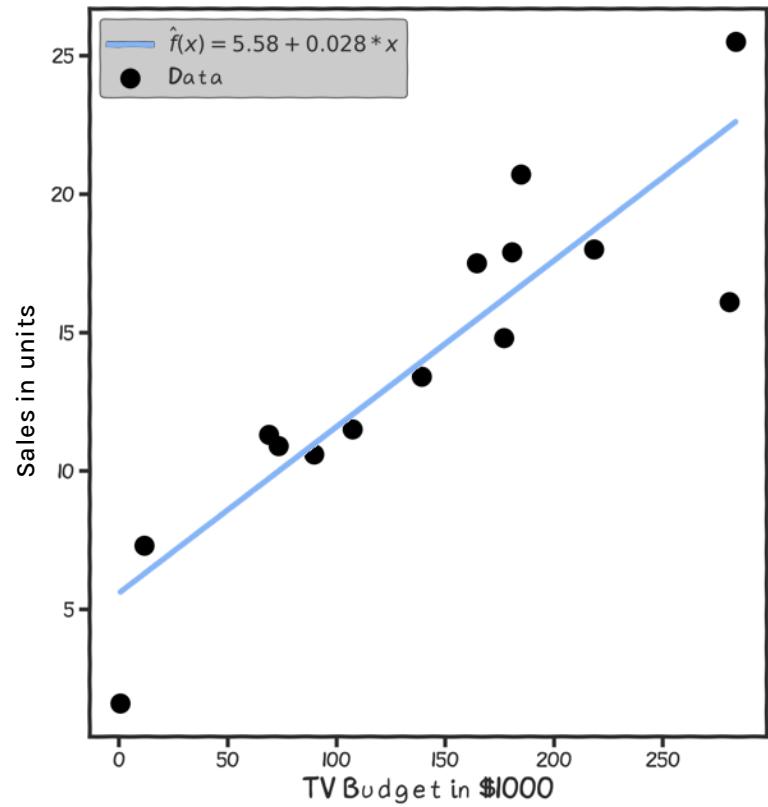
Confidence intervals for the predictors estimates (cont.)

Again



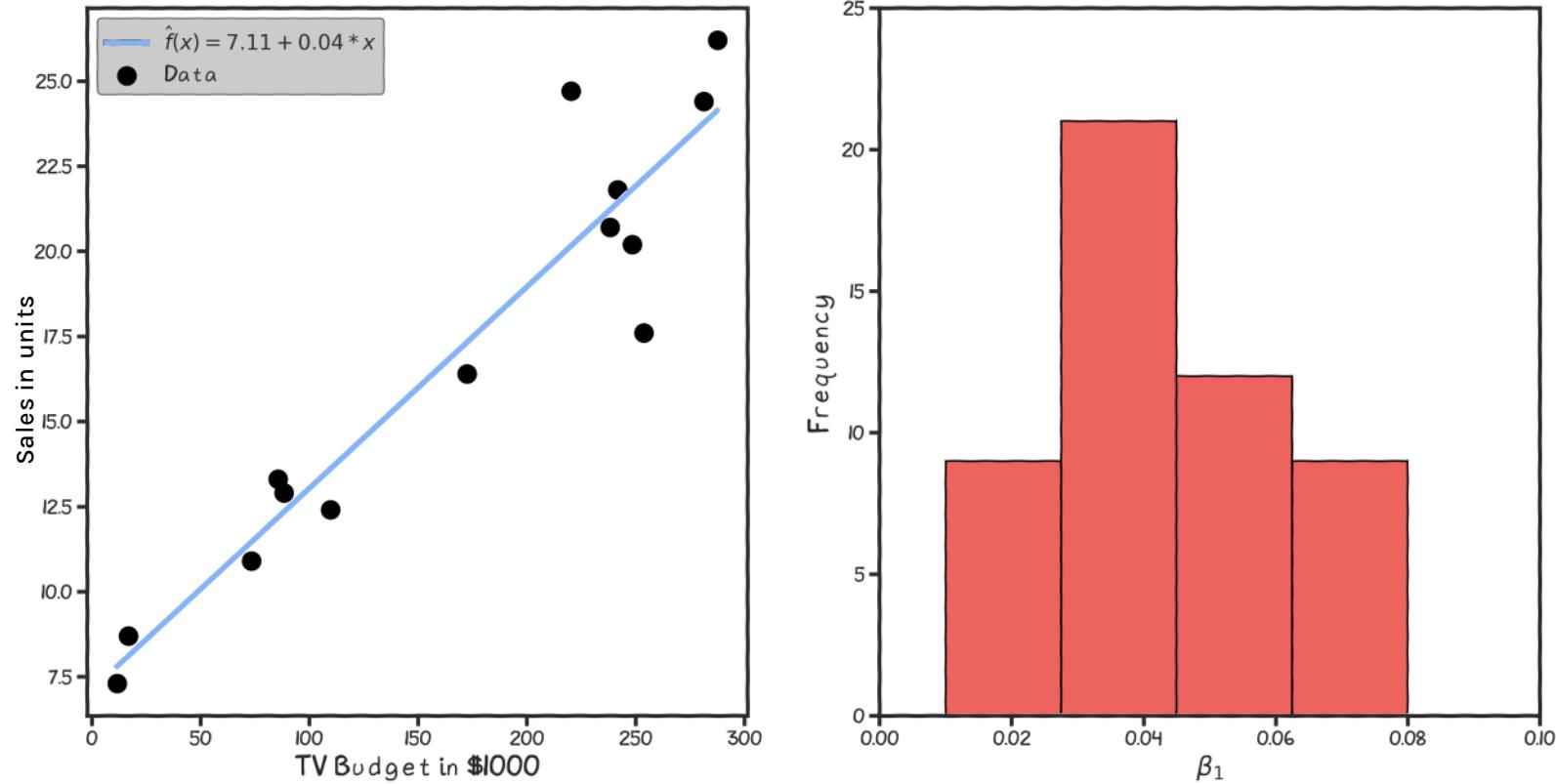
Confidence intervals for the predictors estimates (cont.)

And again



Confidence intervals for the predictors estimates (cont.)

Repeat this for 100 times, until we have enough samples of $\hat{\beta}_0, \hat{\beta}_1$.



From Random Variables to Maximum Likelihood Estimation (MLE)



CS109A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai and Chris Gumb

Dylan Wu
Mt. Fuji

Outline

- What is a random variable?
- Point estimates of random variables. Confidence Intervals, Histogram, Probability, and PDF/PMF
- Known random variables: Uniform, Binomial, Normal
- Joint Distributions
- Modeling Data with Probability Distributions
- Likelihood Theory
- Modeling Linear Regression Probabilistically

Outline

- **What is a random variable?**
- Point estimates of random variables. Confidence Intervals, Histogram, Probability, and PDF/PMF
- Known random variables: Uniform, Binomial, Normal
- Joint Distributions
- Modeling Data with Probability Distributions
- Likelihood Theory
- Modeling Linear Regression Probabilistically

CS109A Olympics



WHO WILL WIN THE 100M DASH?

Option A



THE PROFESSOR
AVERAGE Pace: 13 seconds
Consistency: High

CS109A 100m dash

Option B



Option A



THE HOT SHOT

AVERAGE Pace: 13.0 seconds

Consistency: VERY LOW

CS109A 100m dash

THE BANGALORE CHAMPION

AVERAGE Pace: 13.1 seconds

Consistency: MEDIUM

Option A



Option B



Option C



THE Researcher

AVERAGE Pace: 14 seconds

Consistency: LOW

Option A



Option B



Option C



Option D



Option A



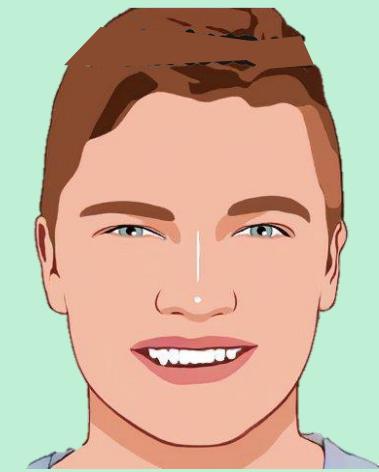
Option B



Option C



Option D



Race #1 13.12 —

12.52

14.25

13.51

Race #2 13.1

RACE TIME

15.01

Race #3 13.0 —

— — —

— — —

13.63

Race #4 12.87 🏆

13.52

13.12 🏅

13.91

Race #5 13.22 PROTOPAPAS

13.24

12.78

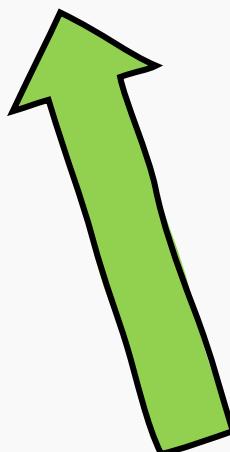
12.32 🏆

CS109A 100m dash

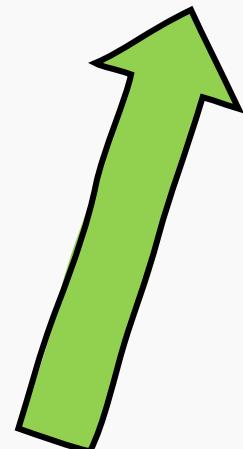
Let X be the race pace for a given 100m dash, then X is called a **random** variable



$$\text{Race Pace} = \text{Average Pace} + \epsilon$$



Constant



Varying

RECAP: Python variables

We have seen variables as something we assign a value to.

- Integer <int>

```
In [2]: a = 2
In [3]: b = 2.5
In [4]: pavloslist = [1,2,3,4,5]
In [5]: pavlosdict ={'John':2,'Pavlos':2,Eric':7}
```

- Float <float>

```
In [6]: type(a)
Out[6]: int
In [7]: type(b)
Out[7]: float
In [8]: type(pavloslist)
Out[8]: list
In [9]: type(pavlosdict)
Out[9]: dict
```

- List <list>

- Dictionary <dict>

Random Variable

- A random variable can be thought of as a **numeric outcome** of a random experiment.
- Unlike the python variables defined before, the **value** of a random variable is not fixed.
- The output of a random variable could be either **discrete** (can only take on specific **values**) or **continuous** (can take on any value within a range).

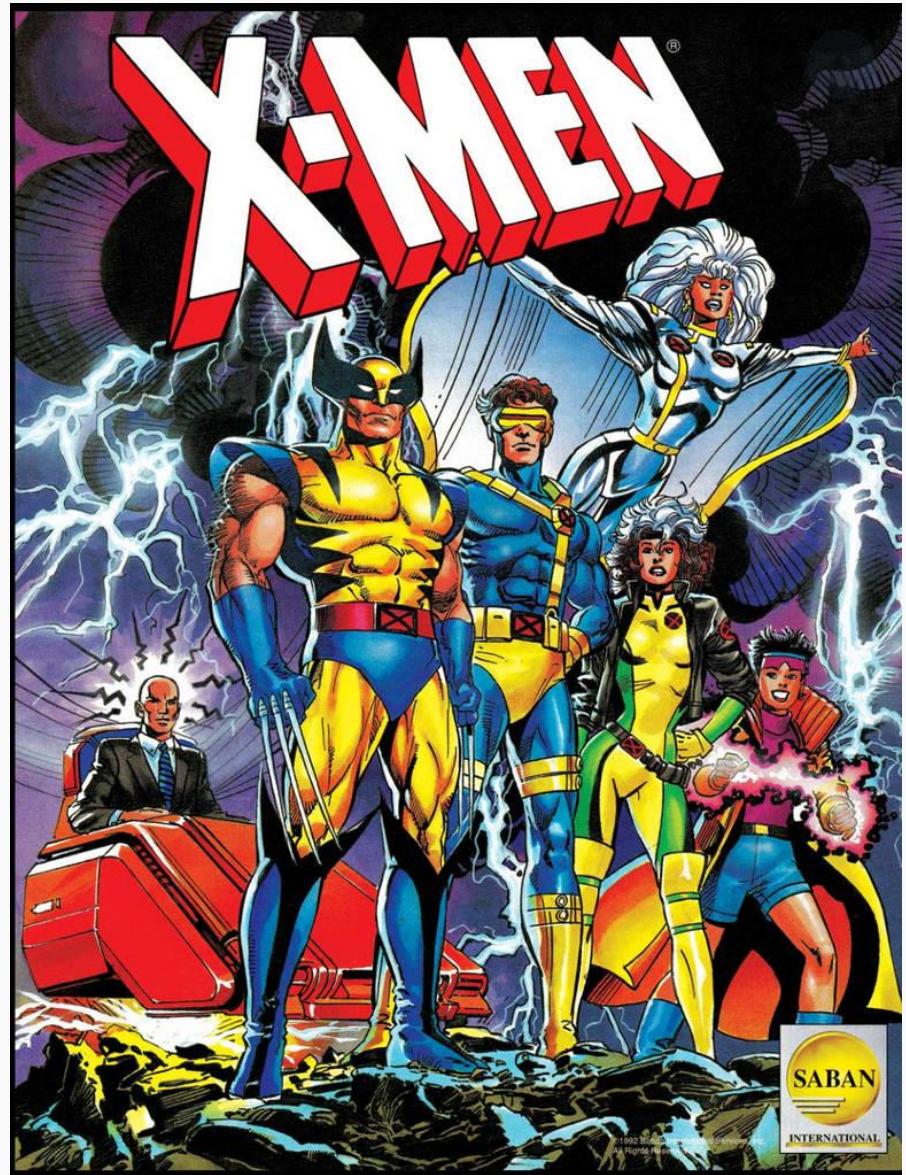
```
In [26]: x = RandomVariable()
```

```
In [27]: x.random
```

```
Out[27]: 0.5632899481539281
```

```
In [28]: x.random
```

```
Out[28]: 0.630954141651853
```





$$X = \text{Average Pace} + \epsilon$$

- What are the possible values of X ?
- What is the maximum value of X ?
- What is the minimum value of X ?
- What is the expected value of X ?
- Are the values of X spread out, or consistent?

AND MANY MORE QUESTIONS ...

```
In [5]: pavlos = Sprinter()
```

```
In [6]: pavlos.time
```

```
Out[6]: 13.431656720548697
```

```
In [7]: pavlos.time
```

```
Out[7]: 13.42798180661262
```

```
In [8]: pavlos.time
```

```
Out[8]: 11.78189462795882
```

```
In [9]: pavlos.time
```

```
Out[9]: 14.77745984741147
```

Simulations



RUNNER: PAVLOS PROTOPAPAS
COUNTRY: CYPRUS
CURRENT TIME: 2.35 s



START

finish



RUNNER: PAVLOS PROTOPAPAS
COUNTRY: CYPRUS
CURRENT TIME: 2.47 s

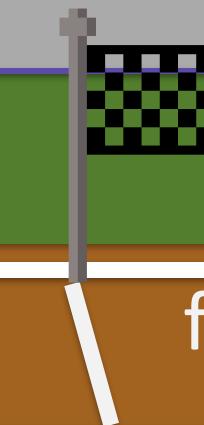


START

finish



RUNNER: PAVLOS PROTOPAPAS
COUNTRY: CYPRUS
CURRENT TIME: 2.75 s



finish

Random Variable

$$X = \text{Average Pace} + \epsilon$$

[13.75, 15.21, 13.65, 13.58, 12.93, 14.23, 12.81, 11.50, 13.09, 12.26, ...]

- We could run the experiment **multiple** times and record the results.
- This will give us a list of **possible values** of the random variable X .
- An **exhaustive** list of all possible values is often called the **population** space of the random experiment.

Let's do it. I will run many runs for you





RUNNER: PA
COUNTRY: C
CURRENT T

Come on guys! How
many
more races do you
need me to run ? I
can't do this all day!

PAS



START

finish

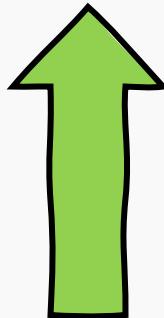
Random Variable

[13.75, 15.21, 13.65, 13.58, 12.93, 14.23, 12.81, 11.50, 13.09, 12.26, ...]

Random Variable

[13.75, 15.21, 13.65, 13.58, 12.93, 14.23, 12.81, 11.50, 13.09, 12.26, ...]

[13.75, 13.65, 12.93, 12.81, 12.26]



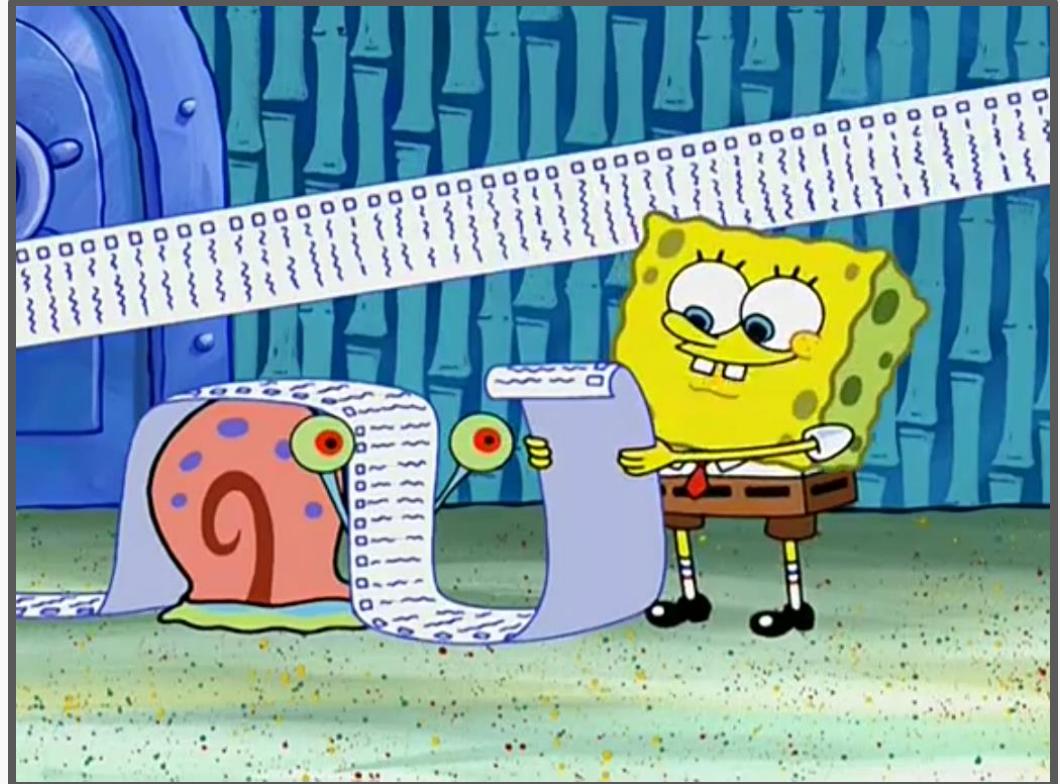
Sample

Properties of a Random Variable

ISSUES?

The outcomes of a random variable captured over multiple simulations is difficult to interpret and consequently difficult to compare to other random variables.

- **ISSUE #1:** We do not have estimates to compare with other random variables.
- **ISSUE #2:** It is difficult to visualize the spread of the outcome.



Properties of a Random Variable

ISSUES?

The outcomes of a random variable captured over multiple simulations is difficult to interpret and consequently difficult to compare to other random variables.

- **ISSUE #1:** We do not have estimates to compare with other random variables.
- **ISSUE #2:** It is difficult to visualize the spread of the outcome.

Description of Random Variables



Point Estimates

Confidence Intervals

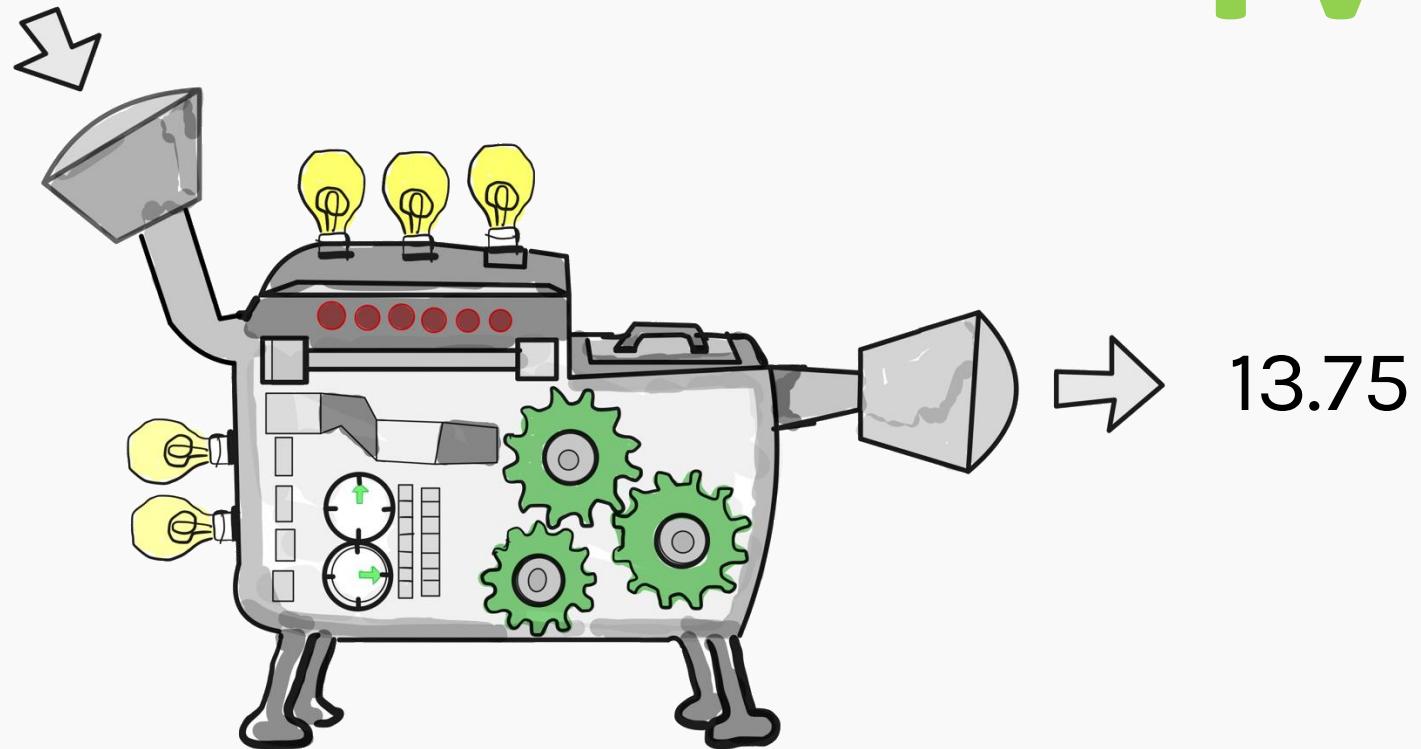
Histogram

Probability Density Function or Probability Mass Functions

Point estimates

Sample
[13.75, 13.65, 12.93, 12.81, 12.26]

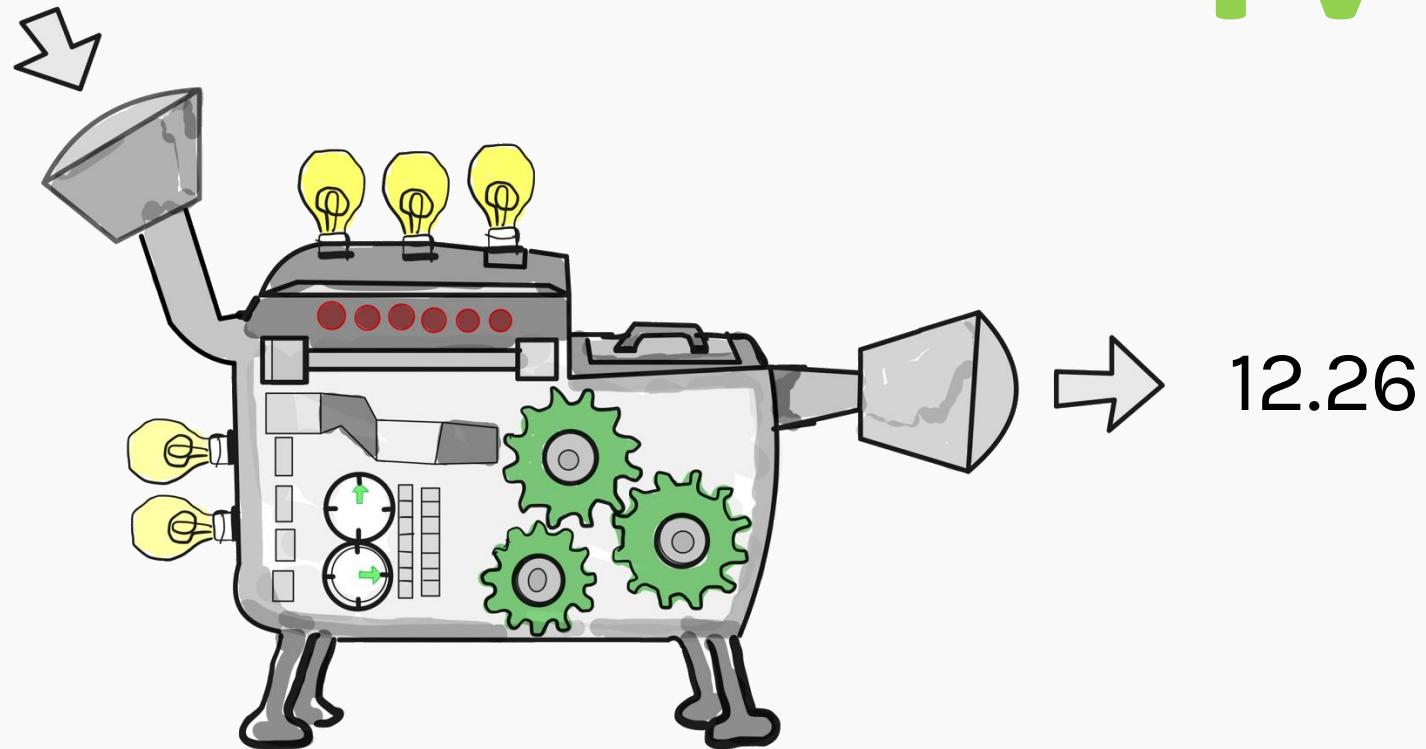
MAX



Point estimates

Sample
[13.75, 13.65, 12.93, 12.81, 12.26]

MIN

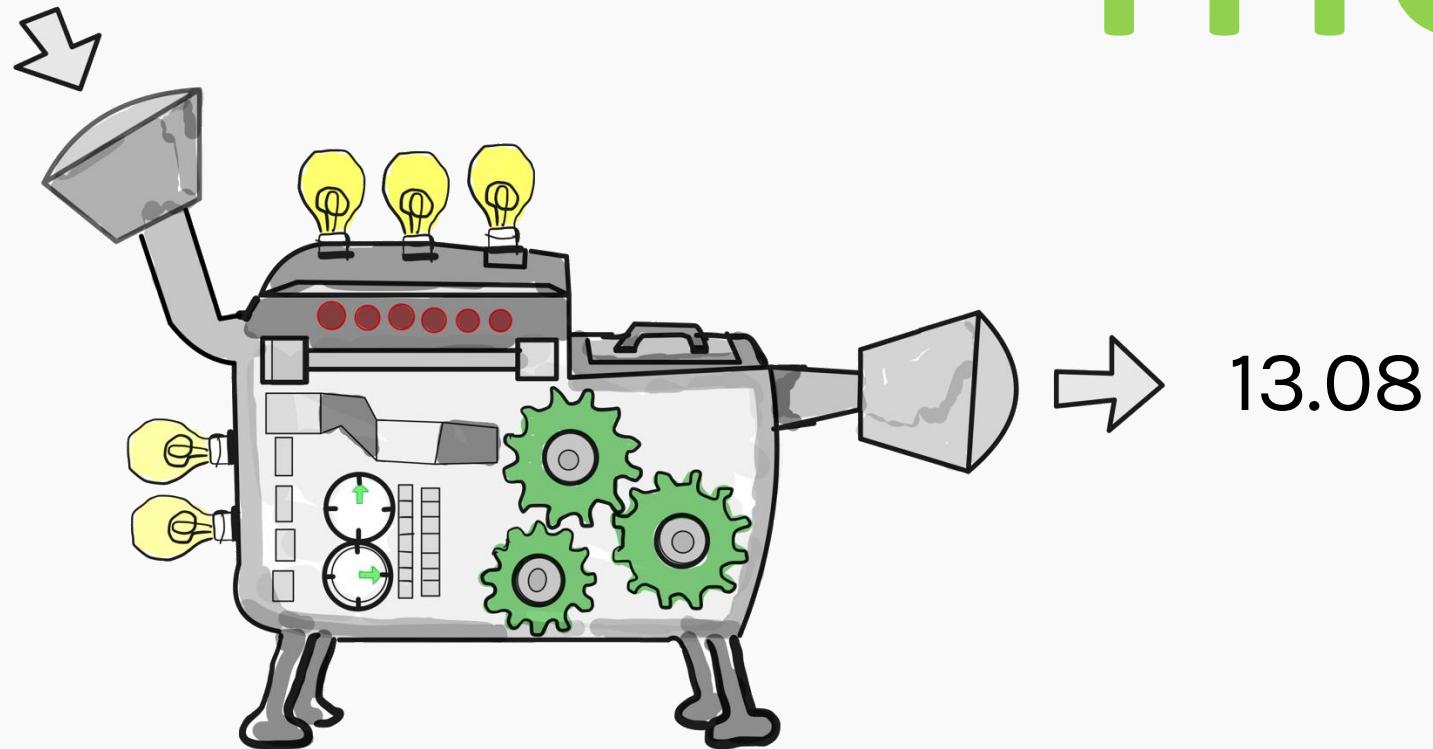


Point estimates

Sample

[13.75, 13.65, 12.93, 12.81, 12.26]

mean



What we want

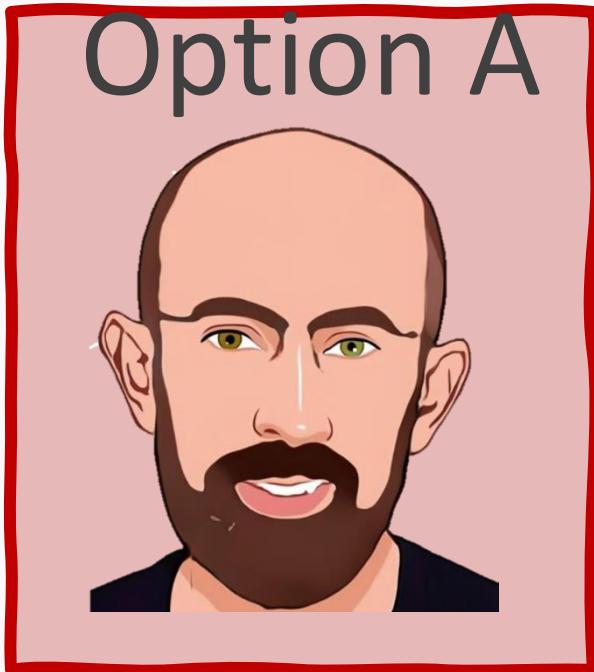
POINT ESTIMATES

- Point estimates can be defined as numbers that give some **information** of the random variable.
- Commonly used point estimates include **max**, **min**, **mean**, **median**, **mode**, **variance**, interquartile range, etc.
- Two major categories describe the **central tendency** & the **spread**.

Population Parameters		Sample Statistics
Mean	μ	\bar{X}
Remember! Population estimates are in Greek and sample estimates are written in roman style!		s^2
Standard Deviation	σ	



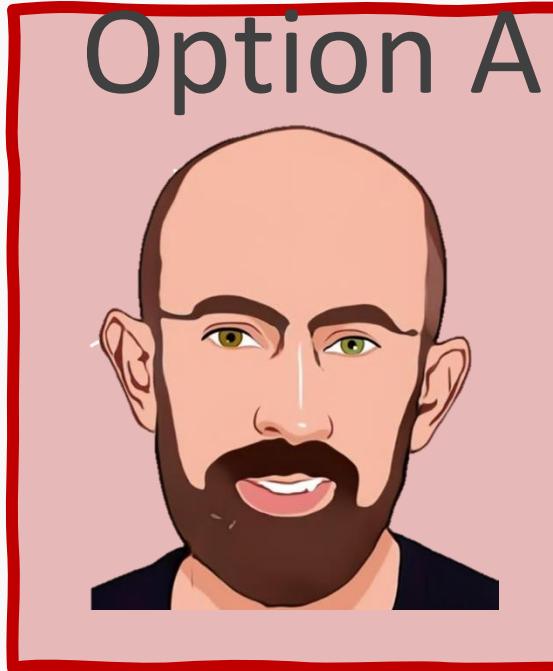
Random Variables - Point estimates



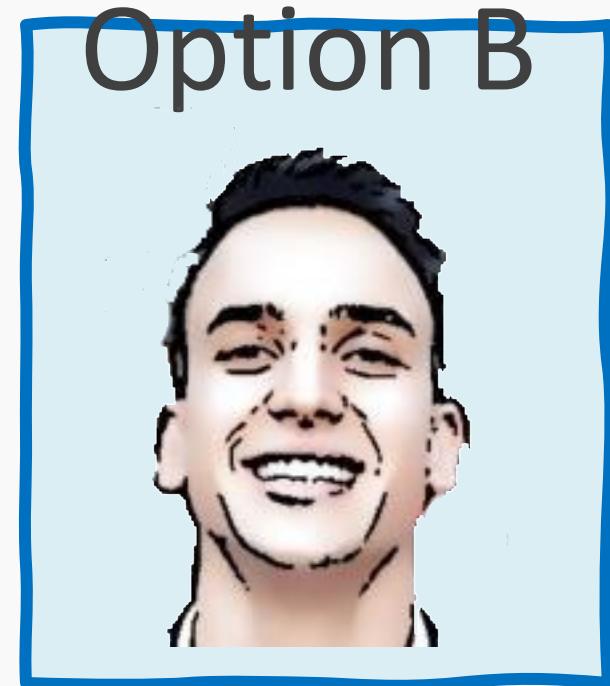
v/s



Random Variables Comparison - Point estimates



13.31	MAX	14.78
12.67	MIN	11.31
13.00	MEAN	13.00
13.01	MEDIAN	13.00
12.67	MODE	12.25



Measure of Central Tendency

Central Tendency

- Mean is the same as the ‘average’ that we are used to. If we know all the outcomes of the population:

$$\text{Population mean, } \mu = \frac{\sum_i x_i}{n}$$

- Sample statistics are calculated in a manner which best approximates the population parameters.
- Sample mean is calculated like population mean:

$$\text{Sample mean, } \bar{x} = \frac{\sum_{i=1}^n x_i}{n}$$

Measure of Spread

Spread

- Standard deviation is a measure of how spread out the data is from the mean. Assuming all of population is known:

$$\text{Population std}(\sigma) : \sqrt{\frac{\sum(x_i - \mu)^2}{n}}$$

- Sample std has a slight correction term to population std:

Sample std is used as an estimate for the population std, using $n-1$ gives more accurate results.

$$\text{Sample std}(s) : \sqrt{\frac{\sum(x_i - \bar{x})^2}{n - 1}}$$

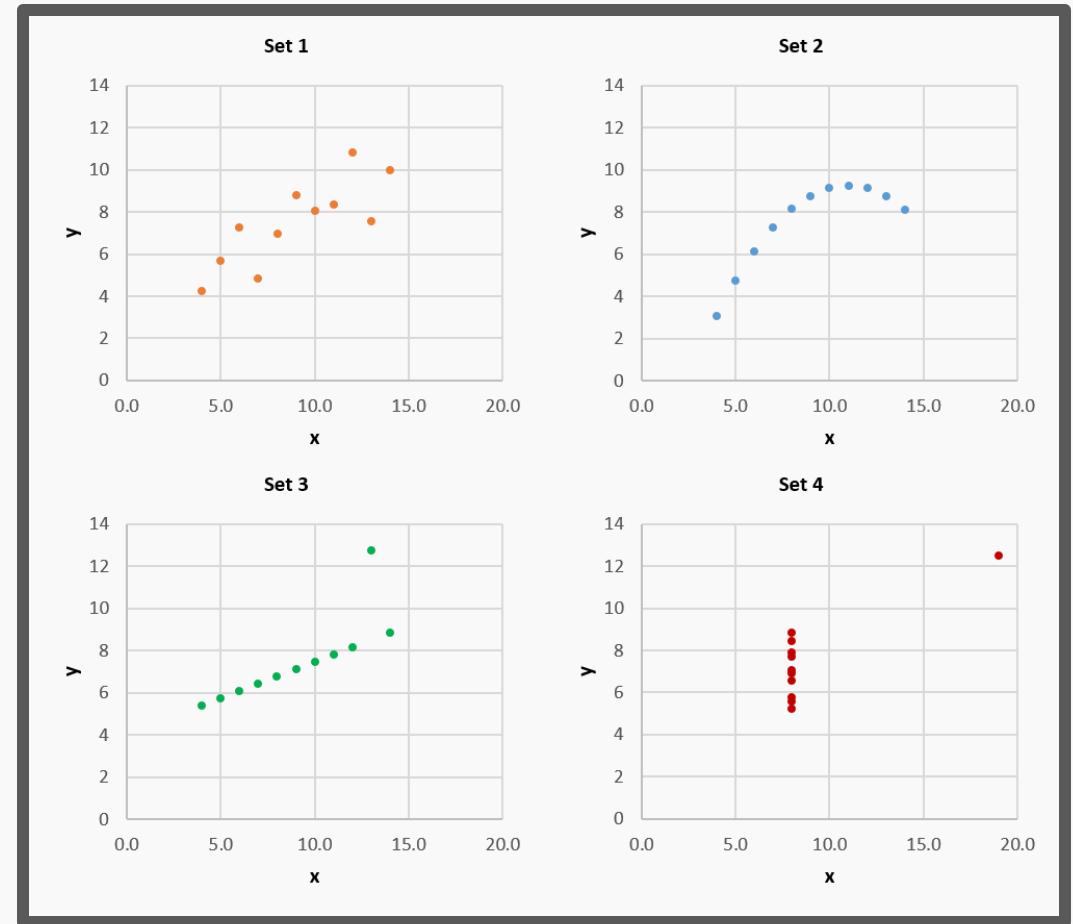
More on this at this [link](#)

Confidence Interval

Confidence Intervals

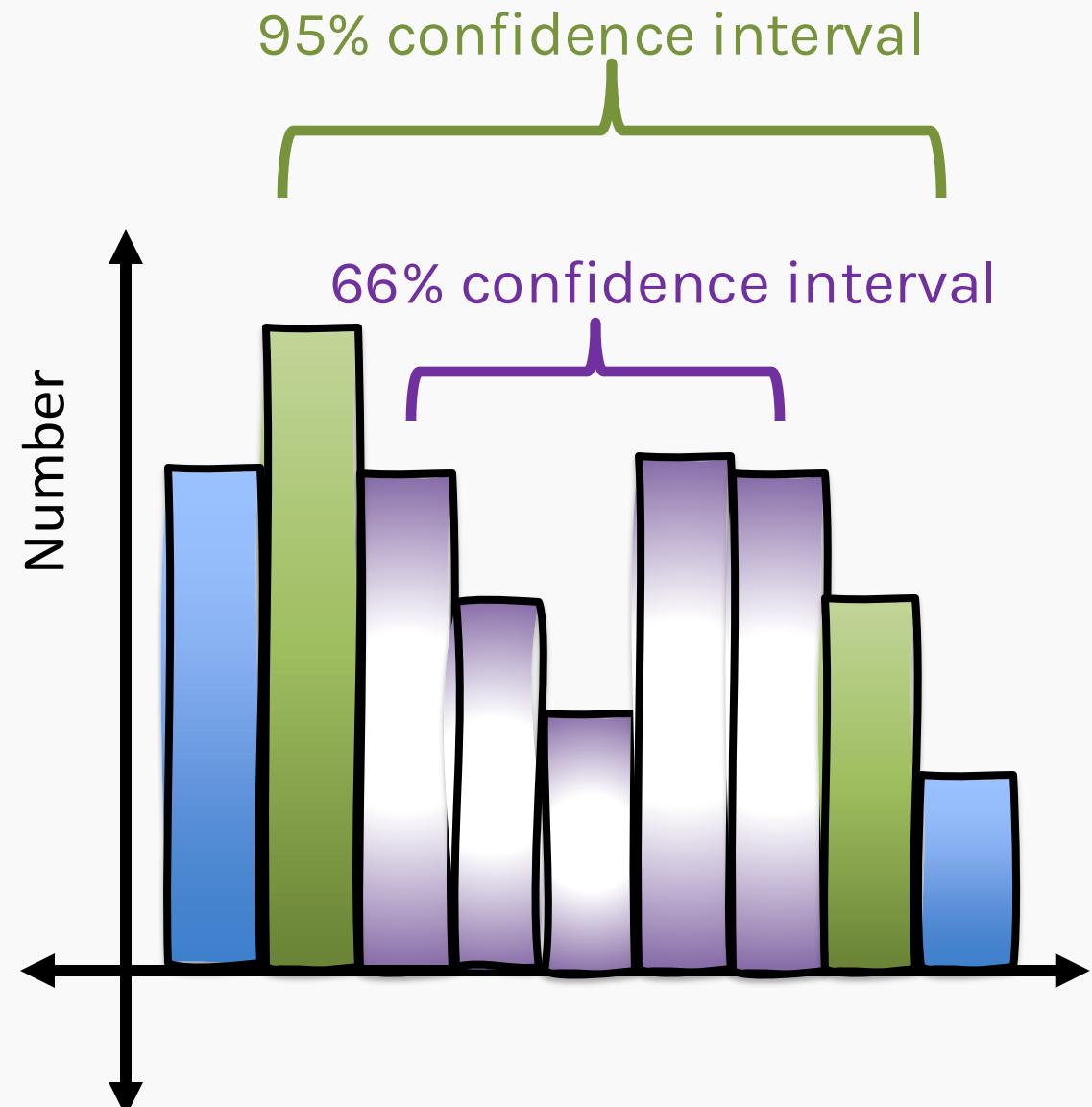
- Point estimates can often be misleading and lead to **imprecise** understanding of the random variable.

Anscombe's Quartet



Confidence Intervals

- Point estimates can often be misleading and lead to **imprecise** understanding of the random variable.
- Unlike point estimates, a confidence interval is a **range** that represents the likely output of a random experiment.
- We often set the **confidence level** before examining the data and it is expressed as %, e.g., 95% confidence



Confidence Intervals

[13.75, 15.21, 13.65, 13.58, 12.93, 14.23, 12.81, 11.50, 13.09, 12.26, ...]



Step #1: Sort the original data from lowest to highest

Confidence Intervals

[13.75, 15.21, 13.65, 13.58, 12.93, 14.23, 12.81, 11.50, 13.09, 12.26, ...]



Step #1: Sort the original data from lowest to highest

[11.50, 12.26 12.81, 12.93, 13.09, 13.58, 13.65, 13.75, 14.23, 15.21, ...]



Step #2: Find the lower confidence range using np.percentile

Confidence Intervals

[13.75, 15.21, 13.65, 13.58, 12.93, 14.23, 12.81, 11.50, 13.09, 12.26, ...]



Step #1: Sort the original data from lowest to highest

[11.50, 12.26 12.81, 12.93, 13.09, 13.58, 13.65, 13.75, 14.23, 15.21, ...]



Step #2: Find the lower confidence range using np.percentile

`np.percentile([11.50, 12.26 12.81, 12.93, 13.09, 13.58, 13.65, 13.75, 14.23, 15.21, ...], 2.5) = 12.80`

[11.50, 12.26 12.81, 12.93, 13.09, 13.58, 13.65, 13.75, 14.23, 15.21, ...]



2.5% of data are on the left of this value

Confidence Intervals

```
[ 11.50, 12.26 12.81, 12.93, 13.09, 13.58, 13.65, 13.75, 14.23, 15.21, ... ]
```



Step #3: Find the upper confidence range again using np.percentile

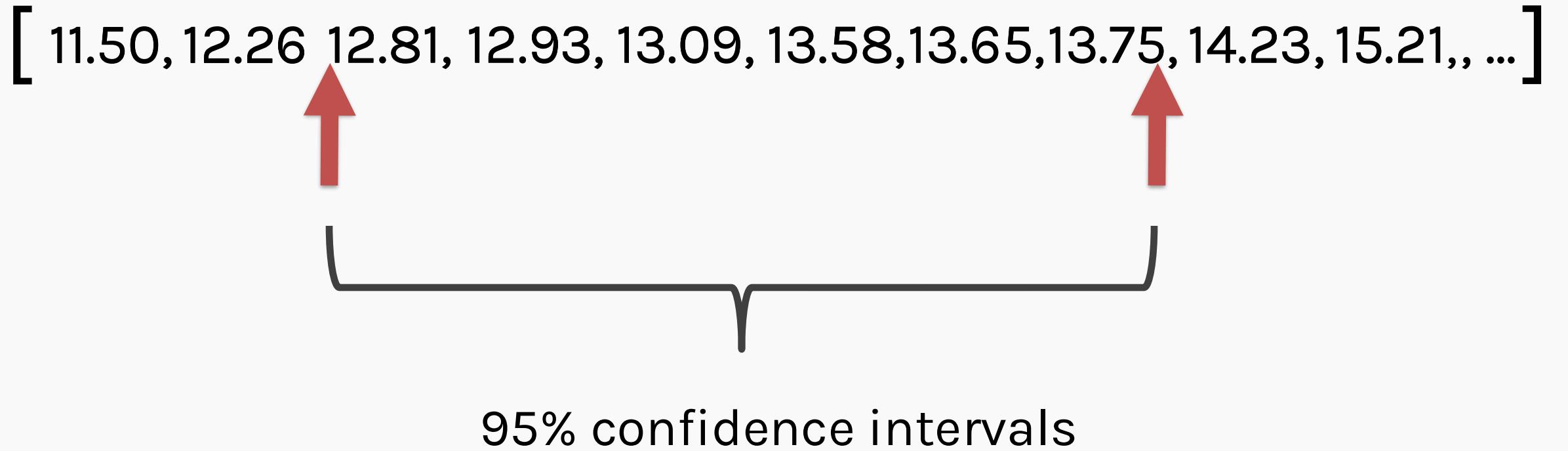
```
np.percentile( [ 11.50, 12.26 12.81, 12.93, 13.09, 13.58, 13.65, 13.75, 14.23, 15.21, ... ], 97.5 ) = 13.71
```

```
[ 11.50, 12.26 12.81, 12.93, 13.09, 13.58, 13.65, 13.75, 14.23, 15.21, ... ]
```

2.5% of data are on the right of this value



Confidence Intervals



Random Variables - Point estimates

Option A



13.31

MAX

14.78

12.67

MIN

11.31

13.00

MEAN

13.00

13.01

MEDIAN

13.00

12.67

MODE

12.25

12.80, 13.20 CI 12.80, 13.20

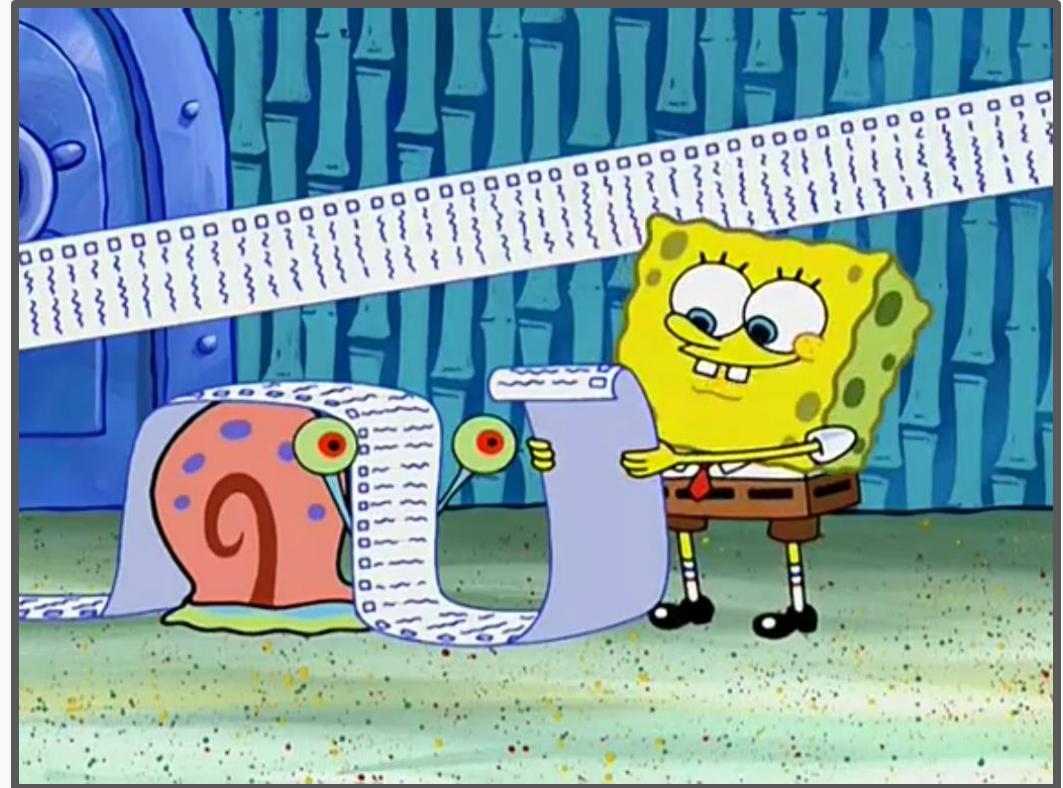
Option B



Properties of a Random Variable

ESTIMATE ISSUES

- Point or interval estimates of random variables do not guarantee a **unique** description of the output.
- Due to its **approximate** nature, it may lead to confounding of different processes.
- A popular example of this is the **Ansccombe's Quartet**; a set of four datasets with same estimates but different distributions.



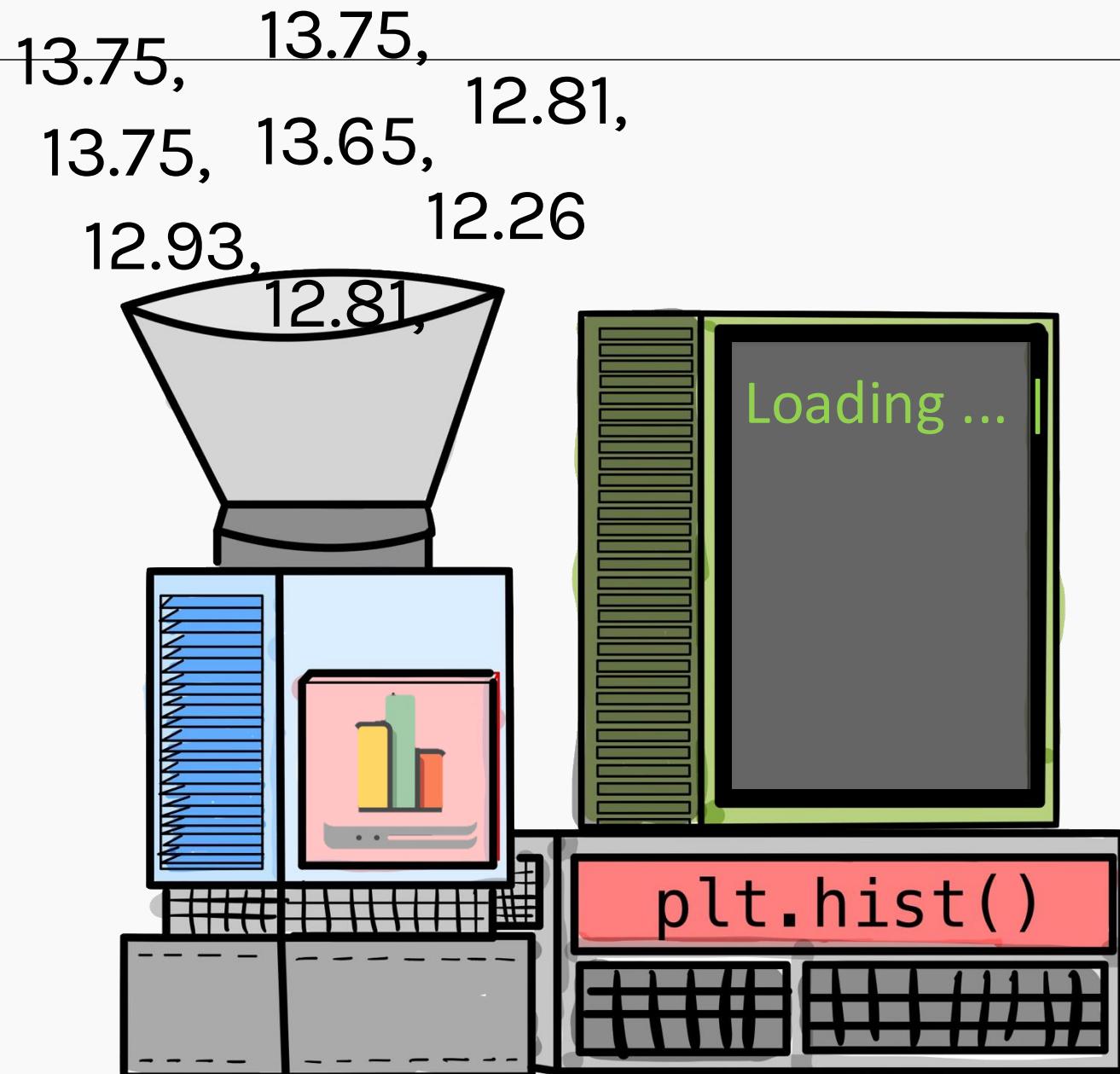
Histogram

Histogram

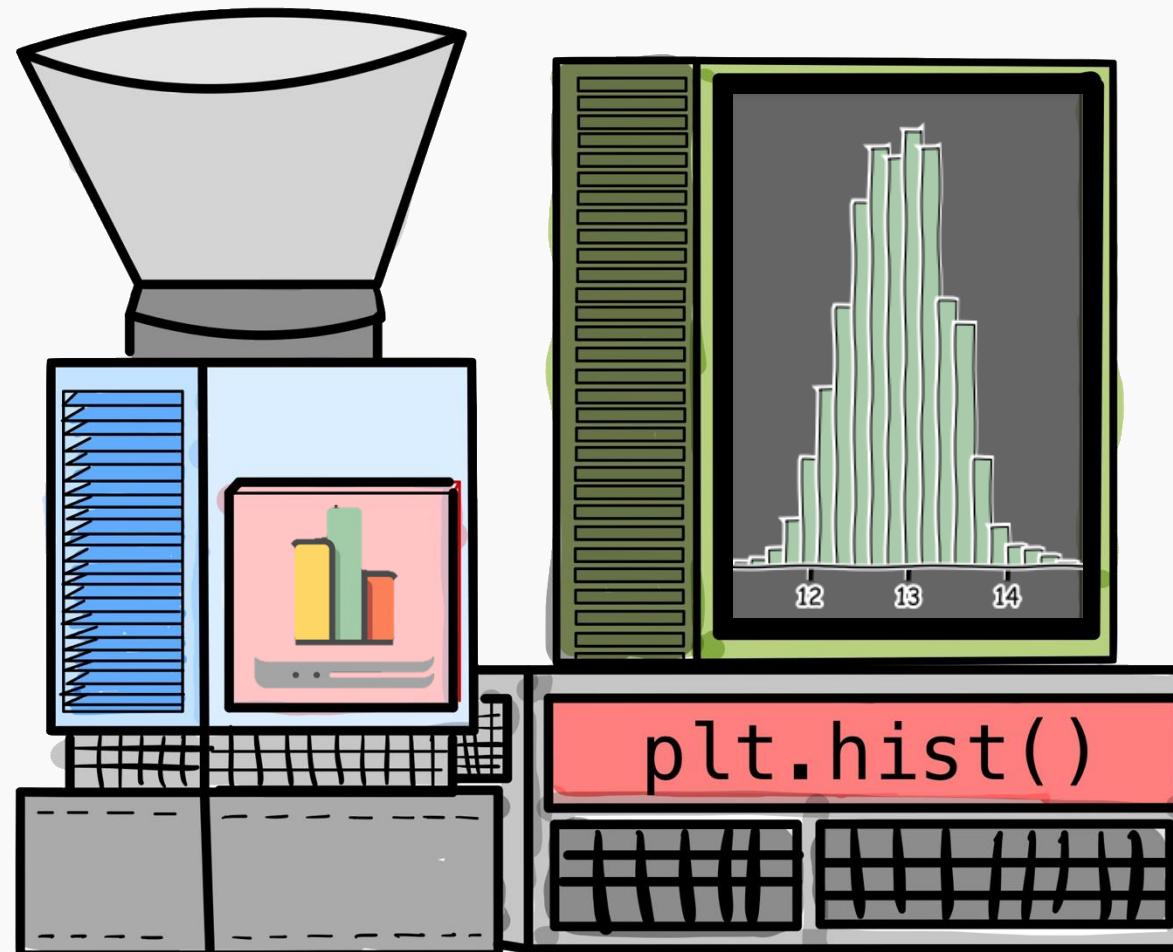
Sample

[13.75, 13.65, 12.93, 12.81, 12.26]

Histogram



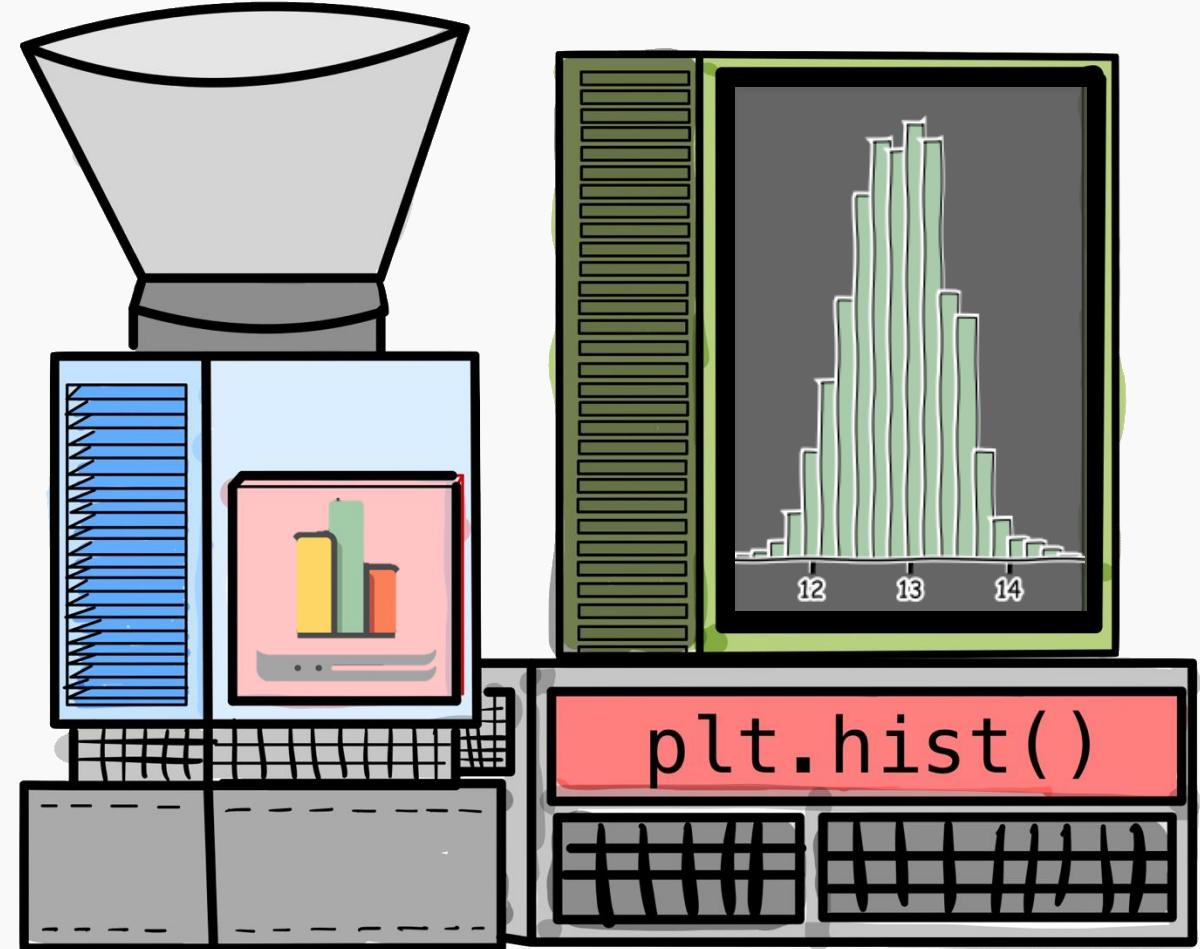
Histogram



Histogram

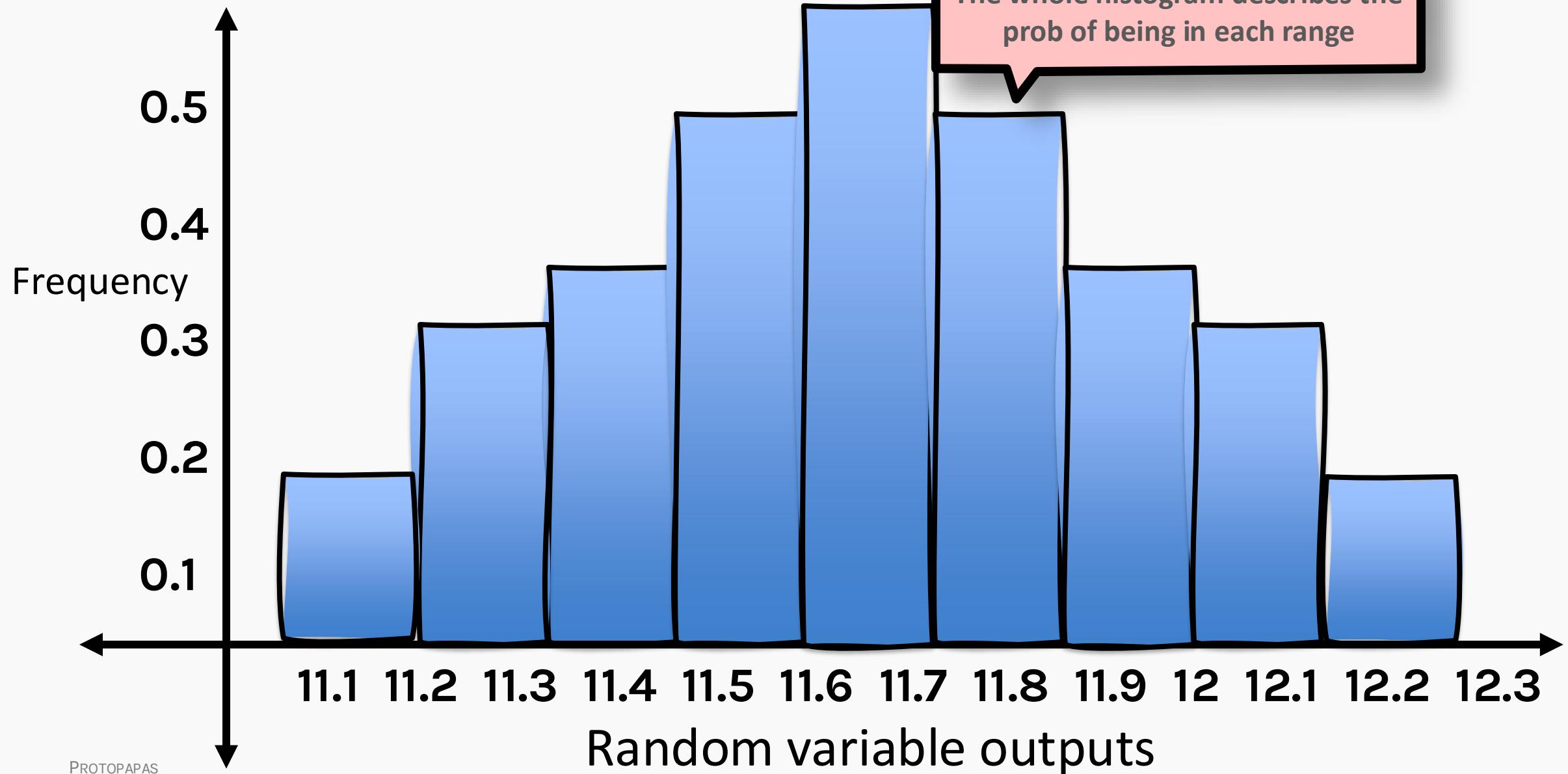
DISTRIBUTIONS

- Histogram (from the Greek word *histos* meaning pole & *gram* meaning chart) is a **visual representation** of the sample.
- It is defined by the **relative frequency** on the y-axis and the **outcomes** of the random variable on the x-axis.
- It can be decorated with point estimates for better description.

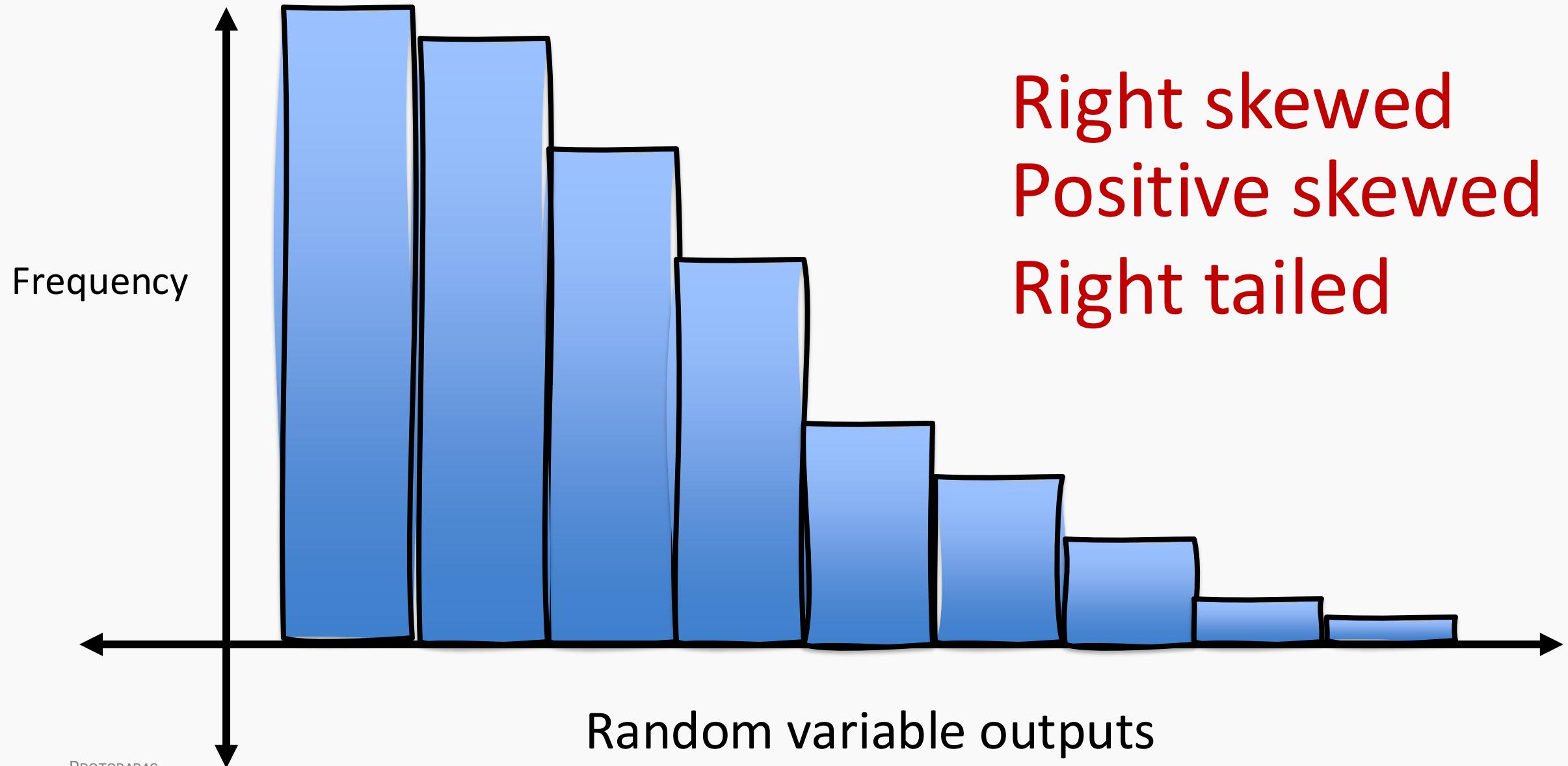


Anatomy of a histogram

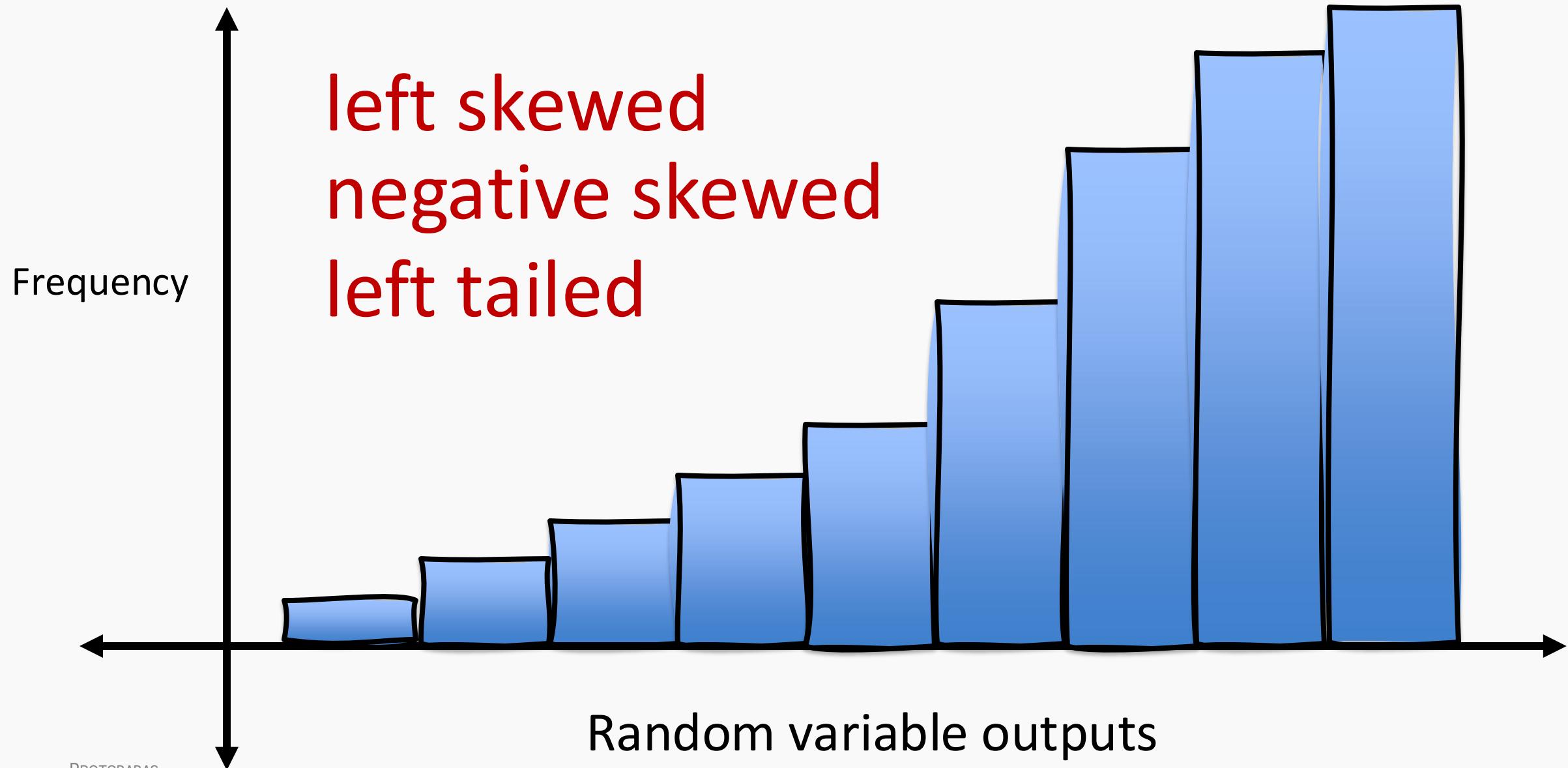
This bin describes the probability of the variable to be in this range. The whole histogram describes the prob of being in each range



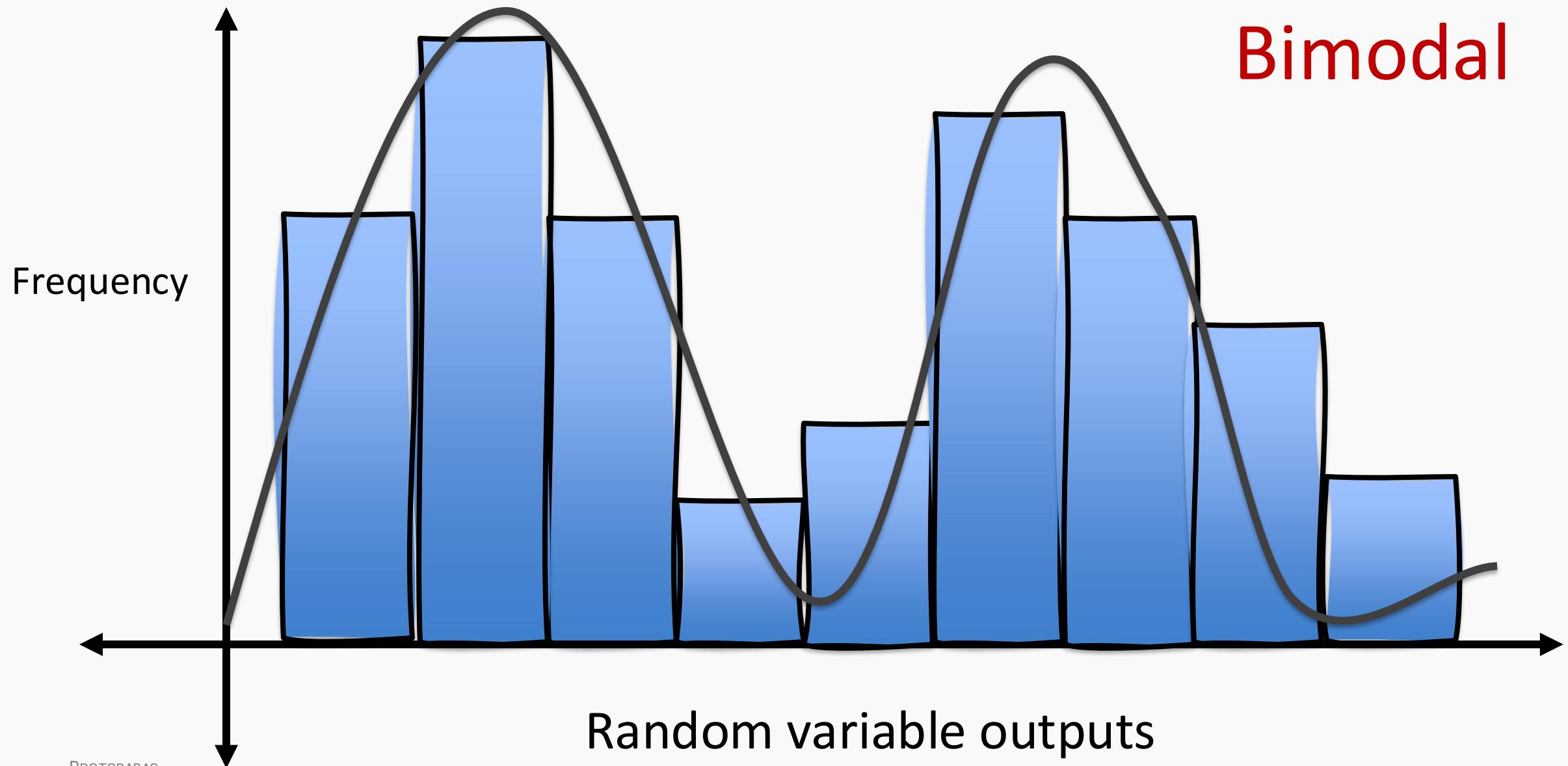
Anatomy of a histogram



Anatomy of a histogram



Anatomy of a histogram

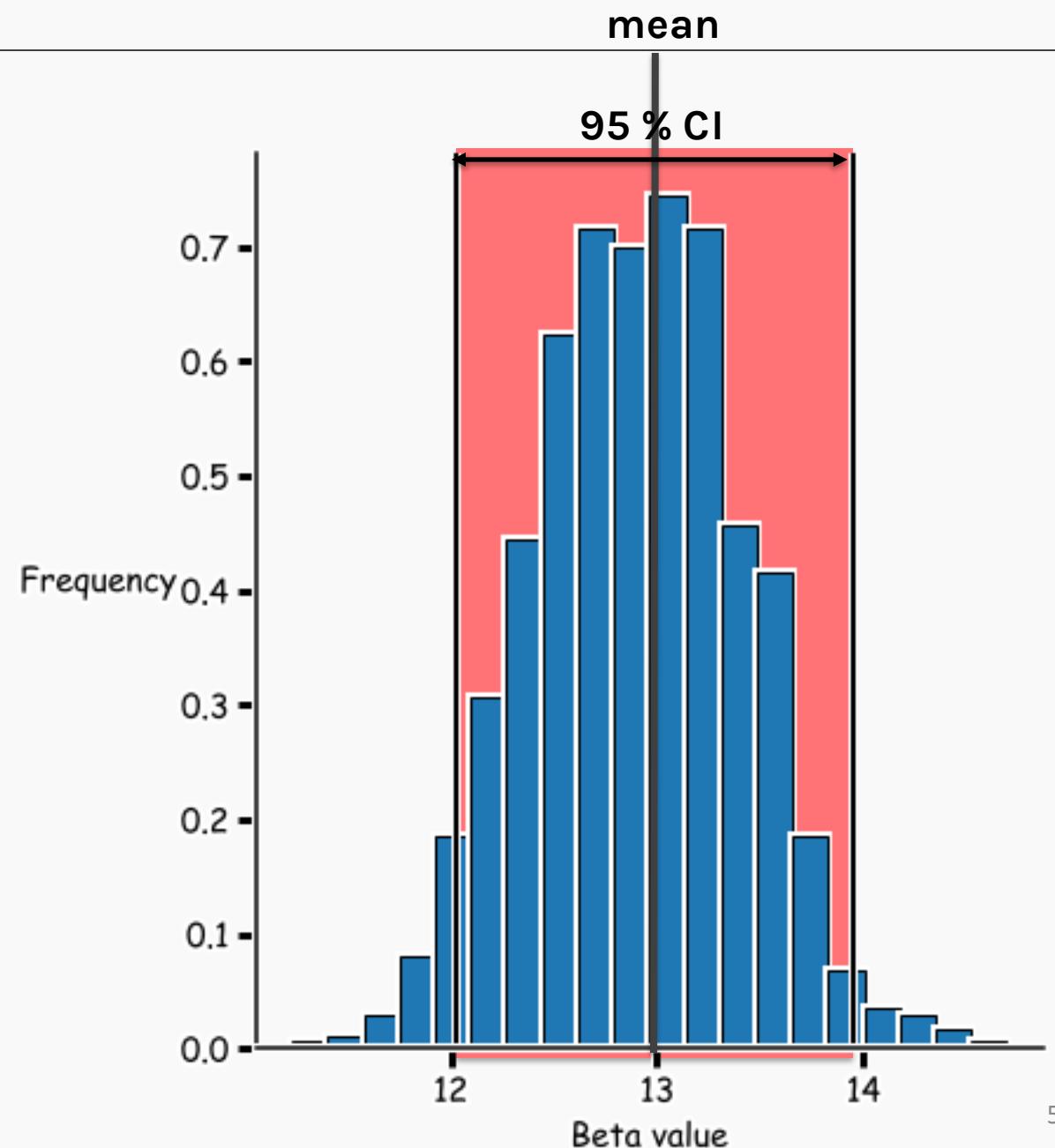


Histogram

ISSUES?

The outcomes of a random variable captured over multiple simulations is difficult to interpret and consequently difficult to compare to other random variables.

- **ISSUE #1:** We do not have estimates to compare with other random variables.
- **ISSUE #2:** It is difficult to visualize the spread of the outcome.



What is probability?

What is probability?

Q: What is probability?

A: A common definition: the long-run, relative frequency* of a random phenomenon/experiment/event.

Q: What values can probabilities take on?

A: Any value between 0 and 1 (including the endpoints).

Q: Why do we care?

A: Because data can be thought of as random realizations of a *data generating process* (whether though sampling or a theoretical construct).



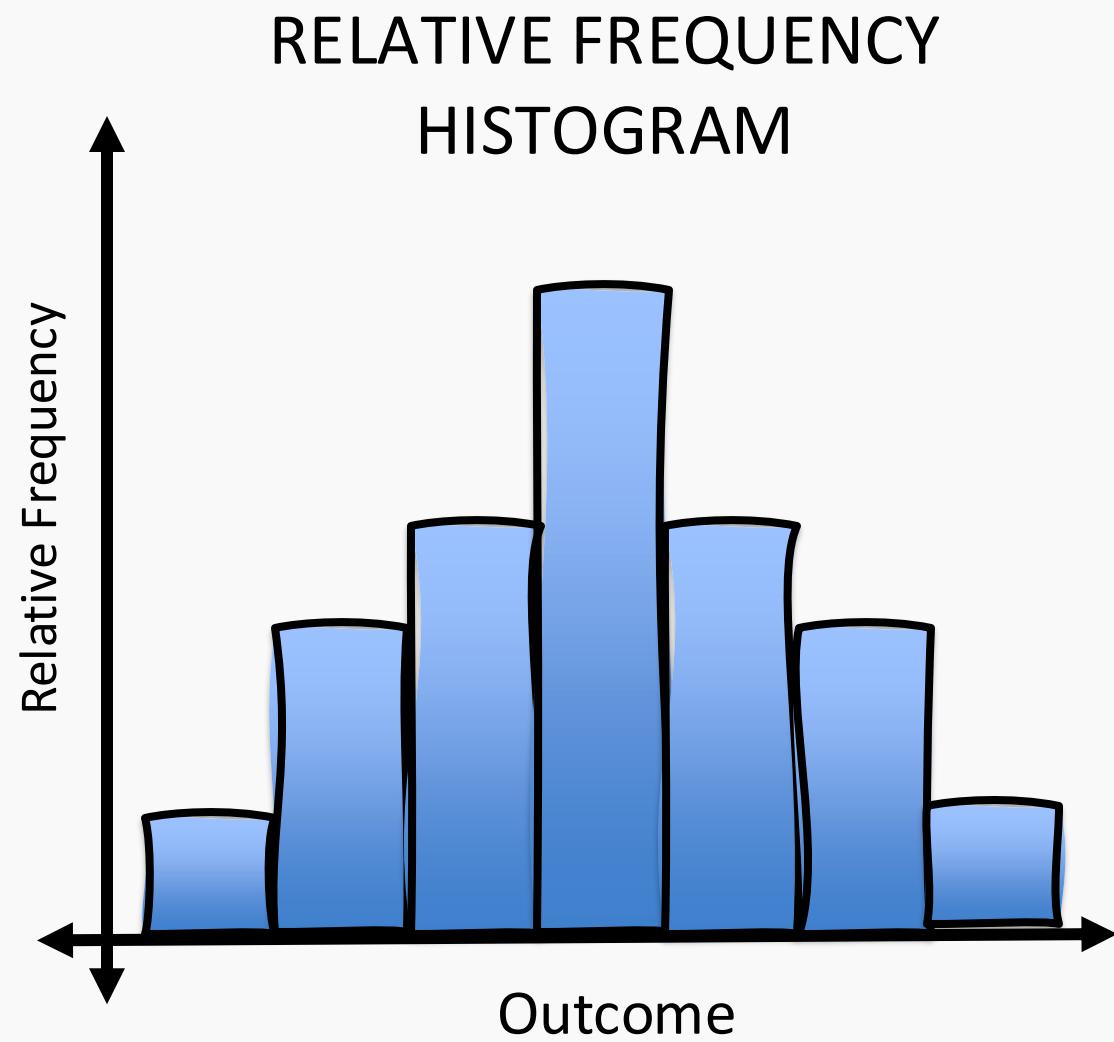
*Note: this ignores the Bayesian definition of probability: a measure of belief.

Known Distributions

Histogram as a Probability Density Function (PDF)

- Recall that a histogram describes the probability of being in a given range (**relative frequency** of the "bins").
- We can describe the probability of being a range with a function.
- This function could be defined for either discrete or continuous variables:
 - In case of continuous, this is the **probability density function (PDF)** for values in a range. This is usually written as $f(x)$.
 - In case of discrete, the range is the discrete value itself. The function is the **probability mass function (PMF)**, denoted as $P(X = x)$ or $p(x)$.

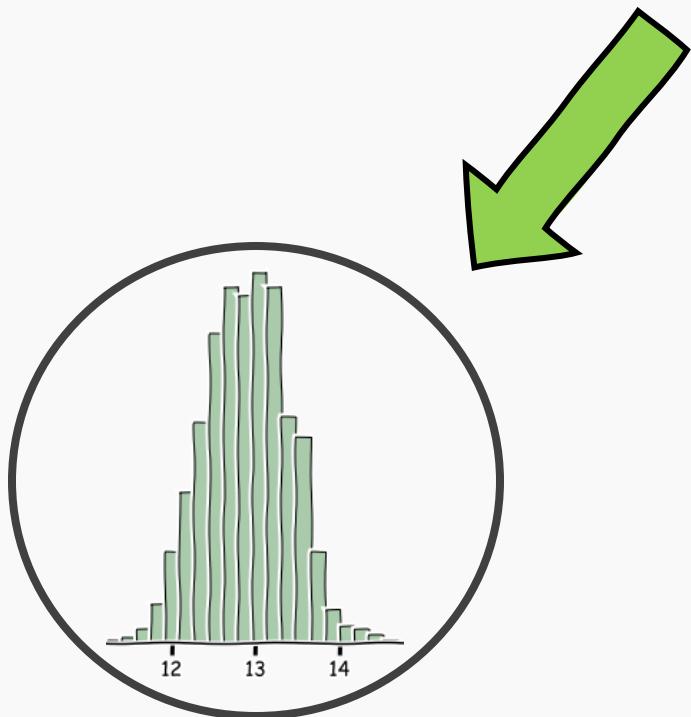
*Note: probabilities for a continuous random variable can be represented as areas under the curve, and thus $P(X = x) = 0$ since there is no width.



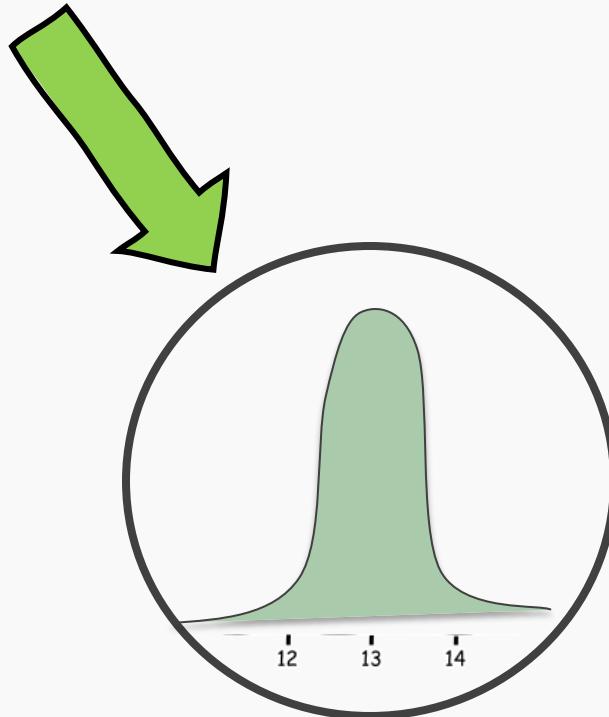
PDF vs PMF

Random Variable

X



Discrete Random Variable

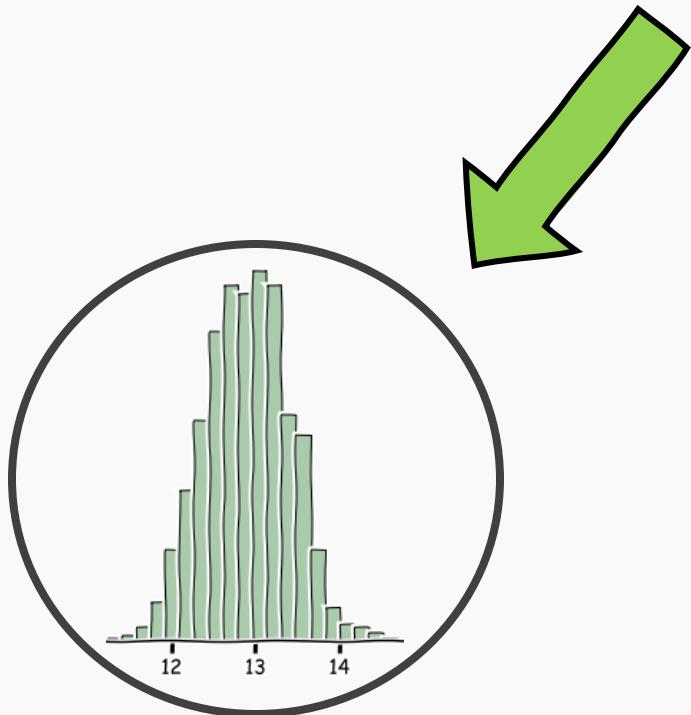


Continuous Random Variable

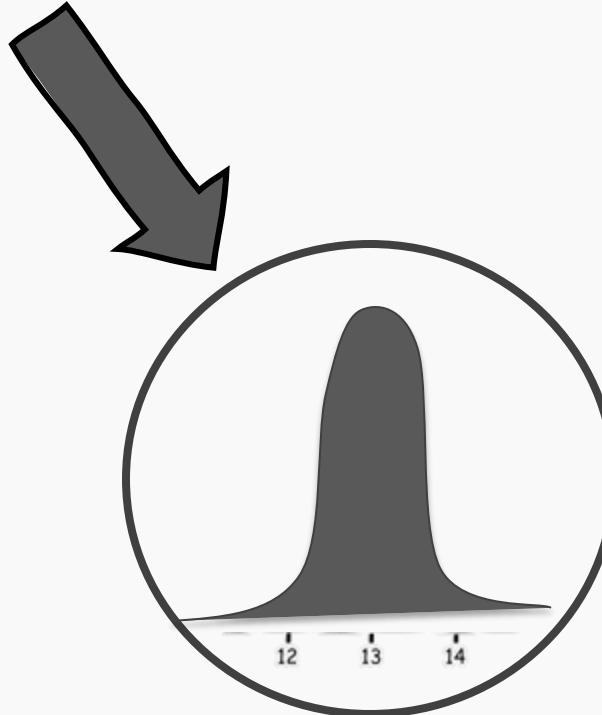
PDF vs PMF

Random Variable

X



Discrete Random Variable



Continuous Random Variable

Discrete Uniform Distribution

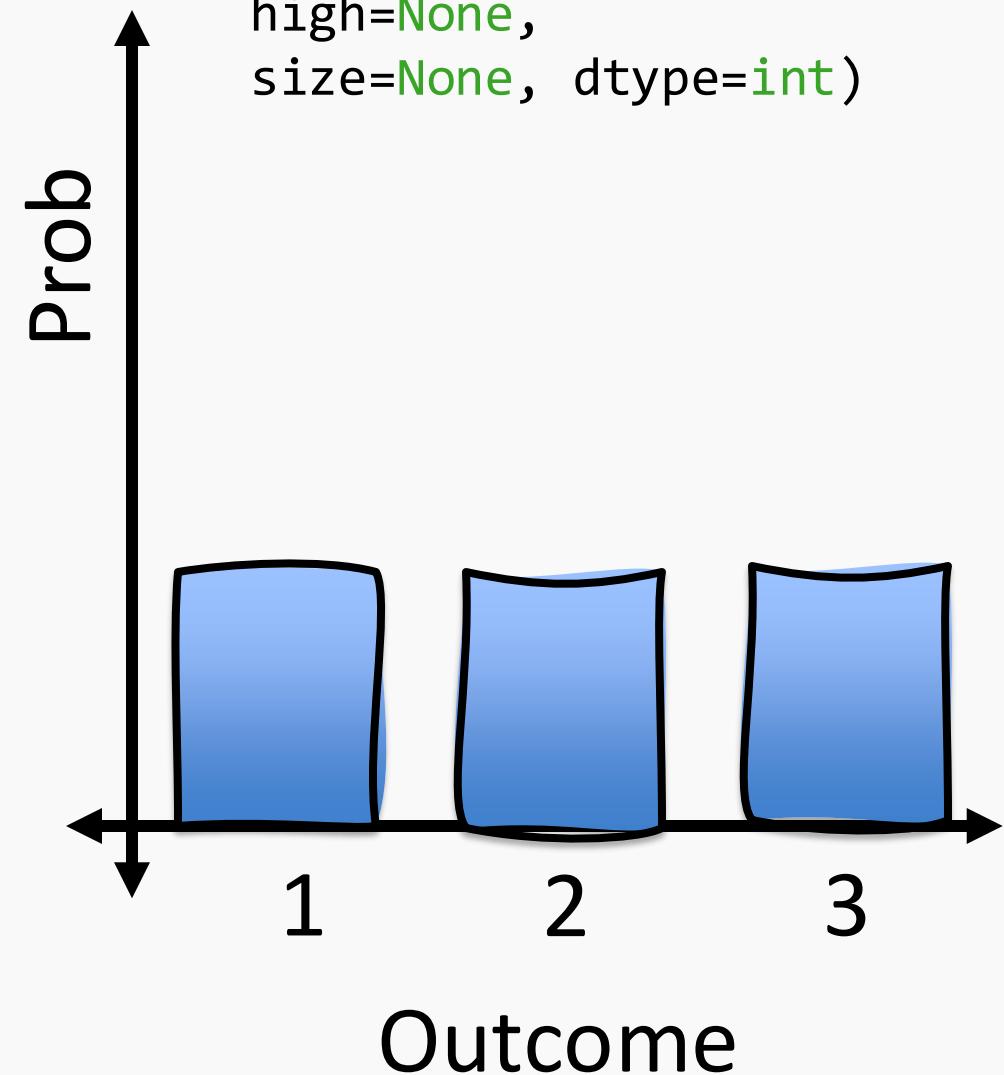
- This distribution occurs when there are a **finite** number of equally likely outcomes possible.

$$\text{PMF: } P(X = 1) = \frac{1}{N}$$

mean $\mu = \frac{a+b}{2}$, where a and b should be the first and last outcomes in the range

$$\text{Variance } \sigma^2 = \frac{(b-a+1)^2 - 1}{12}$$

`np.random.randint(low, high=None, size=None, dtype=int)`



Bernoulli Distribution

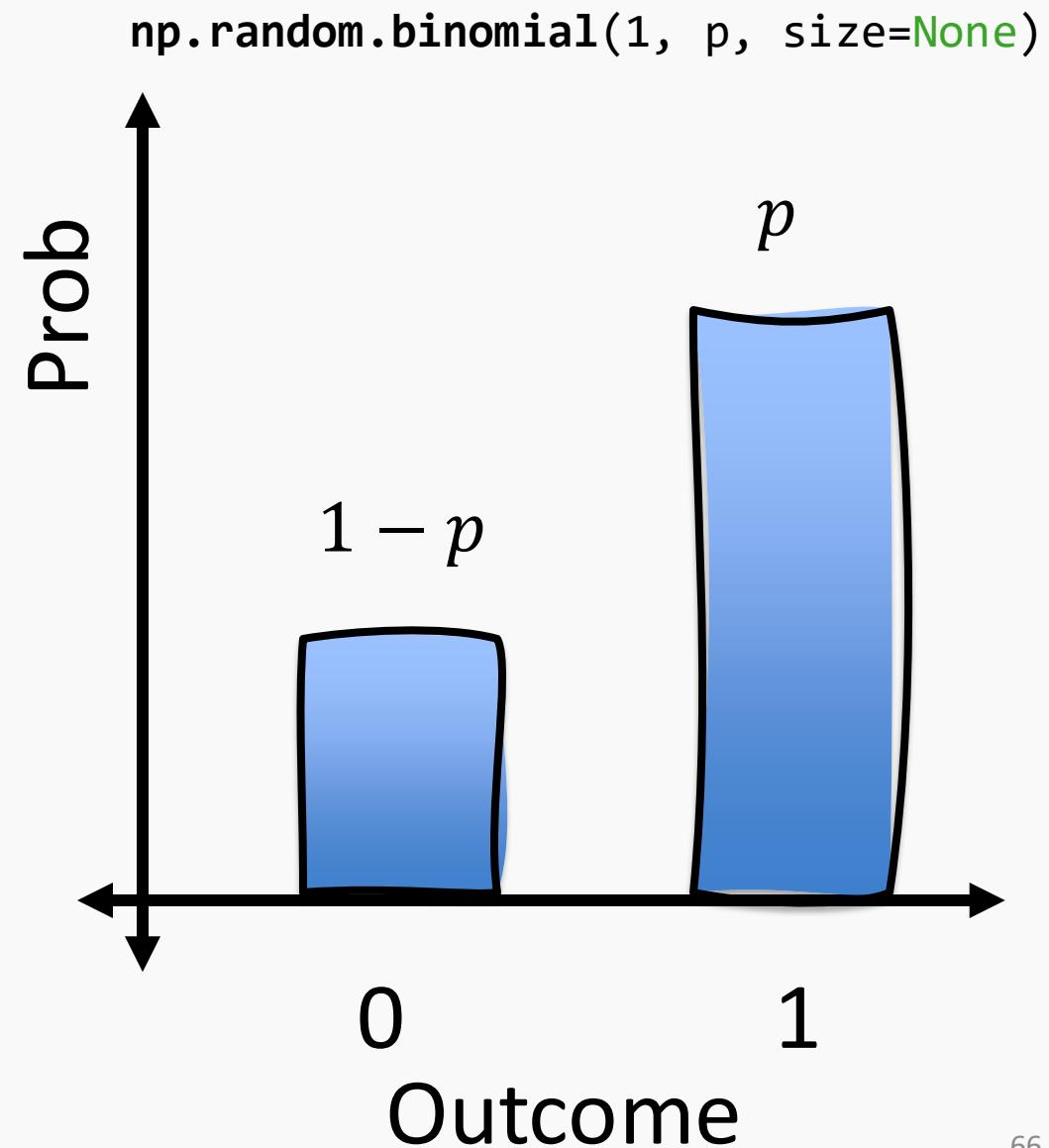
- This distribution can be thought of as a model of possible outcomes of an experiment that asks a *yes-no* question.
- E.g., If you toss a coin, will you get a *head* or a *tail*?

$$\text{PMF: } P(X = x) = p^x(1 - p)^{1-x}$$

where p is the probability of success and $q = 1 - p$ is the probability of failure.

$$\text{mean } \mu = p$$

$$\text{Variance } \sigma^2 = pq$$



Binomial Distribution

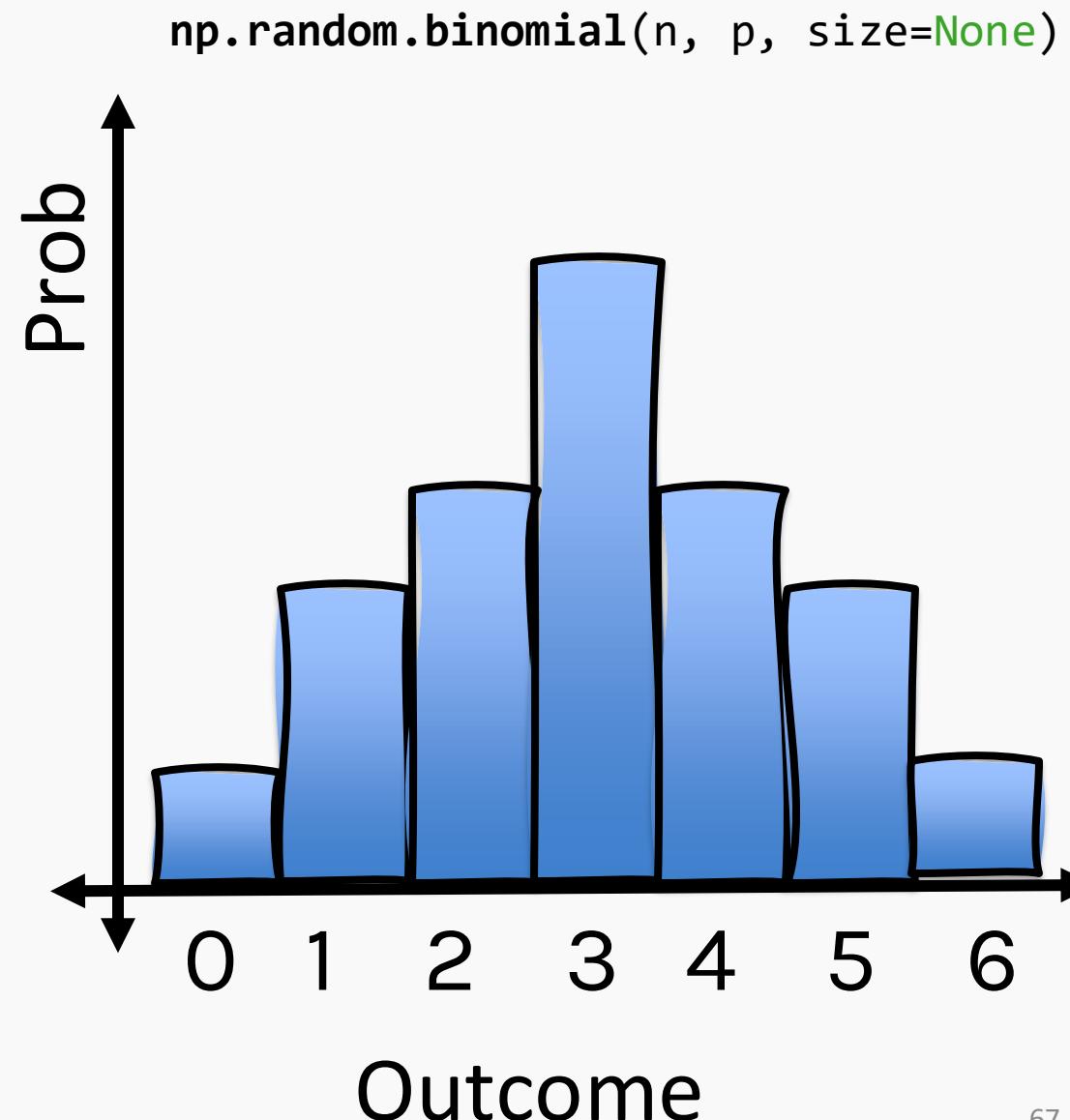
- A binomial distribution with parameters n and p is the distribution of the number of k **successes** in a sequence of n independent experiments, each asking a **yes-no** questions. This is often written as:
 $X \sim \text{Binom}(n, p)$.

$$\text{PMF: } P(X = x) = \binom{n}{k} p^k q^{n-k}$$

where p is the probability of success and $q = 1 - p$ is the probability of failure.

$$\text{mean } \mu = np$$

$$\text{variance } \sigma^2 = npq$$



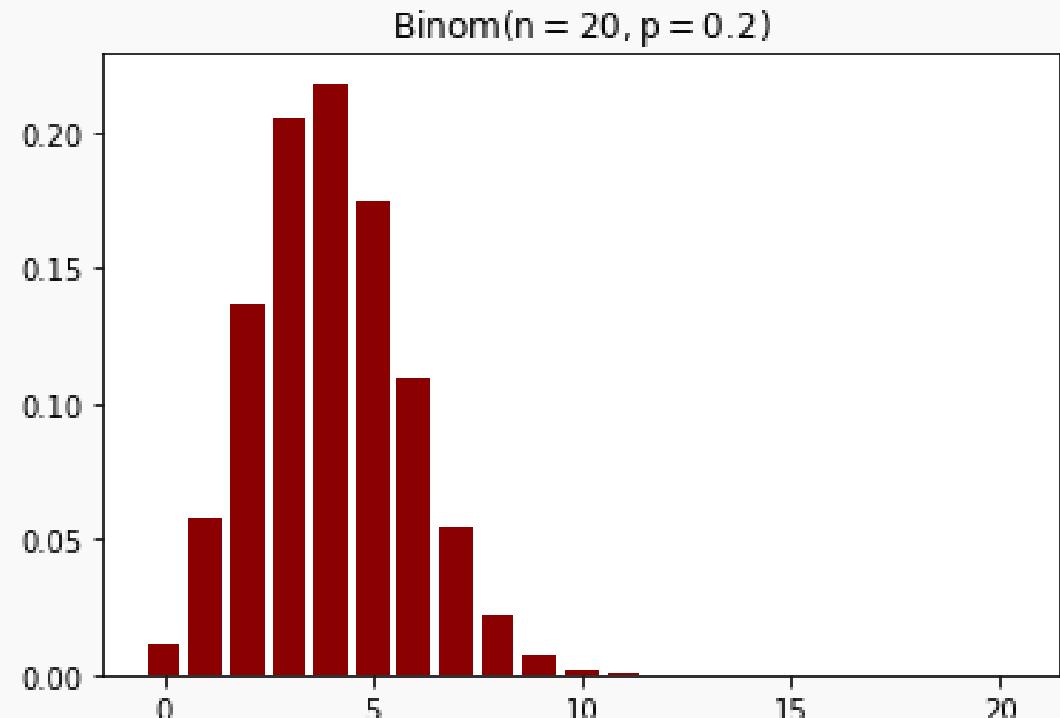
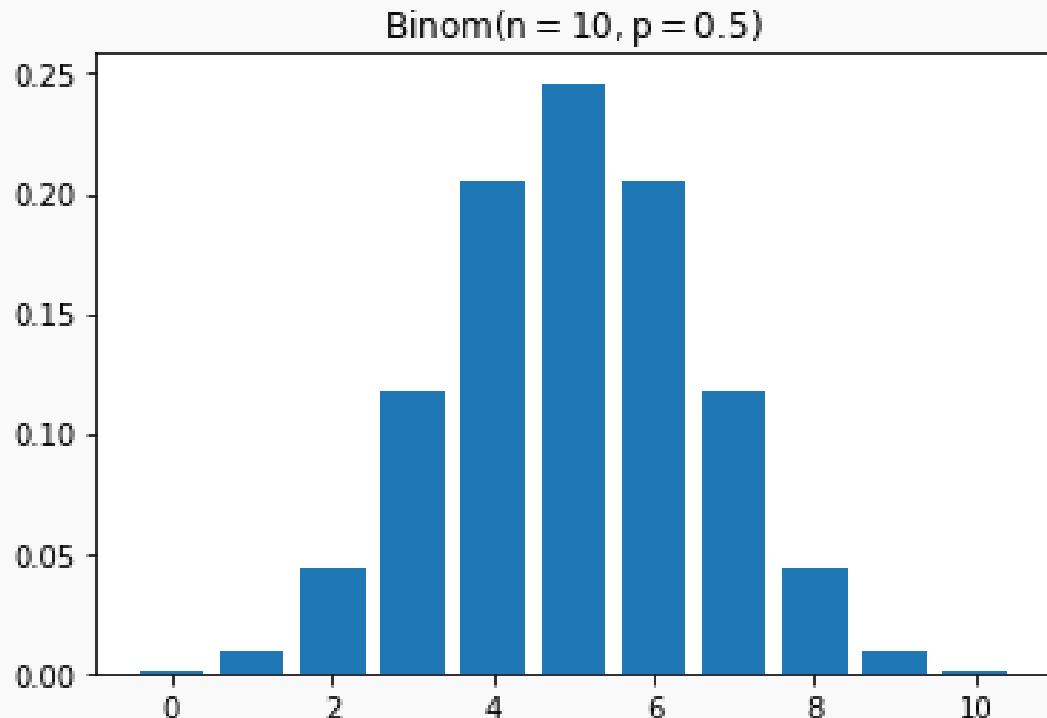
Binomial Distribution

Think counting the number of heads when flipping a biased coin n times.

The binomial distribution is useful to describe polling data (proportion of people who will vote for Biden), survey data (will you take CS1090B next year?), or any data that are binary!

The **Bernoulli distribution** is a special case when $n = 1$.

Binomial Distribution Examples



A binomial distribution has mean np and standard deviation $\sqrt{np(1 - p)}$.

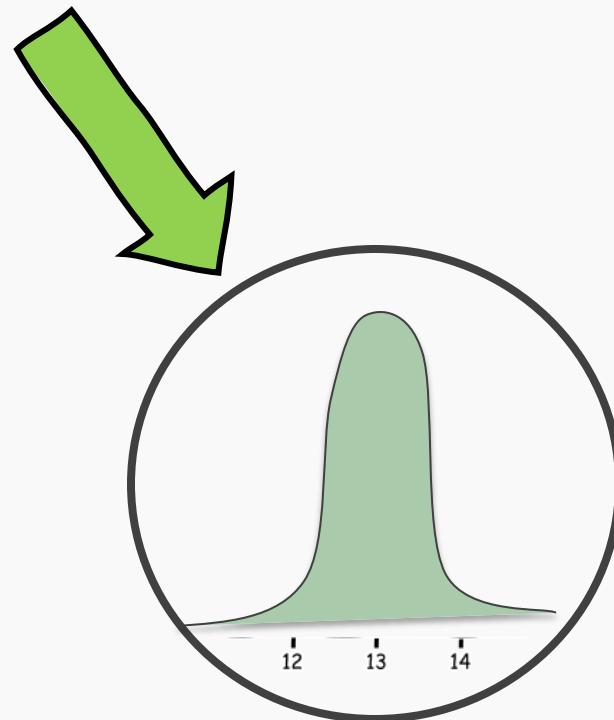
PDF vs PMF

Random Variable

X



Discrete Random Variable



Continuous Random Variable

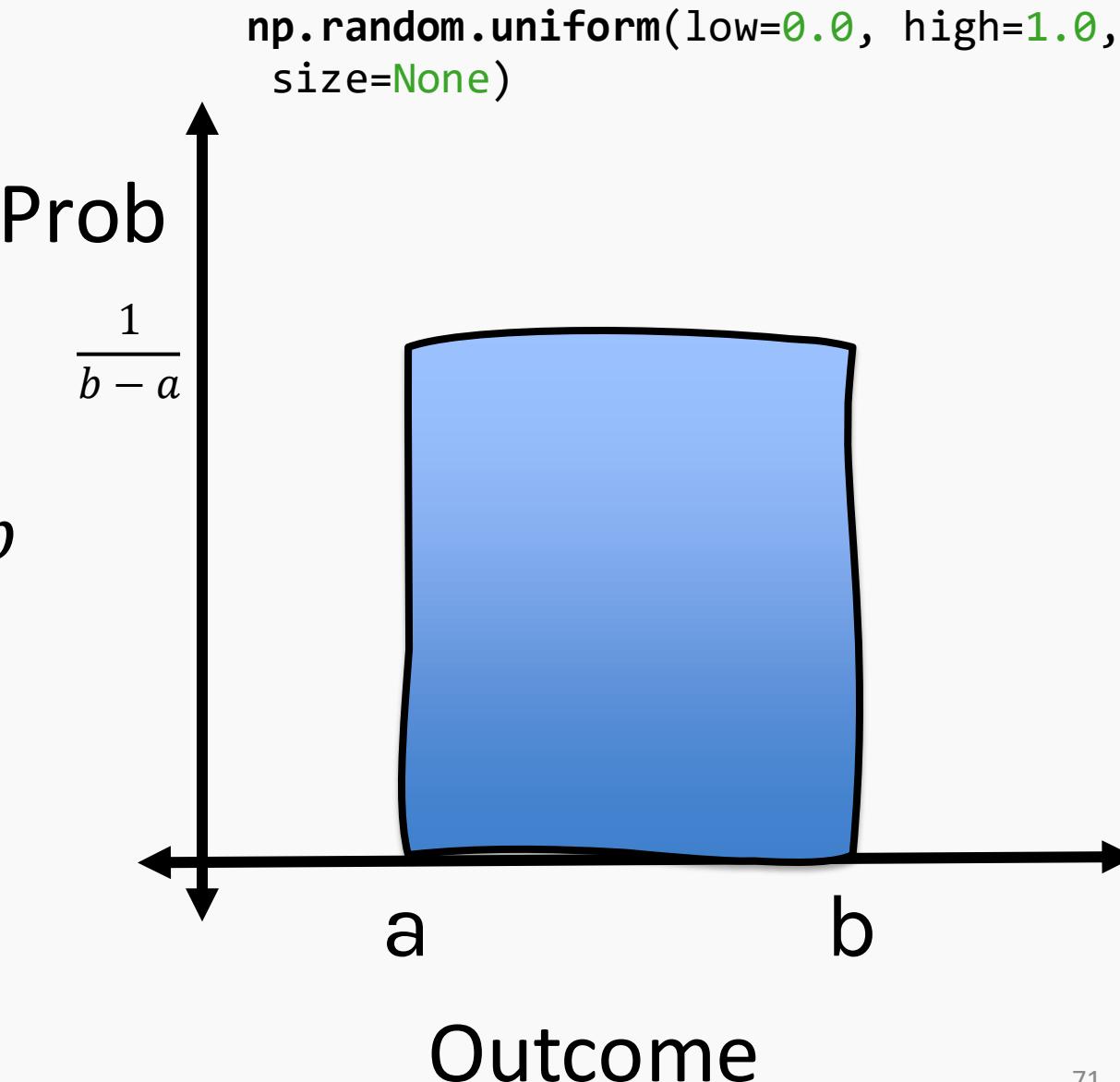
Uniform continuous distribution

- This distribution describes an experiment where there is an arbitrary outcome that lies between certain **bounds**, defined by parameters a and b .

$$\text{PDF: } P(X = x) = \begin{cases} \frac{1}{b-a} & \text{for } a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

$$\text{mean } \mu = \frac{a+b}{2}$$

$$\text{Variance } \sigma^2 = \frac{(b-a)^2}{12}$$



Normal distribution

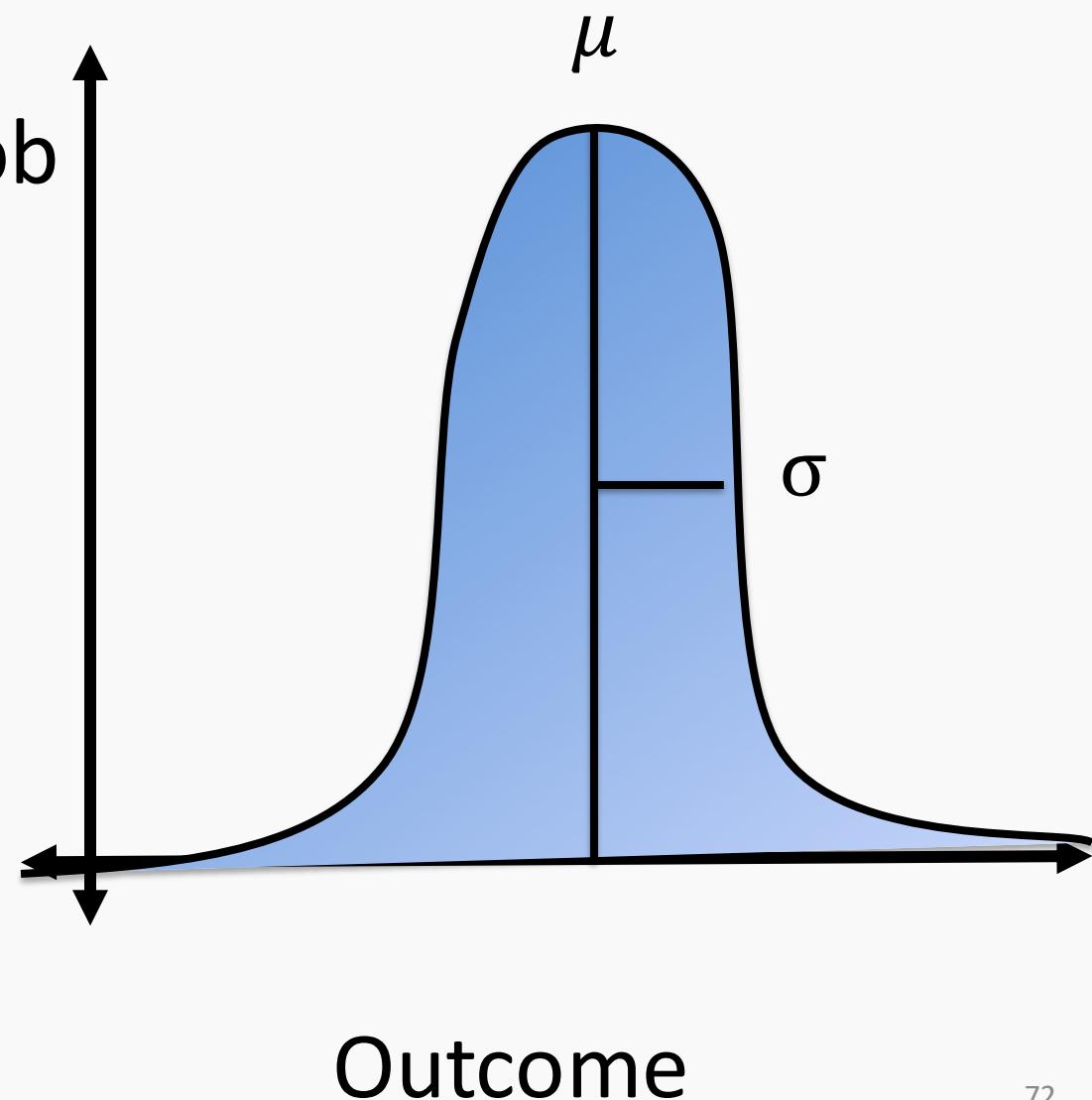
- A normal (or Gaussian) distribution is one of the most used continuous random variables.
- As a result of the [Central Limit Theorem](#), the distribution of sample means from a sufficiently large sample size approximates a normal distribution, regardless of the original population distribution.

$$\text{PDF: } P(X = x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

mean $E[X] = \mu$ (location)

variance $E[(X - \mu)^2] = \sigma^2$ (scale)

```
np.random.normal(loc=0.0, scale=1.0,  
size=None)  
np.random.randn()
```



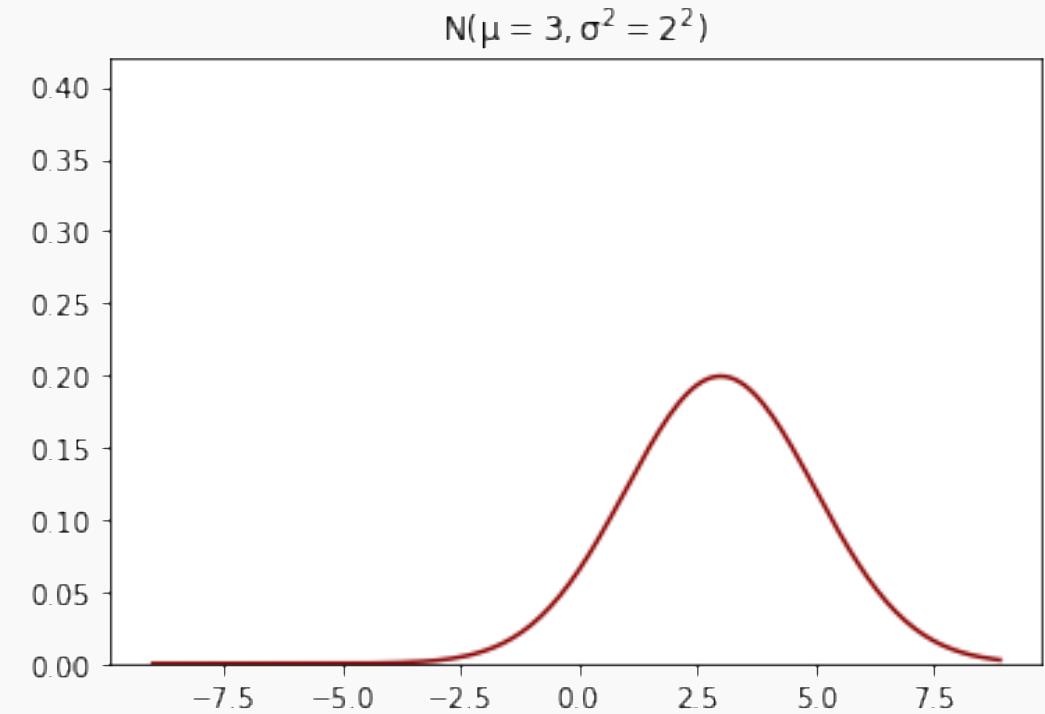
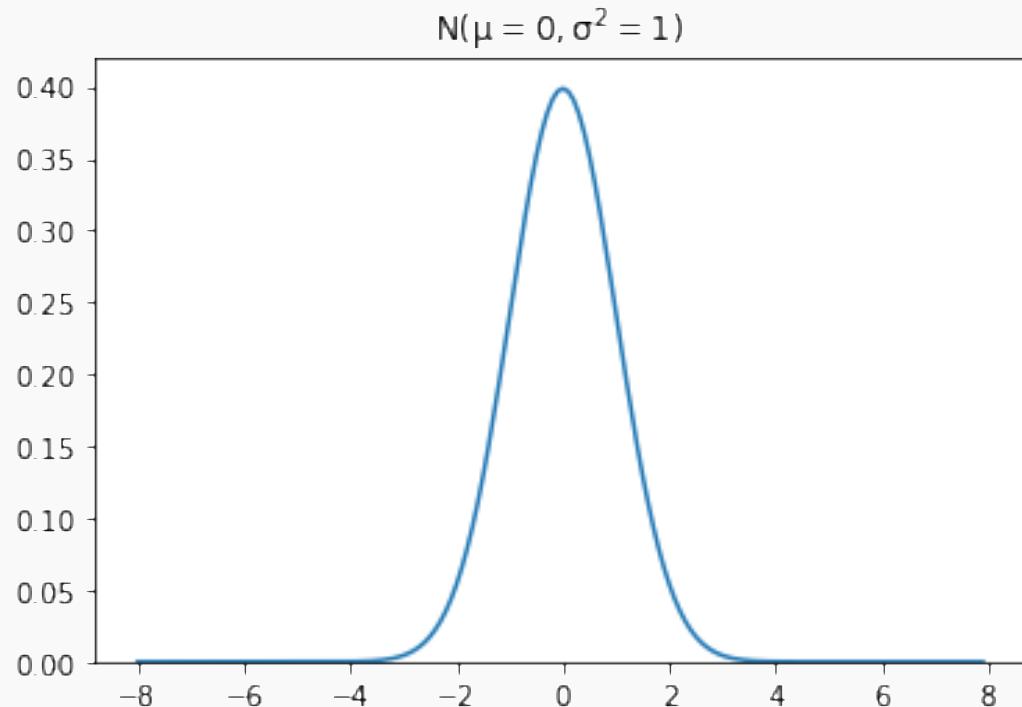
Normal Distribution

The normal distribution (sometimes called the Gaussian) is often referred to as the bell-shaped curve. But the normal distribution isn't the only one that is bell-shaped: *t* distributions are also bell-shaped, for example.

The standard normal distribution is a special case: $Z \sim N(0,1)$.

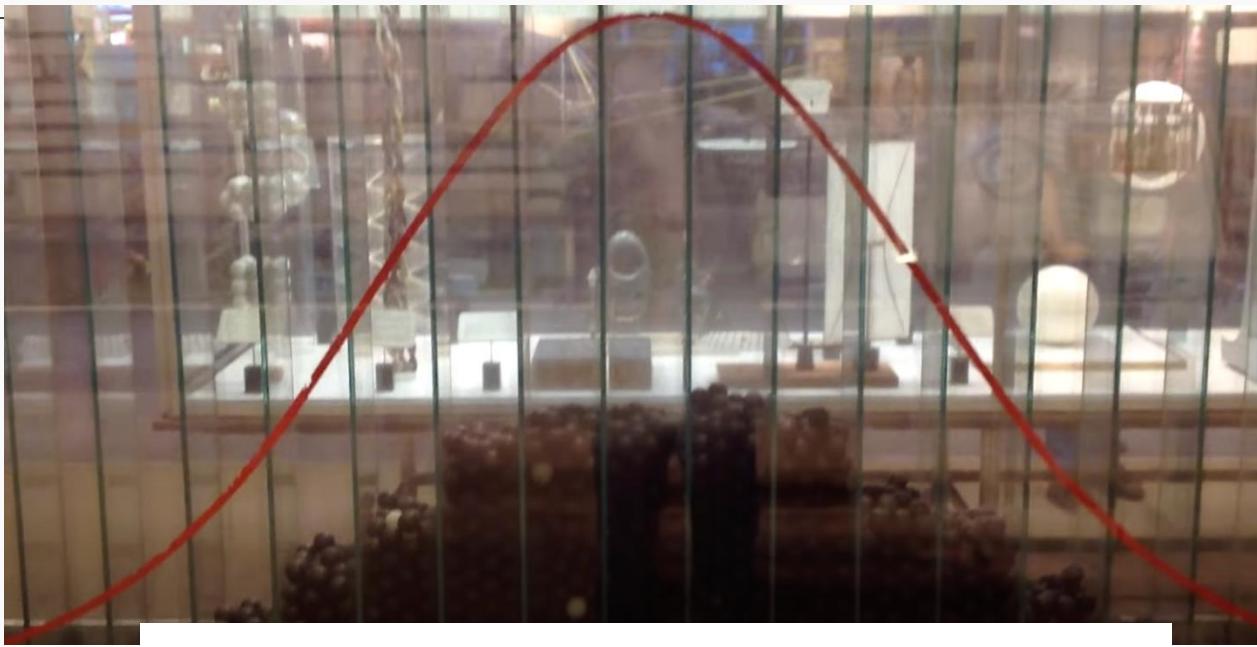
Any normal random variable can be standardized using the formula $Z = \frac{X-\mu}{\sigma}$.

Normal Distribution Examples

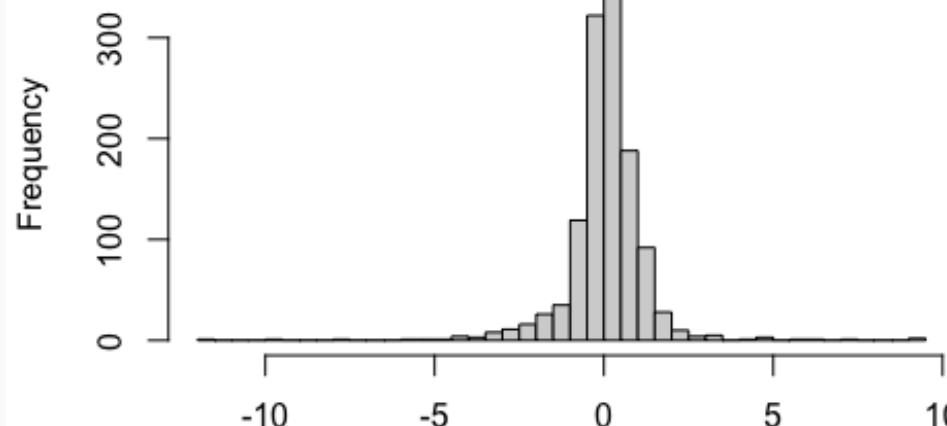


A normal distribution has mean μ and standard deviation σ .

I See Normal Distributions



Daily % Change in SP500 (5+ years)



PROTOPAPAS



75

Central Limit Theorem



Why is the normal distribution used so often?

The **Central Limit Theorem**: random variables that are averages or sums of many other random variables will be approximately normally distributed.

More specifically: if X_1, X_2, \dots, X_n are independent random variables (representing individual observations of data) with mean μ and standard deviation σ (not necessarily normal themselves), then the sample mean

$$\bar{X} = \frac{X_1 + X_2 + \cdots + X_n}{n}$$

will have approximate distribution:

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

Joint Distributions

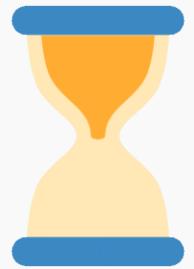


Joint Distributions

What happens to these probability distributions (PMFs and PDFs) when there are multiple random variables involved (aka, multiple observations in a data set)?

Let $f(x_1, x_2, \dots, x_n)$ be the **joint distribution** of n separate random variables. If they all come from the same generative marginal distribution, $f(x_i)$, and are independent, what is the resulting distribution?

$$f(x_1, x_2, \dots, x_n) = f(x_1) \cdot f(x_2) \cdots f(x_n) = \prod_{i=1}^n f(x_i)$$



Digestion Time

GIVEN THE PACE OF
TECHNOLOGY, I PROPOSE
WE LEAVE MATH TO THE
MACHINES AND GO PLAY
OUTSIDE.



Please download and install the Slido app on all computers you use



CS109A: Because even Harvard needs more ways to count things

- ① Start presenting to display the poll results on this slide.

Please download and install the Slido app on all computers you use



CS109A: Where nerdy meets trendy!
  #DataScienceRocks

- ① Start presenting to display the poll results on this slide.

Please download and install the Slido app on all computers you use



CS109A: Where we turn data into gold... and occasionally into memes!

- ① Start presenting to display the poll results on this slide.

Modeling Data with Probability Distributions

Likelihood Theory

The idea of likelihood

The **likelihood** tells us:

Given the model, what is the likelihood of observing this data?

Instead of writing this function with the data (X) as the unknown, we use the same function but uses the parameter(s) as the unknown(s).

Given observed data, what values of the model's parameters are likely?

Modeling Linear Regression Probabilistically

The Simple Linear Regression Model

We've defined the linear regression model to predict the i -th observation's response, Y_i , from a predictor, X_i , to be:

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

For any random variable, ϵ , that has zero mean

$$E(Y_i) = \beta_0 + \beta_1 X_i$$

The error term, ϵ_i , represents the distance the observation lies from the line in the vertical direction of Y .

The Probabilistic Regression Model

If we assume that $\epsilon_i \sim N(0, \sigma^2)$

This regression model can be rewritten as:

$$Y_i | X_i \sim N(\beta_0 + \beta_1 X_i, \sigma^2)$$

The likelihood of a measurement having value Y_i given X_i for a model β_0, β_1

$$L(\beta_0, \beta_1, \sigma^2 | Y_i, X_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(Y_i - (\beta_0 + \beta_1 X_i))^2}{2\sigma^2}}$$

The Probabilistic Regression Model

The likelihood of a measurement having value Y_i given X_i for a model β_0, β_1

$$L(\beta_0, \beta_1, \sigma^2 | Y_i, X_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(Y_i - (\beta_0 + \beta_1 X_i))^2}{2\sigma^2}}$$

This formulation allows us to write out the joint likelihood function for this probability model.

The joint likelihood function for this probability model becomes:

$$L(\beta_0, \beta_1, \sigma^2 | Y, X) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(Y_i - (\beta_0 + \beta_1 X_i))^2}{2\sigma^2}}$$

Assumptions of Linear Regression

Normality of Residuals

Linearity

$$L(\beta_0, \beta_1, \sigma^2 | Y, X) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(Y_i - (\beta_0 + \beta_1 X_i))^2}{2\sigma^2}}$$

Independence

Homoscedasticity

The Likelihood of Linear Regression

The joint likelihood function for this probability model becomes:

$$L(\beta_0, \beta_1, \sigma^2 | Y, X) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(Y_i - (\beta_0 + \beta_1 X_i))^2}{2\sigma^2}}$$

Remember, we stated that the likelihood function tells us:

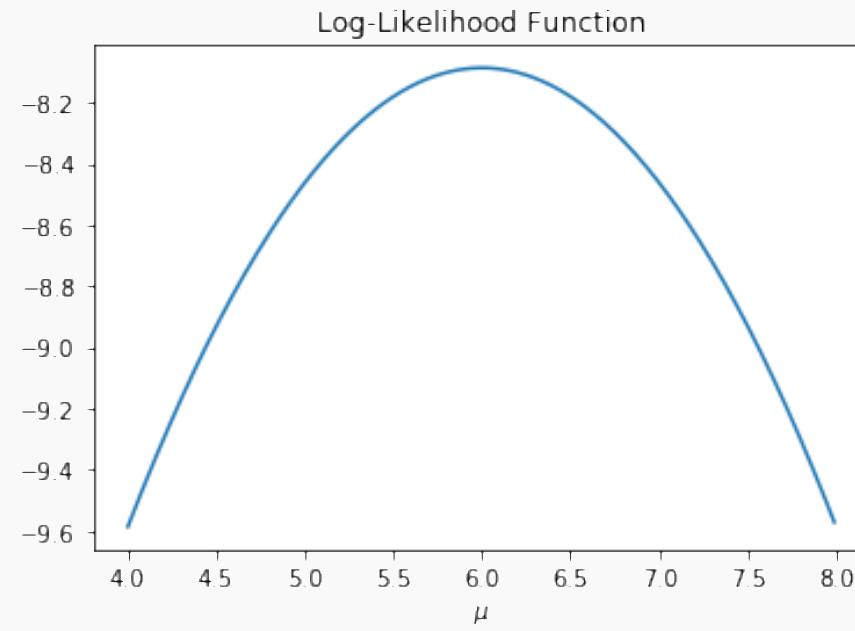
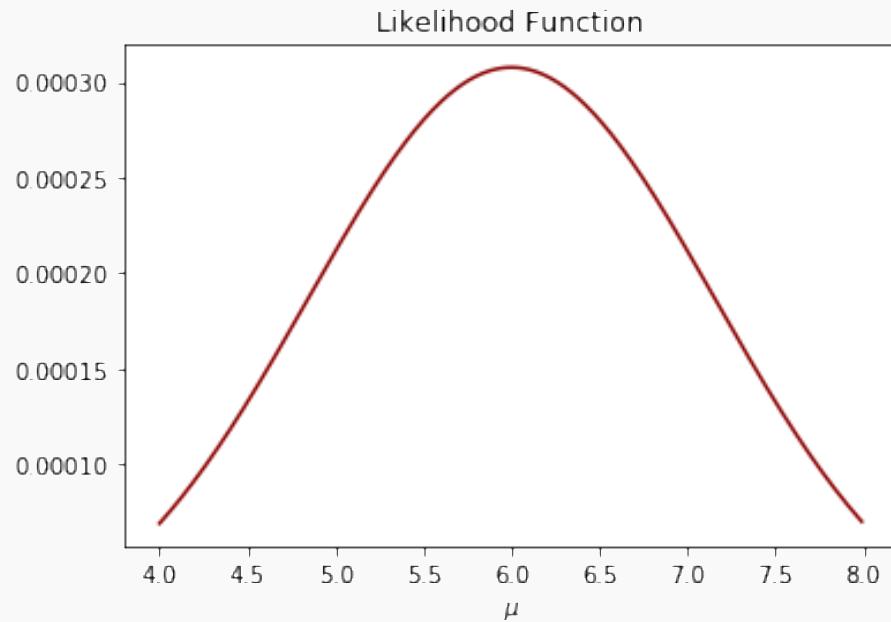
Given the observed data, what values of the model's parameters are likely?

To find the model that is most likely, we simply **maximize** the likelihood function.

The Likelihood of Linear Regression

We observe that the maximum of a function and the maximum of the logarithm of the function yield the same values.

Let's plot the likelihood and log-likelihood functions:



The Likelihood of Linear Regression

So, we can maximize the **log-likelihood** instead of the likelihood function

$$l(\beta_0, \beta_1, \sigma^2 | Y, X) = \ln \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(Y_i - (\beta_0 + \beta_1 X_i))^2}{2\sigma^2}}$$



$$l(\beta_0, \beta_1, \sigma^2 | Y, X) = \sum_{i=1}^n \ln \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(Y_i - (\beta_0 + \beta_1 X_i))^2}{2\sigma^2}}$$



A little bit of algebra

$$l(\beta_0, \beta_1, \sigma^2 | Y, X) = - \sum_{i=1}^n \ln (\sqrt{2\pi\sigma^2}) - \frac{1}{2\sigma^2} \sum_{i=1}^n (Y_i - (\beta_0 + \beta_1 X_i))^2$$

The Likelihood of Linear Regression

Just two more steps before we are done:

1. We don't need to find the σ , so we can replace any term involving σ with constants, which I call C_1, C_2

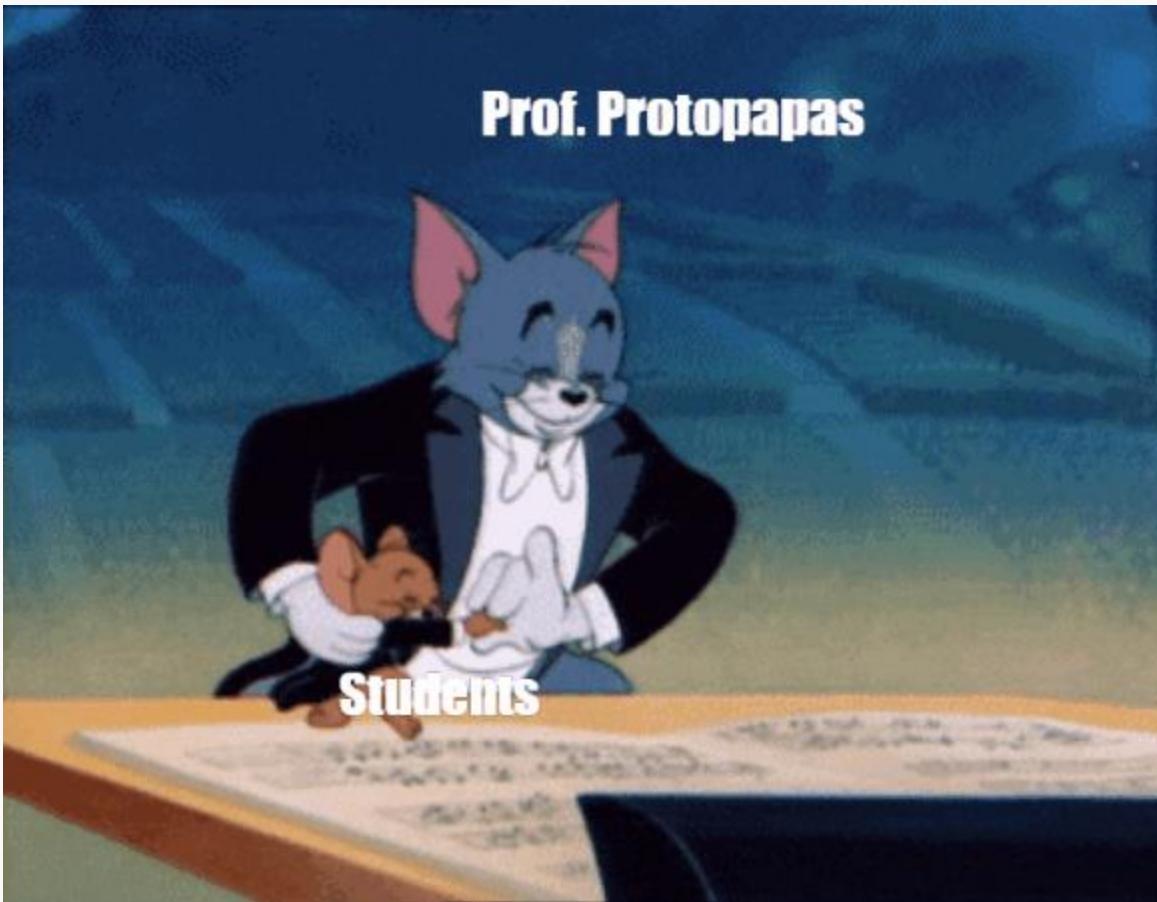
$$l(\beta_0, \beta_1, | Y, X) = C_1 - C_2 \sum_{i=1}^n (Y_i - (\beta_0 + \beta_1 X_i))^2$$

2. Maximizing the log-likelihood is equivalent to minimizing the negative log-likelihood. By dropping all constants, we can define this as our LOSS function, L

$$L = \sum_{i=1}^n (Y_i - (\beta_0 + \beta_1 X_i))^2$$

$$L = \sum_{i=1}^n (Y_i - (\beta_0 + \beta_1 X_i))^2$$

which brings us to our best friend, the Mean Squared Error (MSE)!



Thank you

A photograph of a two-toed sloth hanging from a tree branch in a lush green environment. The sloth is brown and has a long tail. It is hanging from a branch with its front legs and tail. There are large green leaves in the foreground and background.

Comparison of Ridge and Lasso

CS109A Introduction to Data Science

Pavlos Protopapas, Natesh Pillai and Chris Gumb



Photo: George Shaohua Qiao
sloth

Ridge, LASSO - Computational complexity

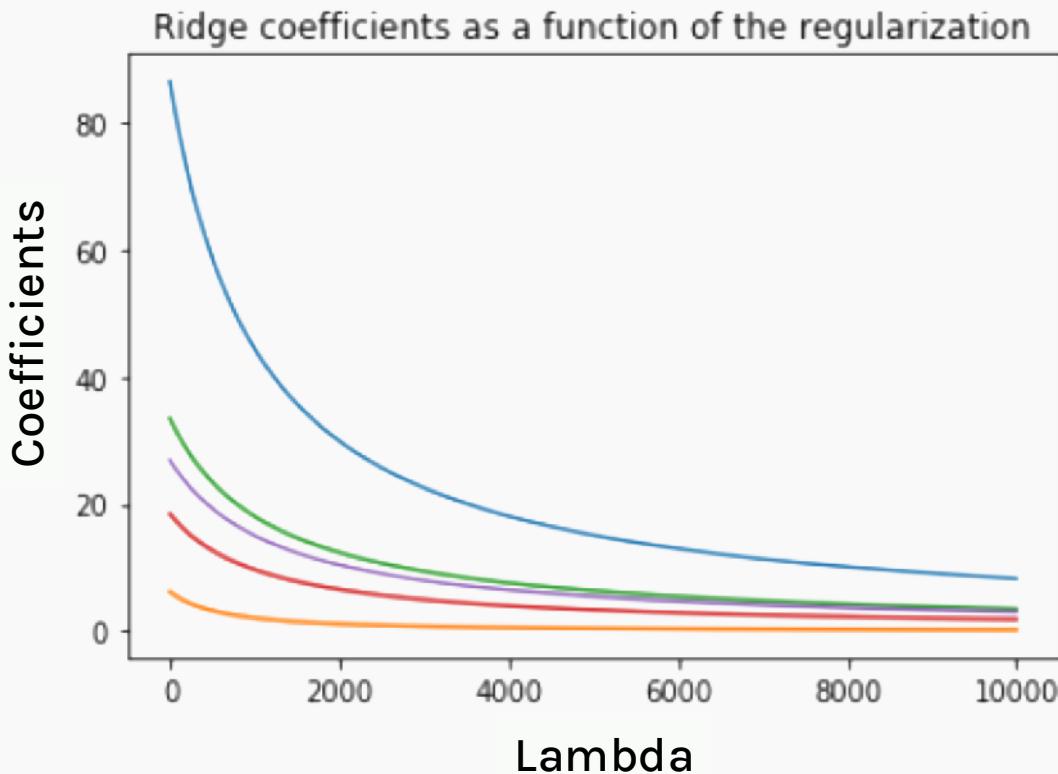
Solution to ridge regression:

$$\beta = (X^T X + \lambda I)^{-1} X^T Y$$

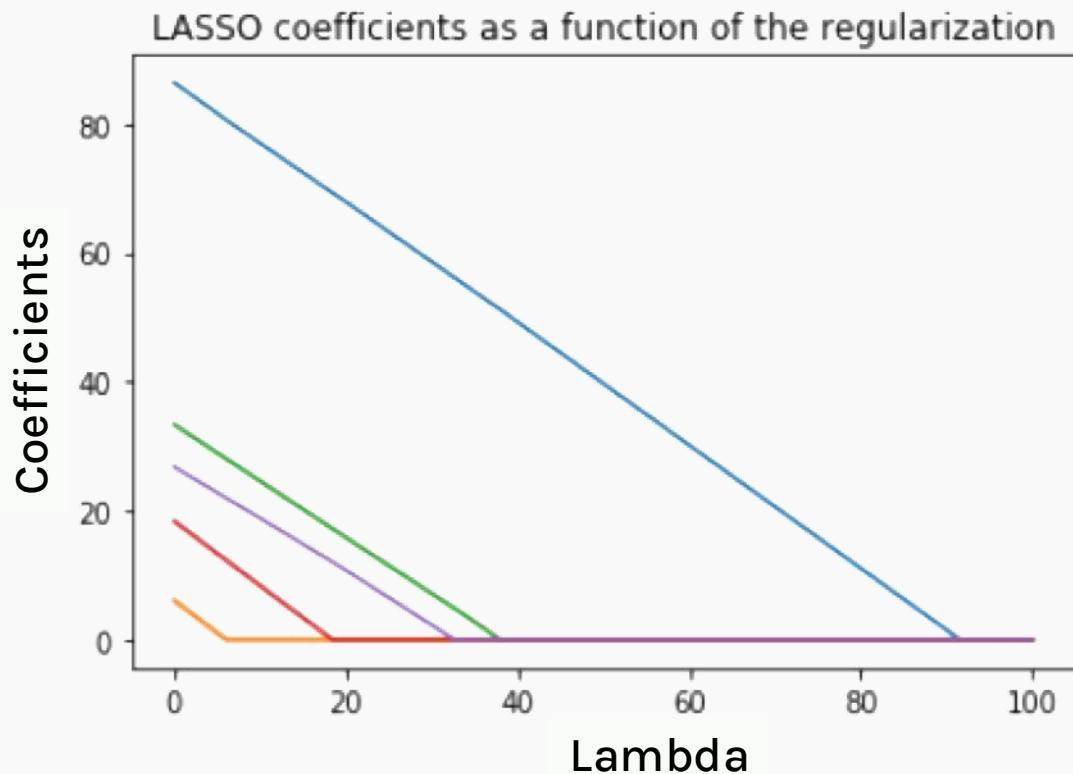
Solution to the LASSO regression:

LASSO has no conventional analytical solution, as the L1 norm has no derivative at zero. We can, however, use the concept of **subdifferential** or **subgradient** to find a manageable expression.

Ridge and LASSO visualized

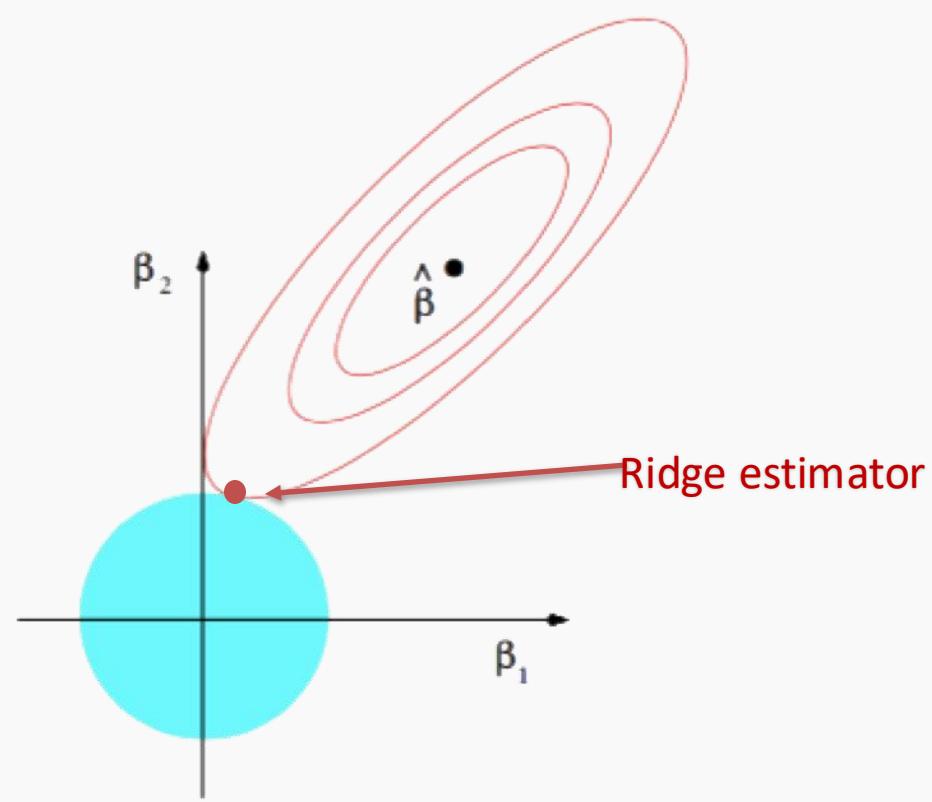


The values of the coefficients decrease as lambda increases, but they are not nullified.

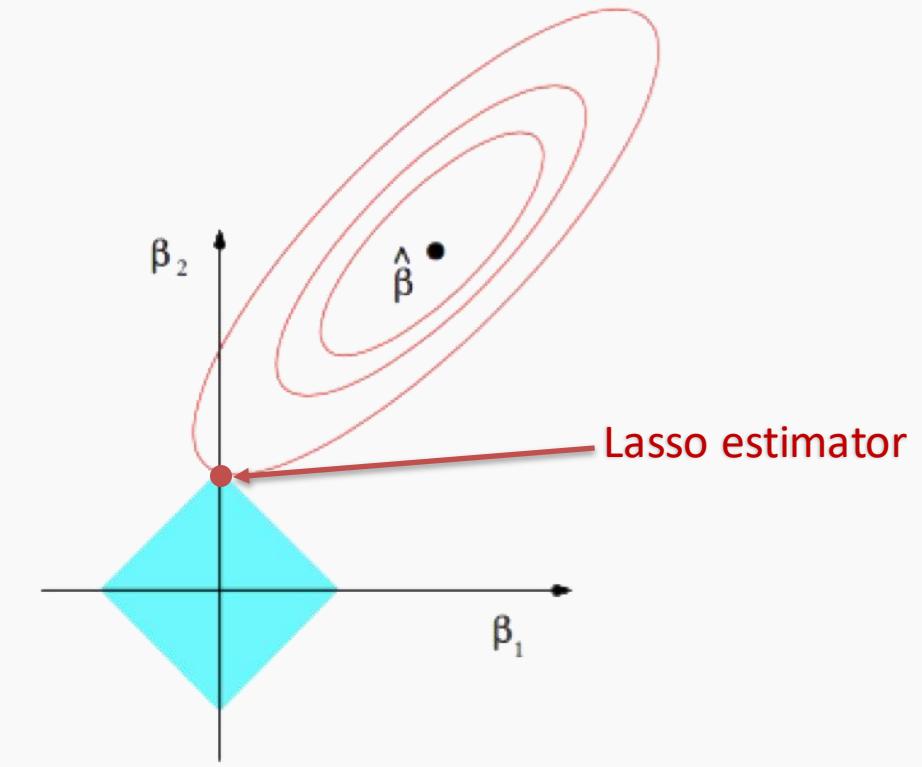


The values of the coefficients decrease as lambda increases and are nullified fast.

Ridge and LASSO visualized



The ridge estimator is where the constraint and the loss intersect.



The Lasso estimator tends to zero out parameters as the OLS loss can easily intersect with the constraint on one of the axes.

Comparing Lasso and Ridge Regression: Pros and Cons

Lasso Regression

Pros:

- 1. Feature Selection:** Automatically performs feature selection by setting some coefficients to zero.
- 2. Simplicity:** Results in simpler models that are easier to interpret.

Cons:

- 1. Computational Complexity:** Can be computationally expensive when dealing with very large datasets.

Ridge Regression

Pros:

- 1. Stability:** Stabilizes the coefficients, making it useful for ill-conditioned or multicollinear data.
- 2. Computational Complexity:** Reduces model complexity, which can lead to more efficient computation and easier interpretation of results.

Cons:

- 1. Feature Selection:** Does not reduce the number of features.



CS109A

Who Wants to be a

Data Scientist?



Which of the following will give the correct β 's in linear regression?

Options

A.
$$\begin{aligned}\beta_1 &= \sum_i (x_i y_i - x_i \bar{y} - \bar{x} y_i + \bar{y}) \\ \beta_0 &= \bar{y} - \beta_1 \bar{x}\end{aligned}$$

B.
$$\beta = (X^T X)^{-1} X^T Y$$

C.
$$\begin{aligned}\beta_1 &= \sum_i (y_i - \beta_0) / \sum_i x_i \\ \beta_0 &= \bar{y} - \beta_1 \bar{x}\end{aligned}$$

D.
$$\begin{aligned}\beta_1 &= \sqrt{(x_{11} - x_{21})^2 + (x_{21} - x_{22})^2} \\ \beta_0 &= \bar{y} - \beta_1 \bar{x}\end{aligned}$$



Which of the following is **NOT** a purpose for model validation?

Options

- A. To understand how well our model generalizes to data beyond the training set.
- B. To determine the best combination of hyperparameters.
- C. To measure, monitor, and hopefully reduce overfitting.
- D. To test our model and report its final performance score.



Thank you

Ridge and Lasso - Hyperparameters

CS109A Introduction to Data Science

Pavlos Protopapas, Natesh Pillai and Chris Gumb



Photo: Alyssa Taliotis
Agra, India

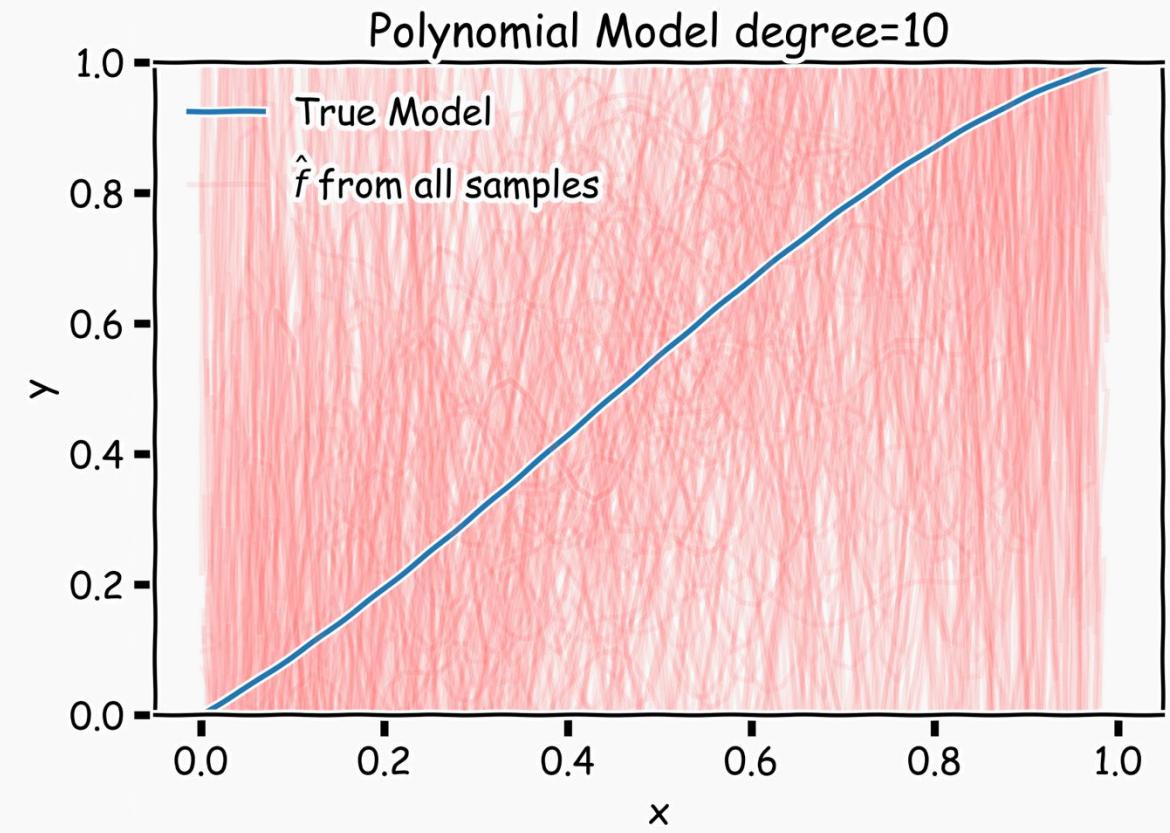
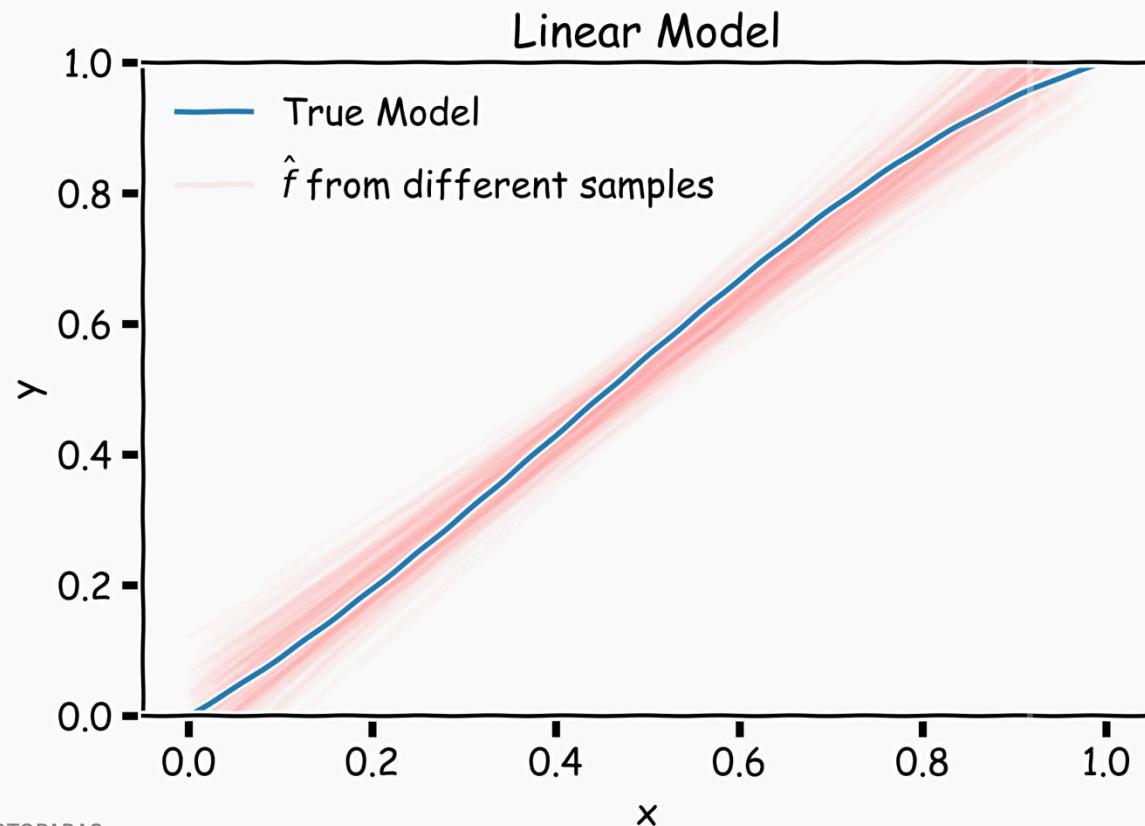
Outline

- Recap – Model Selection
- Generalization Error, Bias Variance Tradeoff
- **Regularization Techniques: Lasso, Ridge**

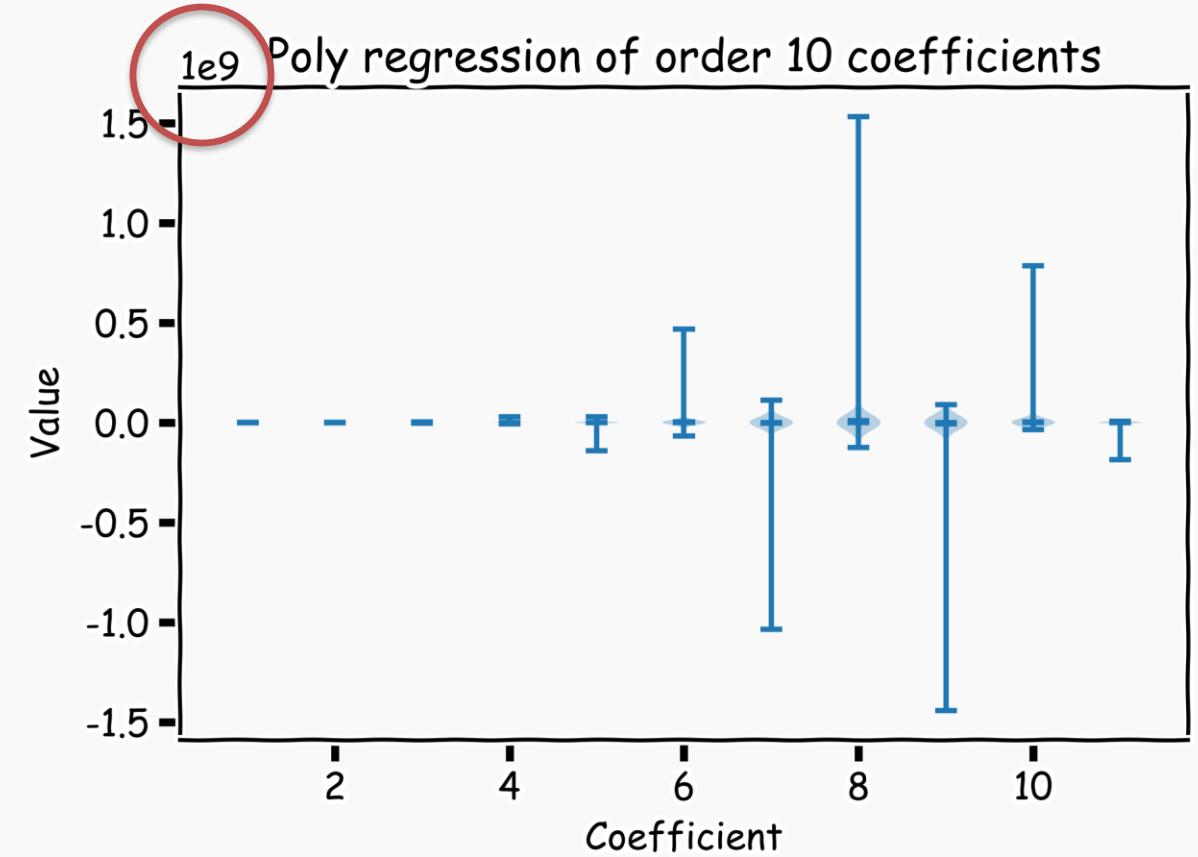
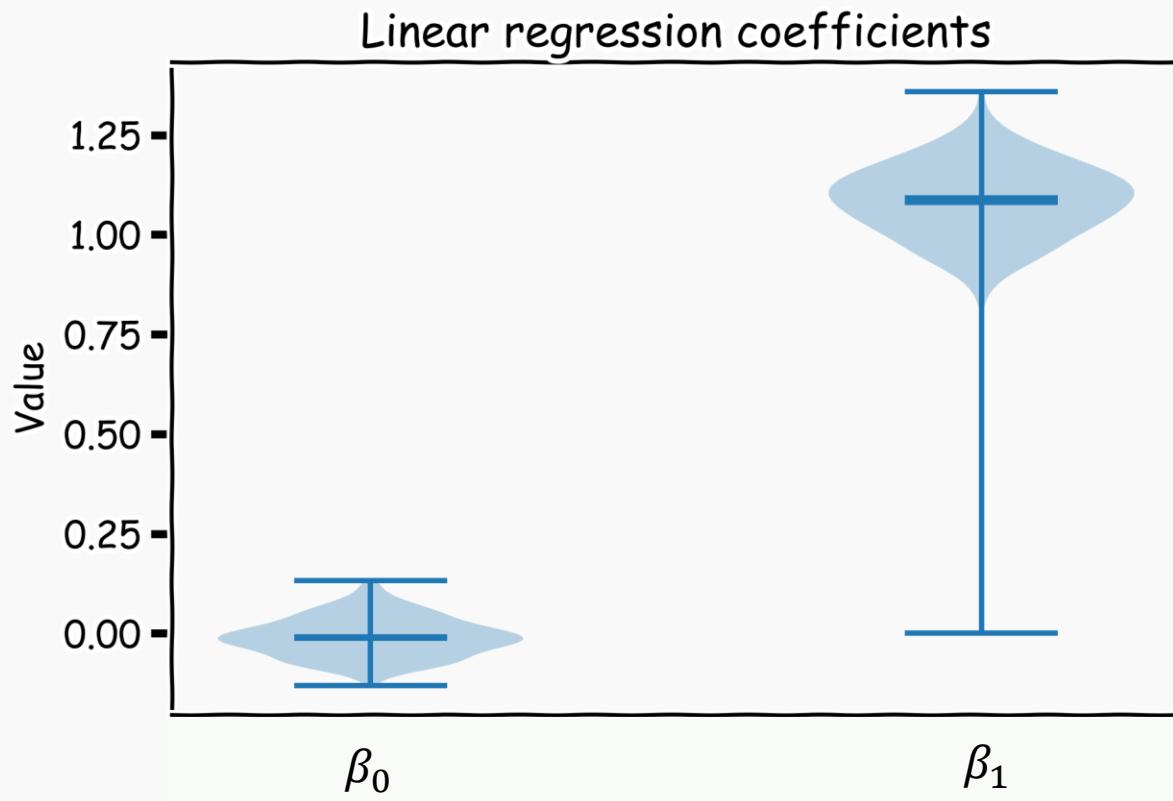
Bias vs Variance

Left: 2000, best fit straight lines, each fitted on a different 20-point training set.

Right: Best-fit models using degree-10 polynomial



Bias vs Variance





Model selection is the application of a principled method to determine the complexity of the model, e.g., choosing a subset of predictors, choosing the degree of the polynomial model etc.

A strong model can lead to overfitting. To avoid overfitting, we need to:

How do we discourage extreme values in the model parameters?

- there are too many terms:
 - the feature space has high dimensionality
 - the polynomial degree is too high
 - too many cross terms are considered
- the coefficients values are too **extreme**

Quiz

How would you discourage extreme values in the model parameters

Options:

- A. Divide all model parameters by a large number
- B. Make sure the causal relationship between predictors and response variable is true
- C. Discard any model with model parameter value larger than 1
- D. Penalize the model with a penalty that is proportional to the value its parameters

Regularization

What we want

Low model error

Minimize:

$$\frac{1}{n} \sum_{i=1}^n |y_i - \beta^\top x_i|^2$$

Discourage extreme values in
model parameters

Minimize:

Regularization

What we want

Low model error

Minimize:

$$\frac{1}{n} \sum_{i=1}^n |y_i - \beta^\top x_i|^2$$

Discourage extreme values in
model parameters

Minimize:

$$L_{reg} = \begin{cases} \sum_{j=1}^J \beta_j^2 \\ \sum_{j=1}^J |\beta_j| \end{cases}$$



What we want

Low model error

Minimize:

$$\frac{1}{n} \sum_{i=1}^n |y_i - \beta^\top x_i|^2$$

Discourage extreme values in
model parameters

Minimize:

$$L_{reg} = \begin{cases} \sum_{j=1}^J \beta_j^2 \\ \sum_{j=1}^J |\beta_j| \end{cases}$$

How do we combine these
two objectives?

Regularization

What we want

Low model error

Discourage extreme values in
model parameters

Minimize:

Minimize:

$$\mathcal{L}_{REG} = \frac{1}{n} \sum_{i=1}^n |y_i - \beta^\top \mathbf{x}_i|^2 + L_{reg}$$

Regularization

What we want

Low model error

Discourage extreme values in
model parameters

Minimize:

λ is the **regularization parameter**. It controls the relative importance between model error and the regularization term

Minize:

$$\mathcal{L}_{REG} = \frac{1}{n} \sum_{i=1}^n |y_i - \beta^\top x_i|^2 + \lambda L_{reg}$$

Regularization

What we want

Low model error

Discourage extreme values in
model parameters

$\lambda = 0$: equivalent to simple linear
regression

mize:

$\lambda = \infty$: yields a model with β' s = 0

$$\mathcal{L}_{REG} = \frac{1}{n} \sum_{i=1}^n |y_i - \beta^\top x_i|^2 + \lambda L_{reg}$$



What we want

Low model error

Discourage extreme values in
model parameters

Minimize:

Minimize:

How do we
determine λ ?

$$\mathcal{L}_{REG} = \frac{1}{n} \sum_{i=1}^n |y_i - \beta^\top x_i|^2 + \lambda L_{reg}$$

Regularization

What we want

Low model error

Discourage extreme values in
model parameters

Minimize:

Minimize:

Cross
Validation!

$$\mathcal{L}_{REG} = \frac{1}{n} \sum_{i=1}^n |y_i - \beta^\top x_i|^2 + \lambda L_{reg}$$

Regularization: **LASSO** Regression

What we want

Low model error

Discourage extreme values in
model parameters

Minimize:

Minimize:

Note that $\sum_{j=1}^J |\beta_j|$ is the ℓ_1
norm of the vector β

$$\mathcal{L}_{LASSO} = \frac{1}{n} \sum_{i=1}^n |y_i - \beta^\top \mathbf{x}_i|^2 + \lambda \sum_{j=1}^J |\beta_j|$$

Regularization: **LASSO** Regression



What we want

Low model error

Discourage extreme values in
parameters

No need to regularize the bias, β_0
Why?

minimize:

$$\mathcal{L}_{LASSO} = \frac{1}{n} \sum_{i=1}^n |y_i - \beta^\top x_i|^2 + \lambda \sum_{j=1}^J |\beta_j|$$

Regularization: **LASSO** Regression

Lasso regression: minimize \mathcal{L}_{LASSO} with respect to β 's

$$\mathcal{L}_{LASSO} = \frac{1}{n} \sum_{i=1}^n |y_i - \boldsymbol{\beta}^\top \mathbf{x}_i|^2 + \lambda \sum_{j=1}^J |\beta_j|$$

Regularization: **Ridge** Regression

Ridge regression: minimize \mathcal{L}_{RIDGE} with res

Note that $\sum_{j=1}^J \beta_j^2$ is the L_2 norm square
of the vector β

$$\mathcal{L}_{RIDGE} = \frac{1}{n} \sum_{i=1}^n |y_i - \beta^\top x_i|^2 + \lambda \sum_{j=1}^J \beta_j^2$$

Regularization: **Ridge** Regression

Ridge regression: minimize \mathcal{L}_{RIDGE} with respect to β 's

$$\mathcal{L}_{RIDGE} = \frac{1}{n} \sum_{i=1}^n |y_i - \boldsymbol{\beta}^\top \mathbf{x}_i|^2 + \lambda \sum_{j=1}^J \beta_j^2$$

No need to regularize the bias, β_0 , since it is not connected to the predictors.

Ridge regularization with only validation : step by step

For ridge regression there exist an analytical solution for the coefficients:

$$\hat{\beta}_{Ridge}(\lambda) = (X^T X + \lambda I)^{-1} X^T Y$$

1. split data into $\{\{X, Y\}_{train}, \{X, Y\}_{validation}, \{X, Y\}_{test}\}$
2. for λ in $\{\lambda_{min}, \dots, \lambda_{max}\}$:
 1. determine the β that minimizes the L_{ridge} , $\beta_{Ridge}(\lambda) = (X^T X + \lambda I)^{-1} X^T Y$, using the train data.
 2. record $L_{MSE}(\lambda)$ using validation data.

Ridge regularization with only validation : step by step

For ridge regression there exist an analytical solution for the coefficients:

$$\hat{\beta}_{Ridge}(\lambda) = (X^T X + \lambda I)^{-1} X^T Y$$

1. split data into $\{\{X, Y\}_{train}, \{X, Y\}_{validation}, \{X, Y\}_{test}\}$
2. for λ in $\{\lambda_{min}, \dots, \lambda_{max}\}$:
 1. determine the β that minimizes the L_{ridge} , $\beta_{Ridge}(\lambda) = (X^T X + \lambda I)^{-1} X^T Y$, using the train data.
 2. record $L_{MSE}(\lambda)$ using validation data.
3. select the λ that minimizes the MSE loss on the validation data,

$$\lambda_{ridge} = \operatorname{argmin}_\lambda L_{MSE}(\lambda)$$

Ridge regularization with only validation : step by step

For ridge regression there exist an analytical solution for the coefficients:

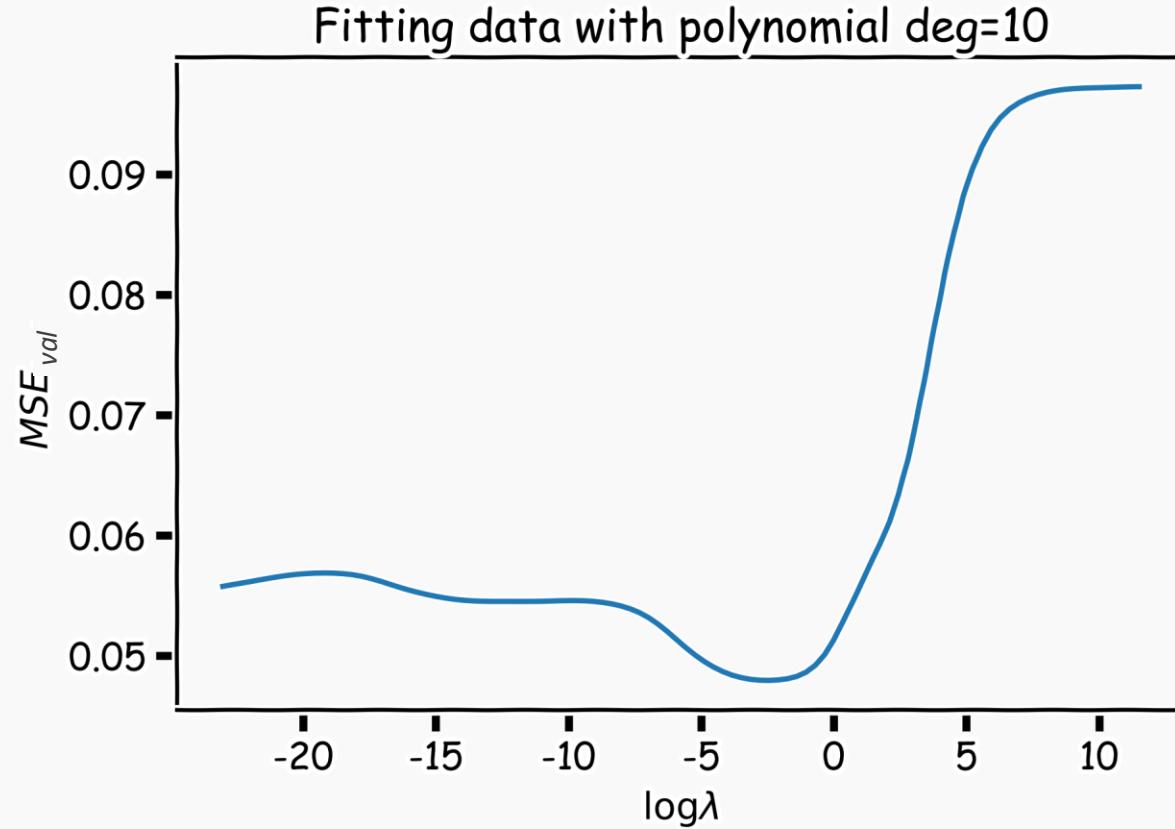
$$\hat{\beta}_{Ridge}(\lambda) = (X^T X + \lambda I)^{-1} X^T Y$$

1. split data into $\{\{X, Y\}_{train}, \{X, Y\}_{validation}, \{X, Y\}_{test}\}$
2. for λ in $\{\lambda_{min}, \dots, \lambda_{max}\}$:
 1. determine the β that minimizes the L_{ridge} , $\beta_{Ridge}(\lambda) = (X^T X + \lambda I)^{-1} X^T Y$, using the train data.
 2. record $L_{MSE}(\lambda)$ using validation data.
3. select the λ that minimizes the MSE loss on the validation data,

$$\lambda_{ridge} = \operatorname{argmin}_\lambda L_{MSE}(\lambda)$$

4. Refit the model using both train and validation data, $\{\{X, Y\}_{train}, \{X, Y\}_{validation}\}$, now using λ_{ridge} , resulting to $\hat{\beta}_{ridge}(\lambda_{ridge})$
5. Report MSE or R^2 on $\{X, Y\}_{test}$ given the $\hat{\beta}_{ridge}(\lambda_{ridge})$

Ridge regularization with validation only



Lasso regularization with validation only: step by step

For Lasso regression, there is **no** analytical solution for the coefficients, so we use a **solver**.

1. split data into $\{\{X, Y\}_{train}, \{X, Y\}_{validation}, \{X, Y\}_{test}\}$
2. for λ in $\{\lambda_{min}, \dots, \lambda_{max}\}$:
 - A. determine the β that minimizes the L_{lasso} , $\beta_{lasso}(\lambda)$, using the train data. **This is done using a solver.**
 - B. record $L_{MSE}(\lambda)$ using the validation data.

Lasso regularization with validation only: step by step

For Lasso regression, there is **no** analytical solution for the coefficients, so we use a **solver**.

1. split data into $\{\{X, Y\}_{train}, \{X, Y\}_{validation}, \{X, Y\}_{test}\}$
2. for λ in $\{\lambda_{min}, \dots, \lambda_{max}\}$:
 - A. determine the β that minimizes the L_{lasso} , $\beta_{lasso}(\lambda)$, using the train data. **This is done using a solver.**
 - B. record $L_{MSE}(\lambda)$ using the validation data.
3. select the λ that minimizes the **MSE loss** on the validation data,

$$\lambda_{lasso} = \operatorname{argmin}_\lambda L_{MSE}(\lambda)$$

Lasso regularization with validation only: step by step

For Lasso regression, there is **no** analytical solution for the coefficients, so we use a **solver**.

1. split data into $\{\{X, Y\}_{train}, \{X, Y\}_{validation}, \{X, Y\}_{test}\}$
2. for λ in $\{\lambda_{min}, \dots, \lambda_{max}\}$:
 - A. determine the β that minimizes the L_{Lasso} , $\beta_{Lasso}(\lambda)$, using the train data. **This is done using a solver.**
 - B. record $L_{MSE}(\lambda)$ using the validation data.
3. select the λ that minimizes the **MSE loss** on the validation data,
$$\lambda_{Lasso} = \operatorname{argmin}_{\lambda} L_{MSE}(\lambda)$$
4. Refit the model using both **train and validation data**, $\{\{X, Y\}_{train}, \{X, Y\}_{validation}\}$, now using λ_{Lasso} , resulting to $\hat{\beta}_{Lasso}(\lambda_{Lasso})$
5. Report MSE or R^2 on $\{X, Y\}_{test}$ given the $\hat{\beta}_{Lasso}(\lambda_{Lasso})$

Ridge regularization with CV: step by step



Ridge regularization with CV: step by step

1. remove $\{X, Y\}_{test}$ from data
2. split the rest of data into K folds, $\{\{X, Y\}_{train}^{-k}, \{X, Y\}_{val}^k\}$

	λ_1	λ_2	...	λ_n
k_1				
k_2				
...				
k_n				

Ridge regularization with CV: step by step

1. remove $\{X, Y\}_{test}$ from data
2. split the rest of data into K folds, $\{\{X, Y\}_{train}^{-k}, \{X, Y\}_{val}^k\}$
3. for k in $\{1, \dots, K\}$
 for λ in $\{\lambda_0, \dots, \lambda_n\}$:

	λ_1	λ_2	...	λ_n
k_1				
k_2				
...				
k_n				

Ridge regularization with CV: step by step

1. remove $\{X, Y\}_{test}$ from data
2. split the rest of data into K folds, $\{\{X, Y\}_{train}^{-k}, \{X, Y\}_{val}^k\}$
3. for k in $\{1, \dots, K\}$
 for λ in $\{\lambda_0, \dots, \lambda_n\}$:

λ_1	λ_2	\dots	λ_n
k_1	L_{11}		
k_2			
\dots			
k_n			

- A. determine the β that minimizes the L_{ridge} , $\beta_{ridge}(\lambda, k) = (X^T X + \lambda I)^{-1} X^T Y$,
using the train data of the fold, $\{X, Y\}_{train}^{-k}$.
- B. record $L_{MSE}(\lambda, k)$ using the validation data of the fold $\{X, Y\}_{val}^k$

Ridge regularization with CV: step by step

1. remove $\{X, Y\}_{test}$ from data
2. split the rest of data into K folds, $\{\{X, Y\}_{train}^{-k}, \{X, Y\}_{val}^k\}$
3. for k in $\{1, \dots, K\}$

for λ in $\{\lambda_0, \dots, \lambda_n\}$:

- A. determine the β that minimizes the L_{ridge} , $\beta_{ridge}(\lambda, k) = (X^T X + \lambda I)^{-1} X^T Y$, using the train data of the fold, $\{X, Y\}_{train}^{-k}$.
- B. record $L_{MSE}(\lambda, k)$ using the validation data of the fold $\{X, Y\}_{val}^k$

	λ_1	λ_2	...	λ_n
k_1	L_{11}	L_{12}
k_2	L_{21}
...
k_n

Ridge regularization with CV: step by step

1. remove $\{X, Y\}_{test}$ from data
2. split the rest of data into K folds, $\{\{X, Y\}_{train}^{-k}, \{X, Y\}_{val}^k\}$
3. for k in $\{1, \dots, K\}$

for λ in $\{\lambda_0, \dots, \lambda_n\}$:

A. determine the β that minimizes the L_{ridge} , $\beta_{ridge}(\lambda, k) = (X^T X + \lambda I)^{-1} X^T Y$,
using the train data of the fold, $\{X, Y\}_{train}^{-k}$.

B. record $L_{MSE}(\lambda, k)$ using the validation data of the fold $\{X, Y\}_{val}^k$

At this point we have a 2-D matrix, rows are for different k , and columns are for different λ values.

	λ_1	λ_2	...	λ_n
k_1	L_{11}	L_{12}
k_2	L_{21}
...
k_n

Ridge regularization with CV: step by step

1. remove $\{X, Y\}_{test}$ from data
2. split the rest of data into K folds, $\{\{X, Y\}_{train}^{-k}, \{X, Y\}_{val}^k\}$
3. for k in $\{1, \dots, K\}$
 - for λ in $\{\lambda_0, \dots, \lambda_n\}$:
 - A. determine the β that minimizes the L_{ridge} , $\beta_{ridge}(\lambda, k) = (X^T X + \lambda I)^{-1} X^T Y$, using the train data of the fold, $\{X, Y\}_{train}^{-k}$.
 - B. record $L_{MSE}(\lambda, k)$ using the validation data of the fold $\{X, Y\}_{val}^k$

At this point we have a 2-D matrix, rows are for different k , and columns are for different λ values.
4. Calculate the average MSE, $\bar{L}_{MSE}(\lambda)$ the for each λ by averaging $L_{MSE}(\lambda, k)$ over k folds.

	λ_1	λ_2	...	λ_n
k_1	L_{11}	L_{12}
k_2	L_{21}
...
k_n
$E[]$	\bar{L}_1	\bar{L}_2	...	\bar{L}_n

Ridge regularization with CV: step by step

1. remove $\{X, Y\}_{test}$ from data
2. split the rest of data into K folds, $\{\{X, Y\}_{train}^{-k}, \{X, Y\}_{val}^k\}$
3. for k in $\{1, \dots, K\}$
 - for λ in $\{\lambda_0, \dots, \lambda_n\}$:
 - A. determine the β that minimizes the L_{ridge} , $\beta_{ridge}(\lambda, k) = (X^T X + \lambda I)^{-1} X^T Y$, using the train data of the fold, $\{X, Y\}_{train}^{-k}$.
 - B. record $L_{MSE}(\lambda, k)$ using the validation data of the fold $\{X, Y\}_{val}^k$

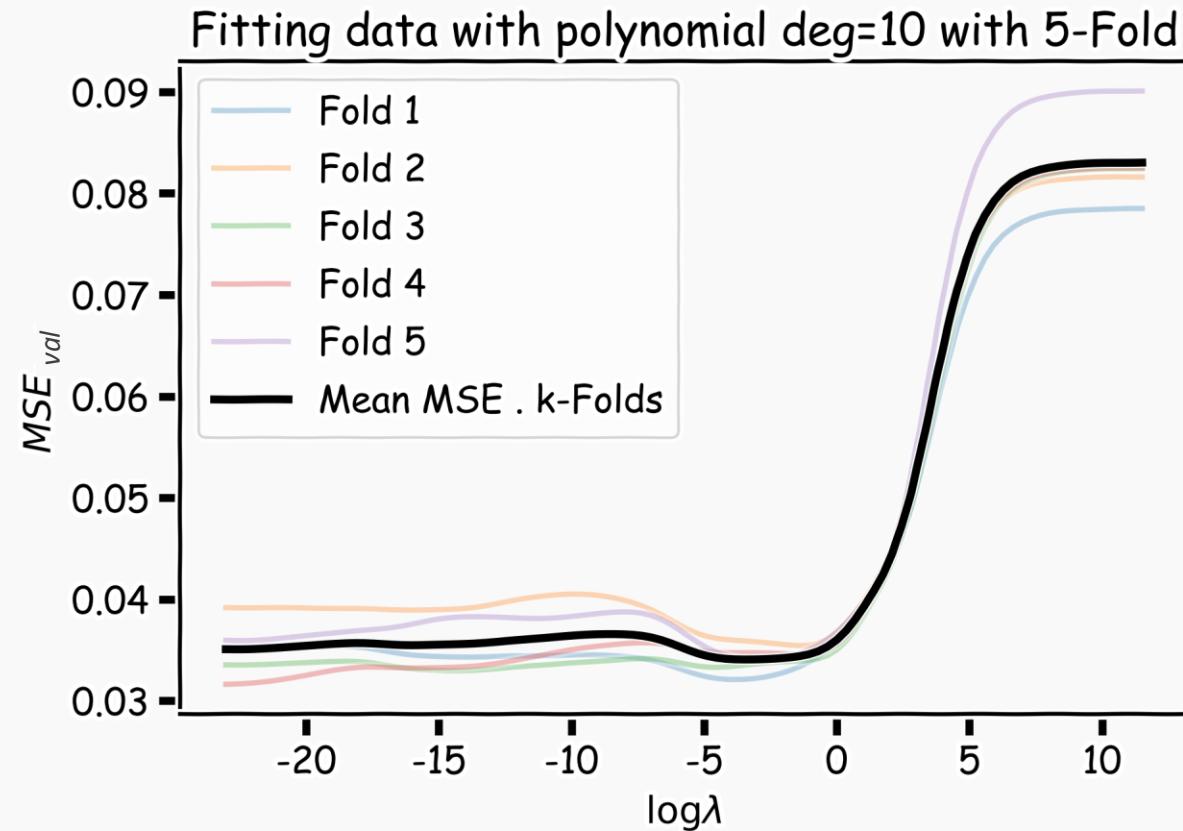
At this point we have a 2-D matrix, rows are for different k , and columns are for different λ values.
 4. Calculate the average MSE, $\bar{L}_{MSE}(\lambda)$ the for each λ by averaging $L_{MSE}(\lambda, k)$ over k folds.
 5. Find the λ that minimizes the $\bar{L}_{MSE}(\lambda)$, resulting to λ_{ridge} .

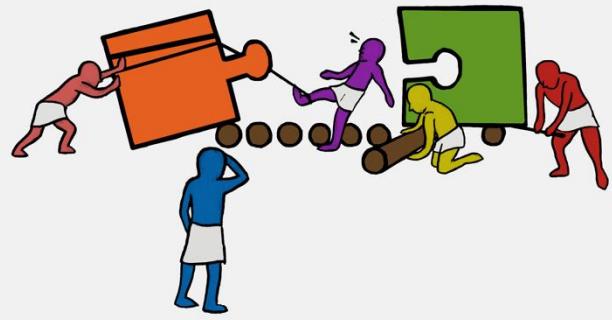
	λ_1	λ_2	...	λ_n
k_1	L_{11}	L_{12}
k_2	L_{21}
...
k_n
$E[]$	\bar{L}_1	\bar{L}_2	...	\bar{L}_n

Ridge regularization with CV: step by step

1. remove $\{X, Y\}_{test}$ from data
2. split the rest of data into K folds, $\{\{X, Y\}_{train}^{-k}, \{X, Y\}_{val}^k\}$
3. for k in $\{1, \dots, K\}$
 - for λ in $\{\lambda_0, \dots, \lambda_n\}$:
 - A. determine the β that minimizes the L_{ridge} , $\beta_{ridge}(\lambda, k) = (X^T X + \lambda I)^{-1} X^T Y$, using the train data of the fold, $\{X, Y\}_{train}^{-k}$.
 - B. record $L_{MSE}(\lambda, k)$ using the validation data of the fold $\{X, Y\}_{val}^k$
- At this point we have a 2-D matrix, rows are for different k , and columns are for different λ values.
4. Calculate the average MSE, $\bar{L}_{MSE}(\lambda)$ the for each λ by averaging $L_{MSE}(\lambda, k)$ over k folds.
5. Find the λ that minimizes the $\bar{L}_{MSE}(\lambda)$, resulting to λ_{ridge} .
6. Refit the model using the full training data, $\{\{X, Y\}_{train}, \{X, Y\}_{val}\}$, resulting to $\hat{\beta}_{ridge}(\lambda_{ridge})$
7. report MSE or R^2 on $\{X, Y\}_{test}$ given the $\hat{\beta}_{ridge}(\lambda_{ridge})$

Ridge regularization with cross-validation only: step by step





Generalization Error and Bias Variance Tradeoff

CS109A Introduction to Data Science

Pavlos Protopapas, Natesh Pillai and Chris Gumb



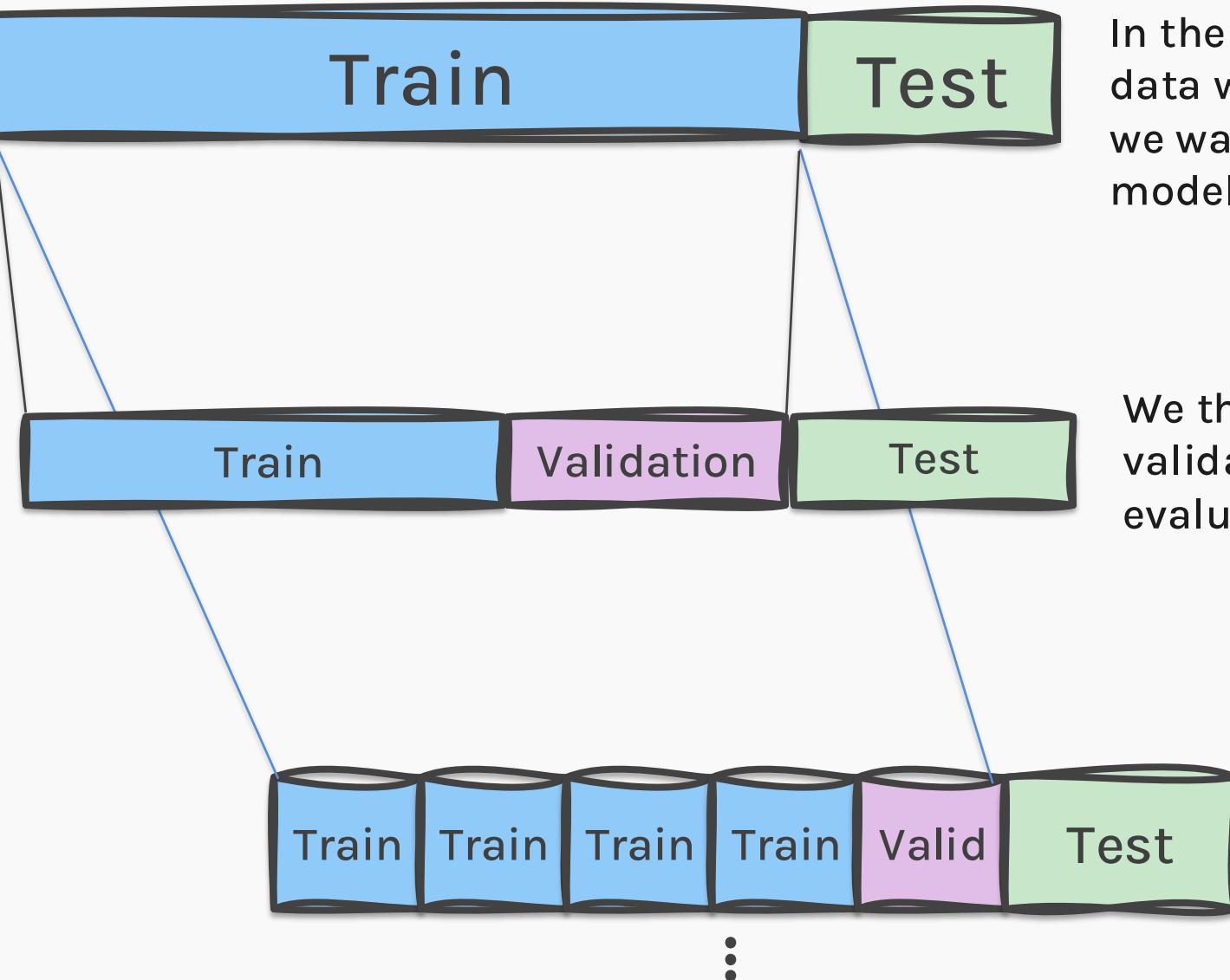
Photo: Junyang Deng
Sayram Lake

Outline

- Recap – Model Selection
- Generalization Error, Bias Variance Tradeoff
- Regularization Techniques: Lasso, Ridge

Outline

- Recap – Model Selection
- Generalization Error, Bias Variance Tradeoff
- Regularization Techniques: Lasso, Ridge



In the beginning, we separated a portion of the data which we never touch until the very end when we want to evaluate the performance of the final model. Normally, this is called train + test split.*

We then saw we can split train data into train + validation (to find the best model) + test (to evaluate the performance of the model).

We then finally saw that we can use cross-validation. It splits the train data into k buckets and uses different chunks of data as the validation set.

* sometimes they (not us!) also call this train + validation split, while meaning train + test

Recall - Model Selection

1. Model selection as a way to avoid overfitting
2. Validation set to select the best model
3. Cross validation to avoid overfitting to the validation set

Ways of model selection:

- Exhaustive search
- Greedy algorithms
- Fine tuning hyper-parameters
- Regularization

When you realize k-Fold Cross Validation can only validate your hyperparameters, not yourself..



Outline

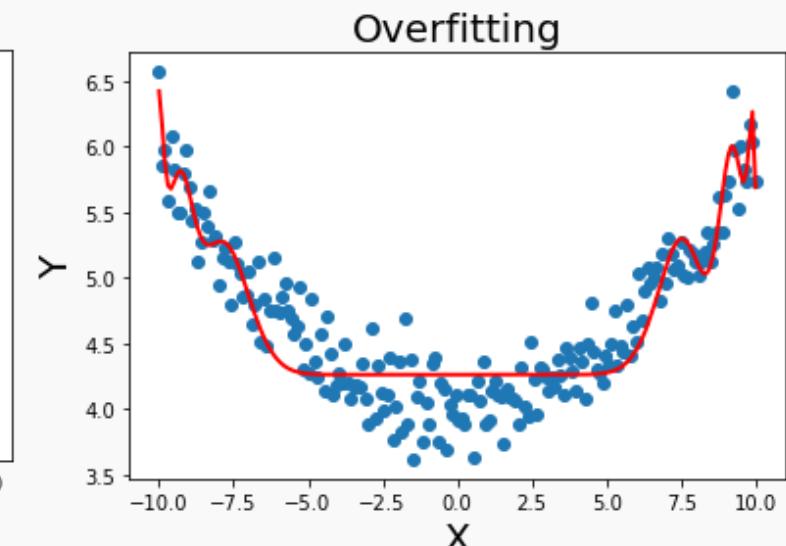
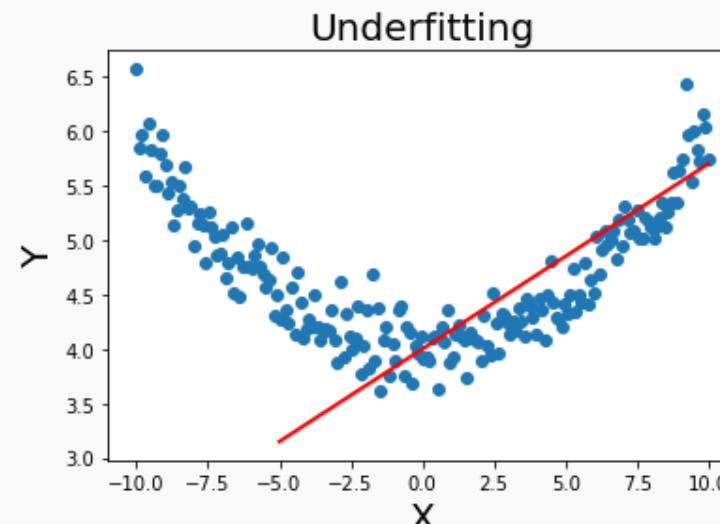
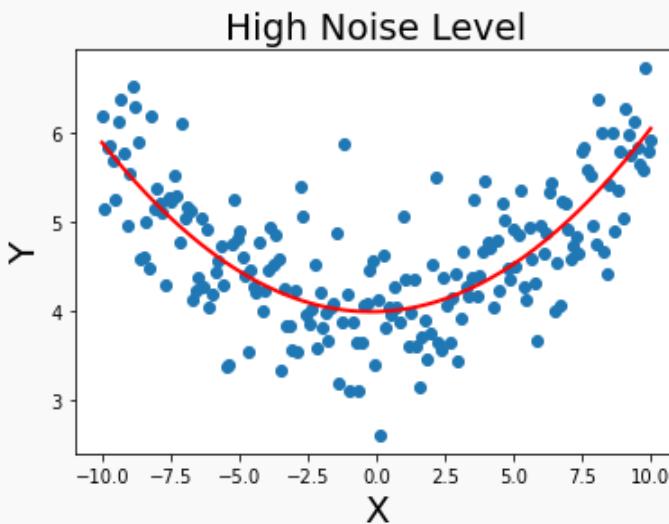
- Recap – Model Selection
- Generalization Error, Bias Variance Tradeoff
- Regularization Techniques: Lasso Ridge

Test Error and Generalization

We know to **evaluate** models on both train and test data because models can do **well** on train data but do **poorly** on new data.

When models do well on new data, it is called **generalization**.

There are at least three ways a model can have a high-test error.



Irreducible and Reducible Errors

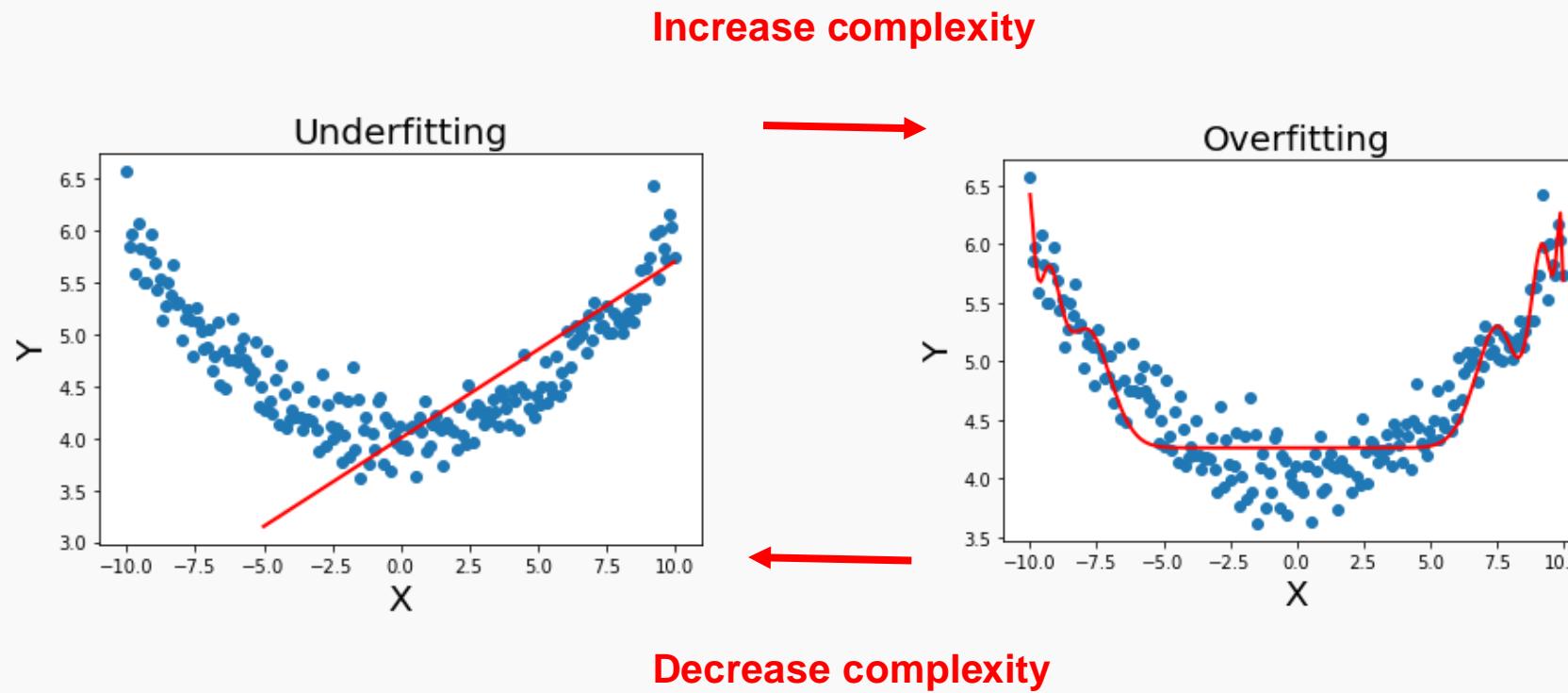
We distinguished the contributions of noise to the generalization error:

Irreducible error (or aleatoric error): we can't do anything to decrease the error due to noise.

Reducible error (or epistemic error): we can decrease the error due to overfitting and underfitting by improving the model.

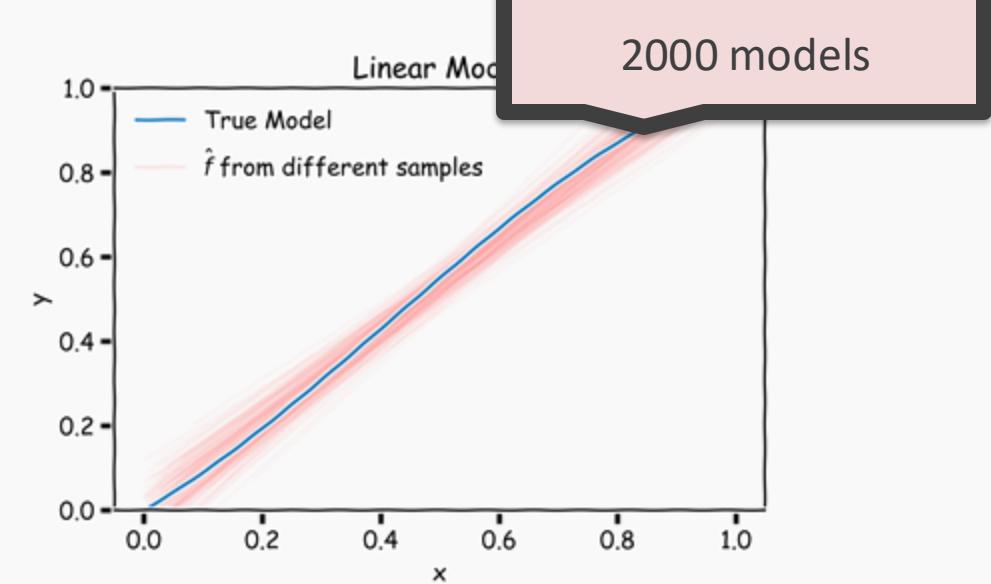
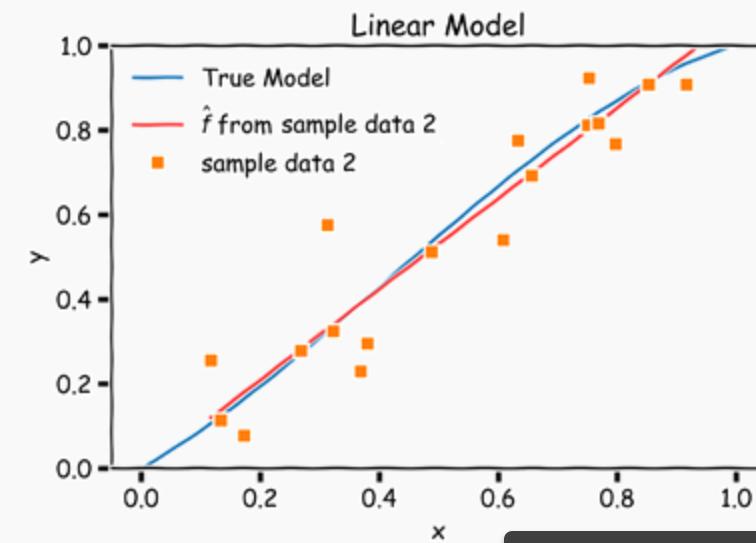
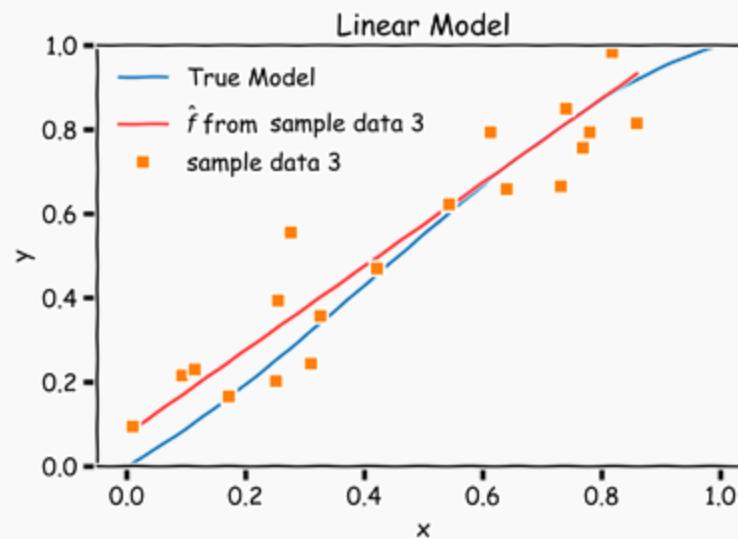
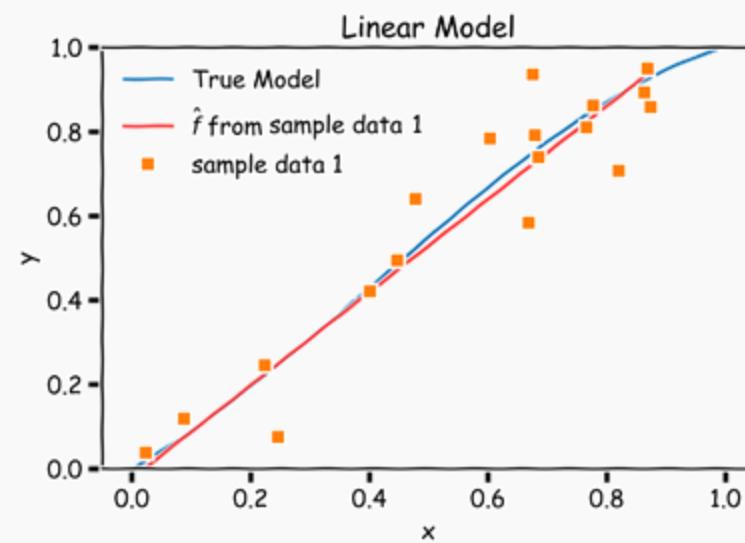
The Bias-Variance: Bias

Reducible error comes from either underfitting or overfitting. There is a tradeoff between the two sources of errors:

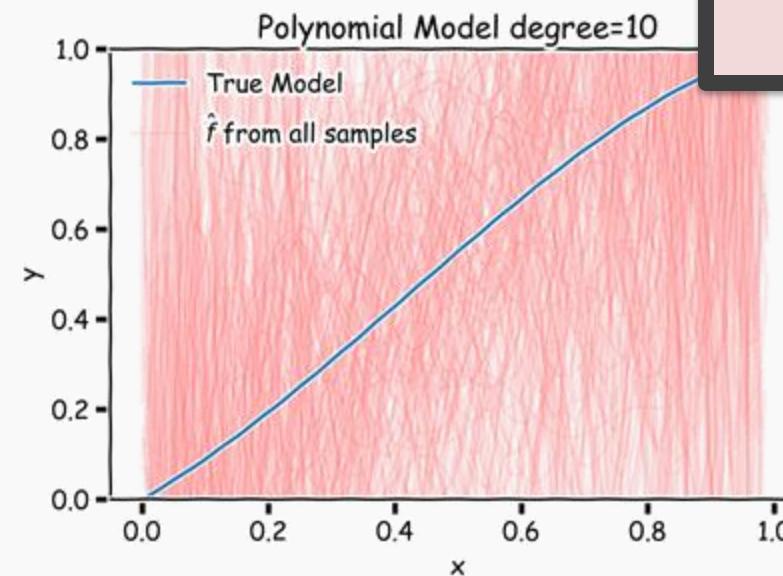
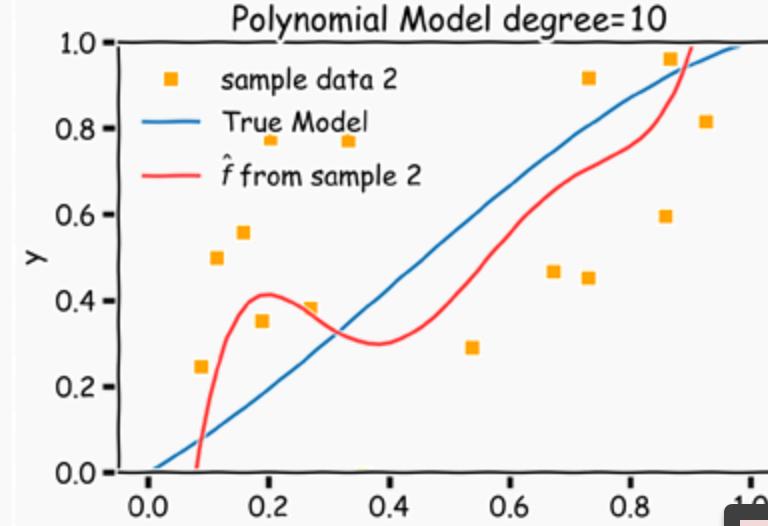
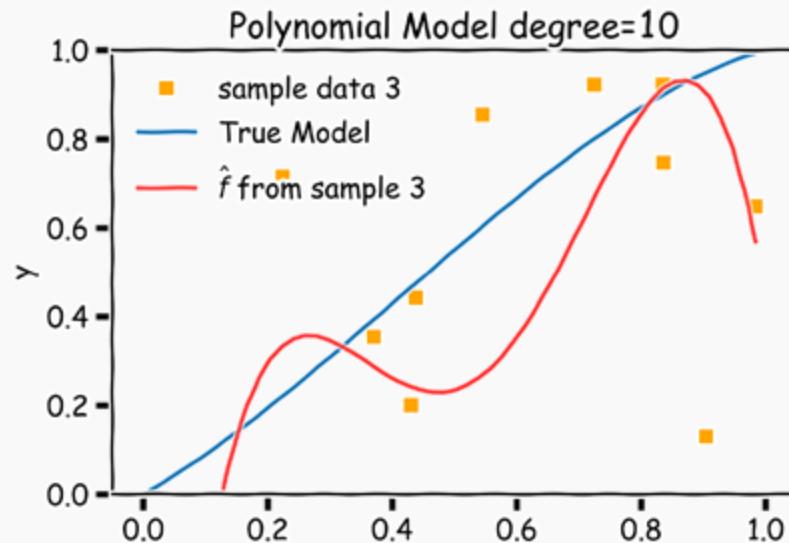
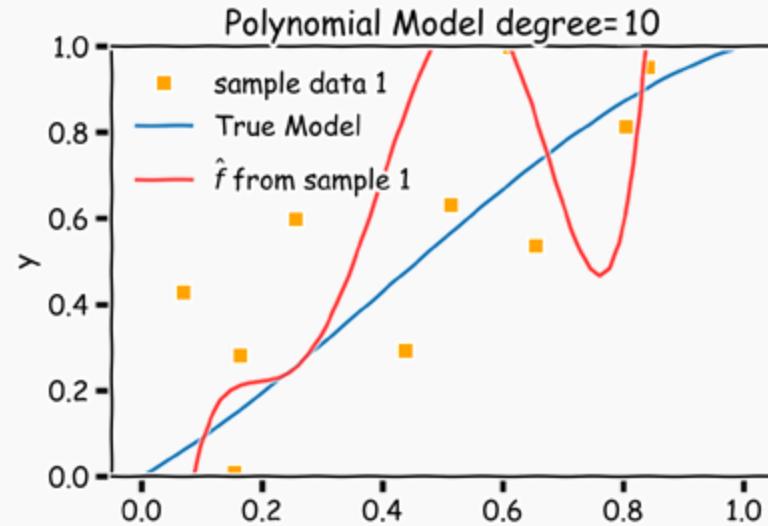




Bias vs Variance: Variance of a SIMPLE model



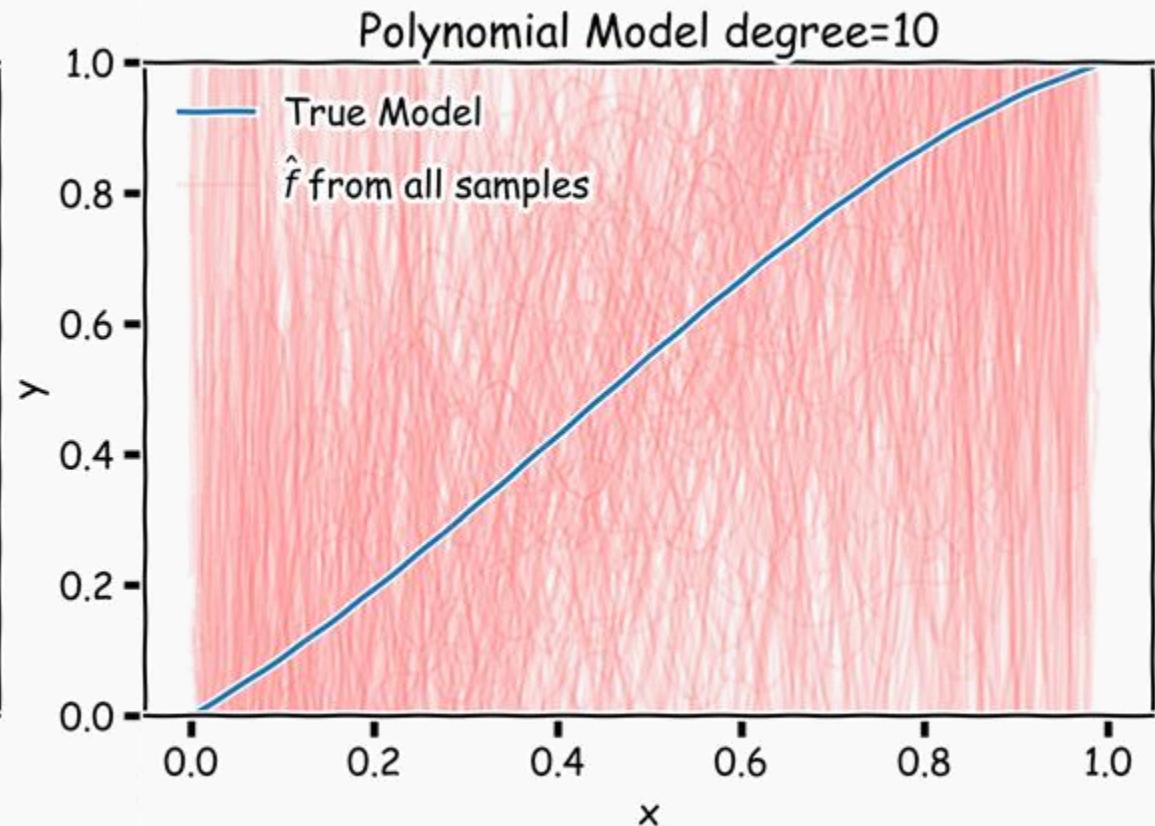
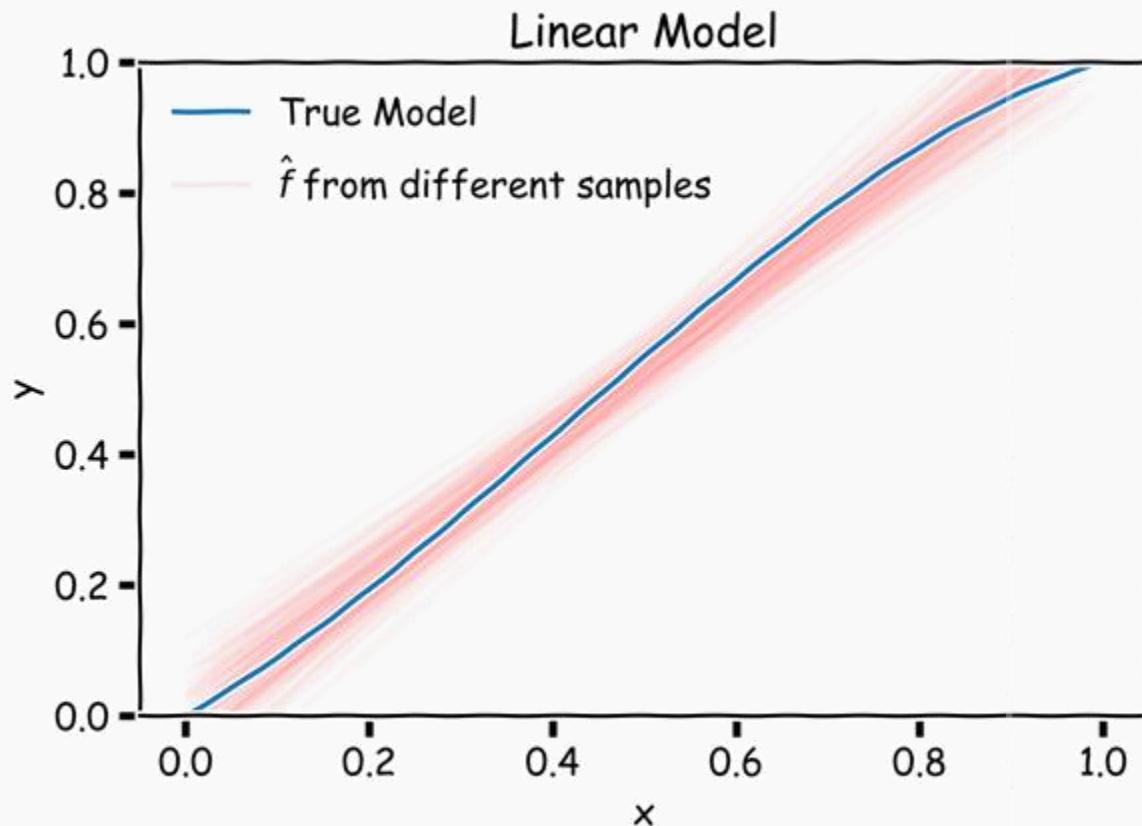
Bias vs Variance: Variance of a COMPLEX model



Bias vs Variance

Left: 2,000 best-fit linear models, each fitted to a different 20-point training set.

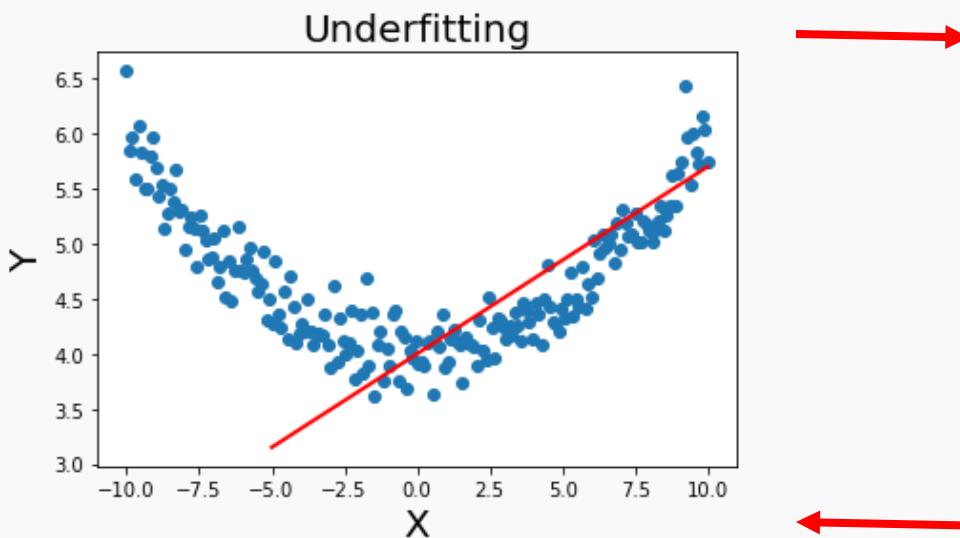
Right: 2,000 best-fit models using degree-10 polynomials.



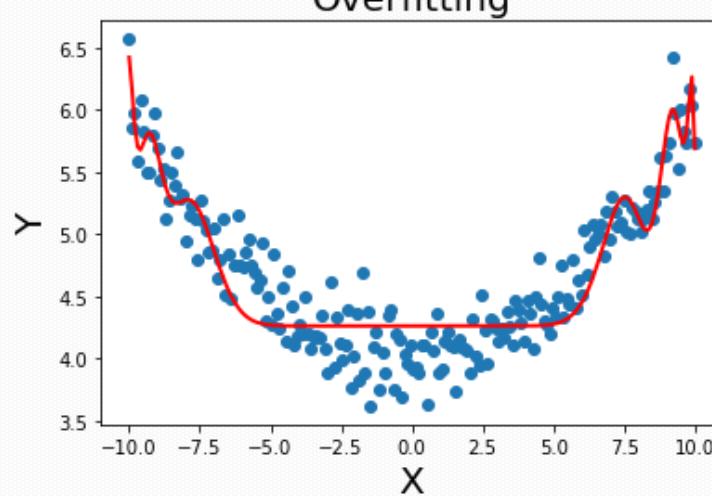
The Bias-Variance: Bias

Reducible error comes from either underfitting or overfitting. There is a tradeoff between the two sources of errors:

Increase complexity

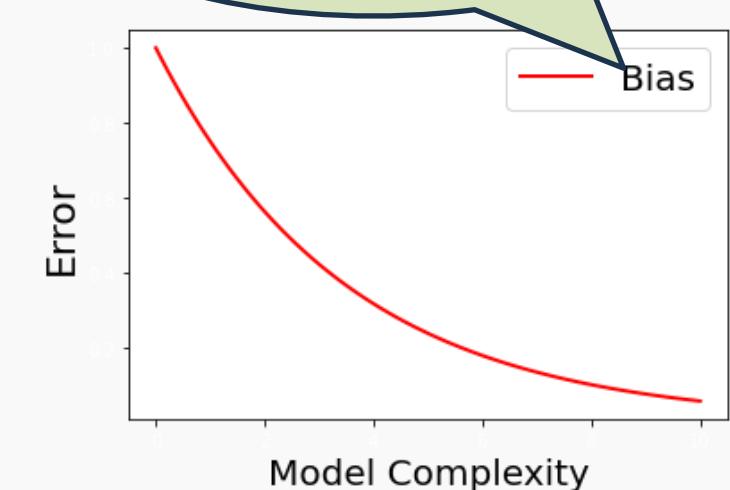


Overfitting



Decrease complexity

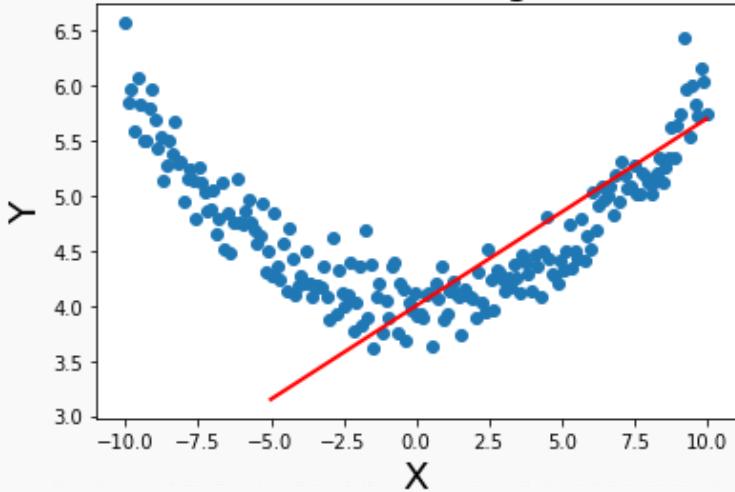
"bias" refers to how far off a model's predictions are from the actual truth.



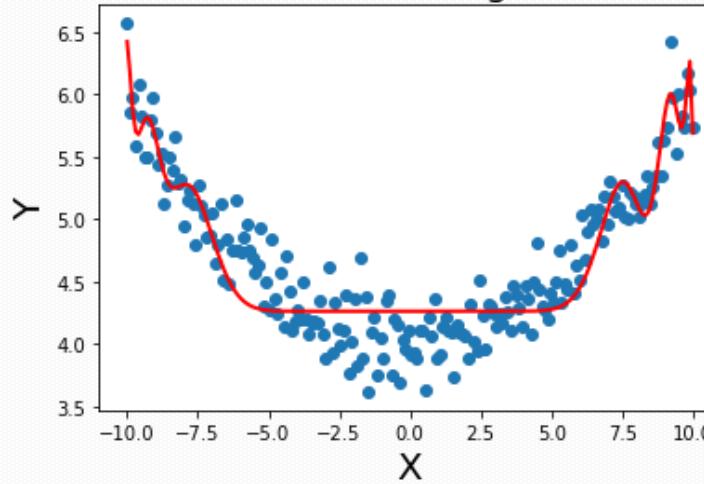
The Bias-Variance Trade Off

Increase complexity

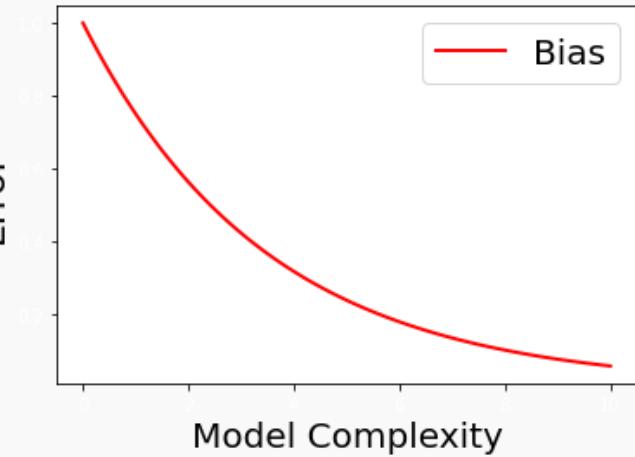
Underfitting



Overfitting

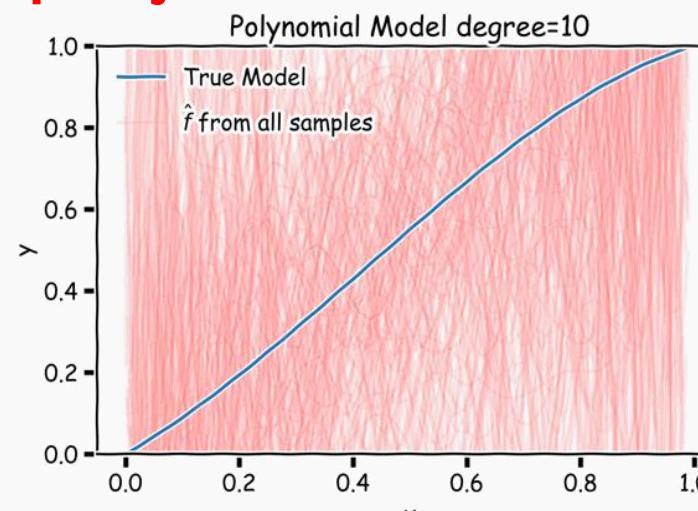
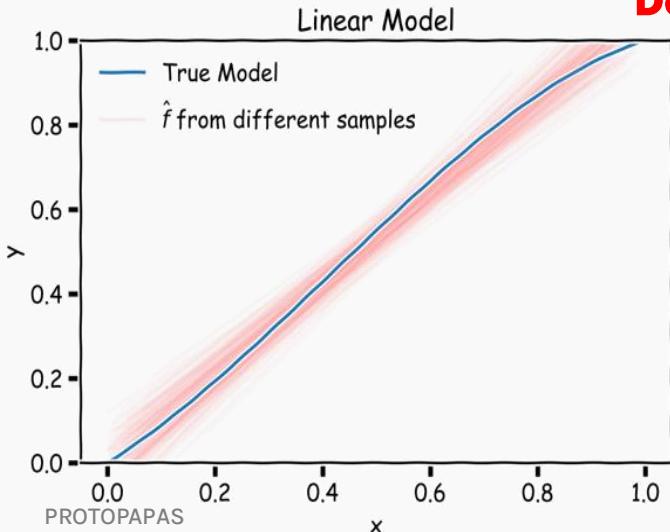


Error

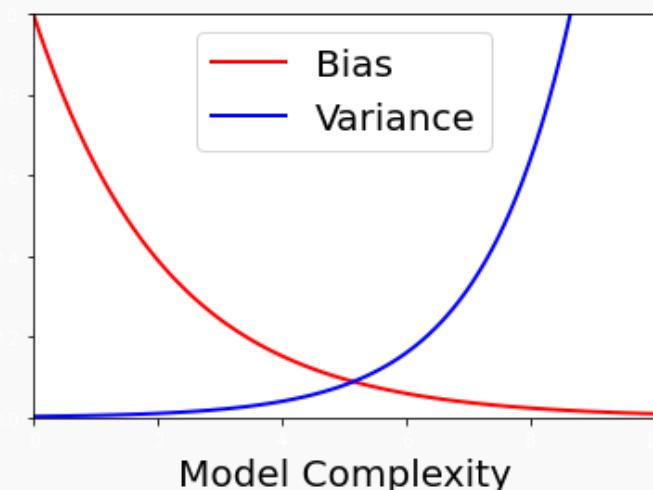


Model Complexity

Decrease complexity



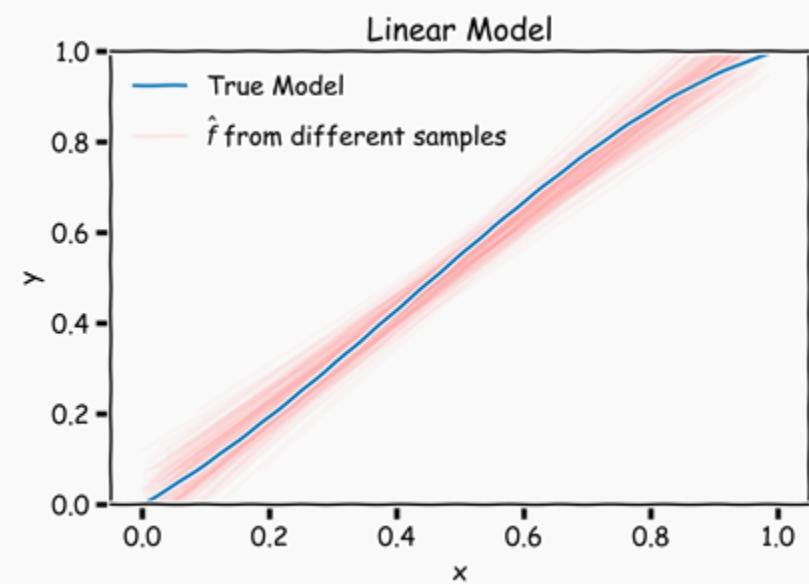
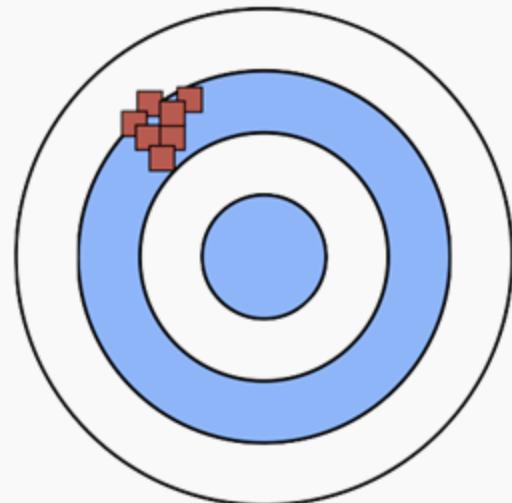
Error



Model Complexity

Low Variance (Precise)

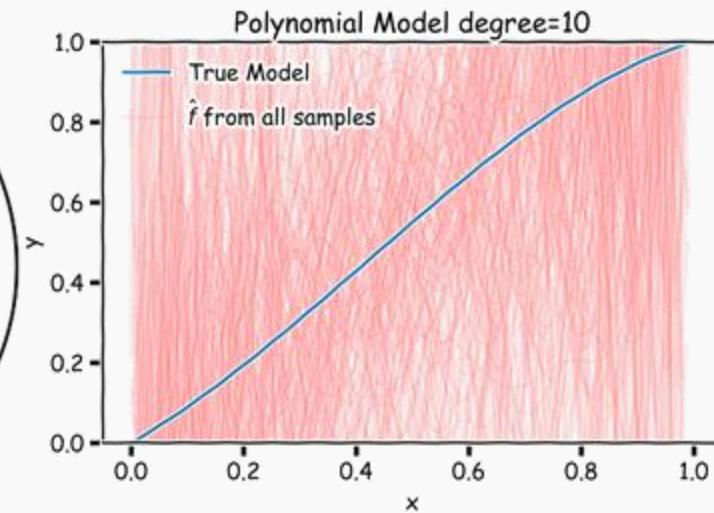
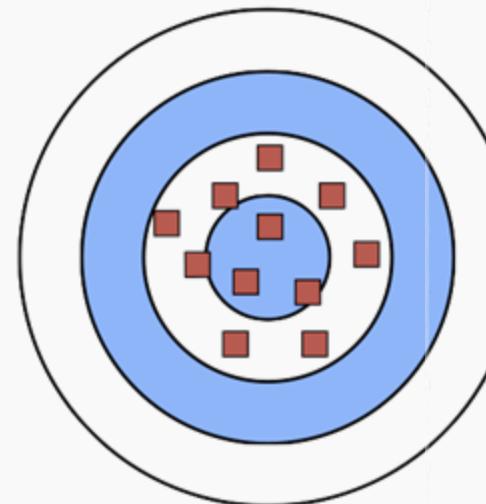
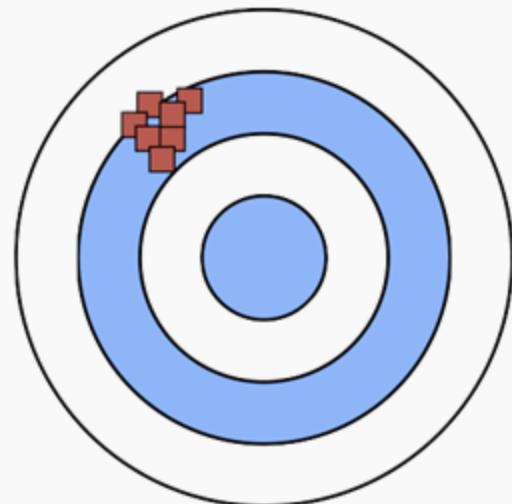
High Bias
(Not Accurate)



Low Variance
(Precise)

Low Bias
(Accurate)

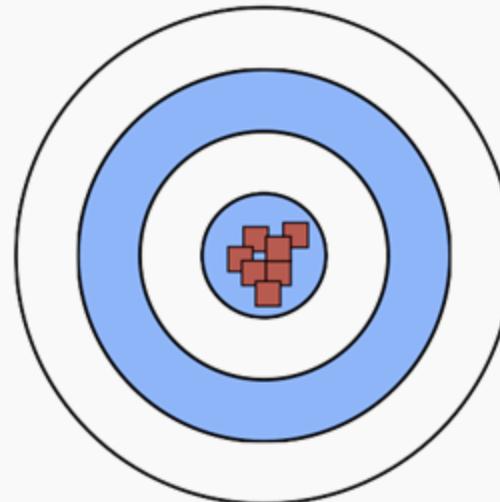
High Variance
(Not Precise)



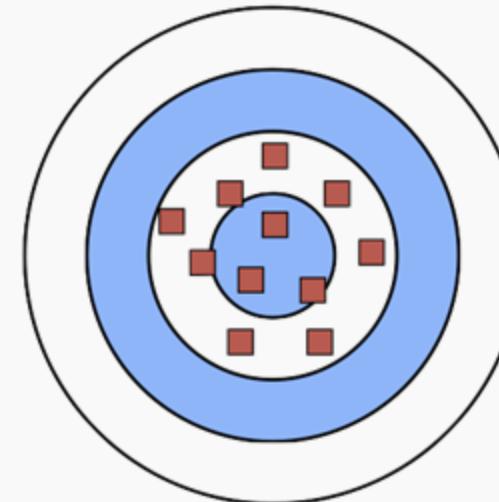
**WE WANT
THIS**

Low Bias
(Accurate)

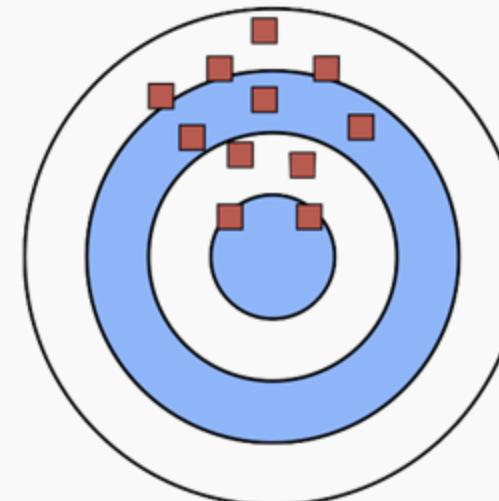
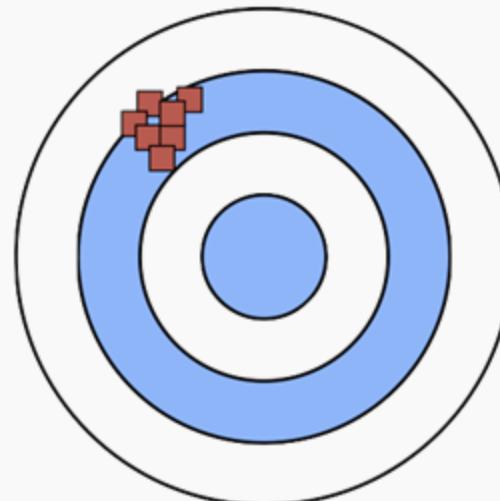
Low Variance
(Precise)



High Variance
(Not Precise)



High Bias
(Not Accurate)



**WE WANT TO
AVOID THIS**

Overfitting

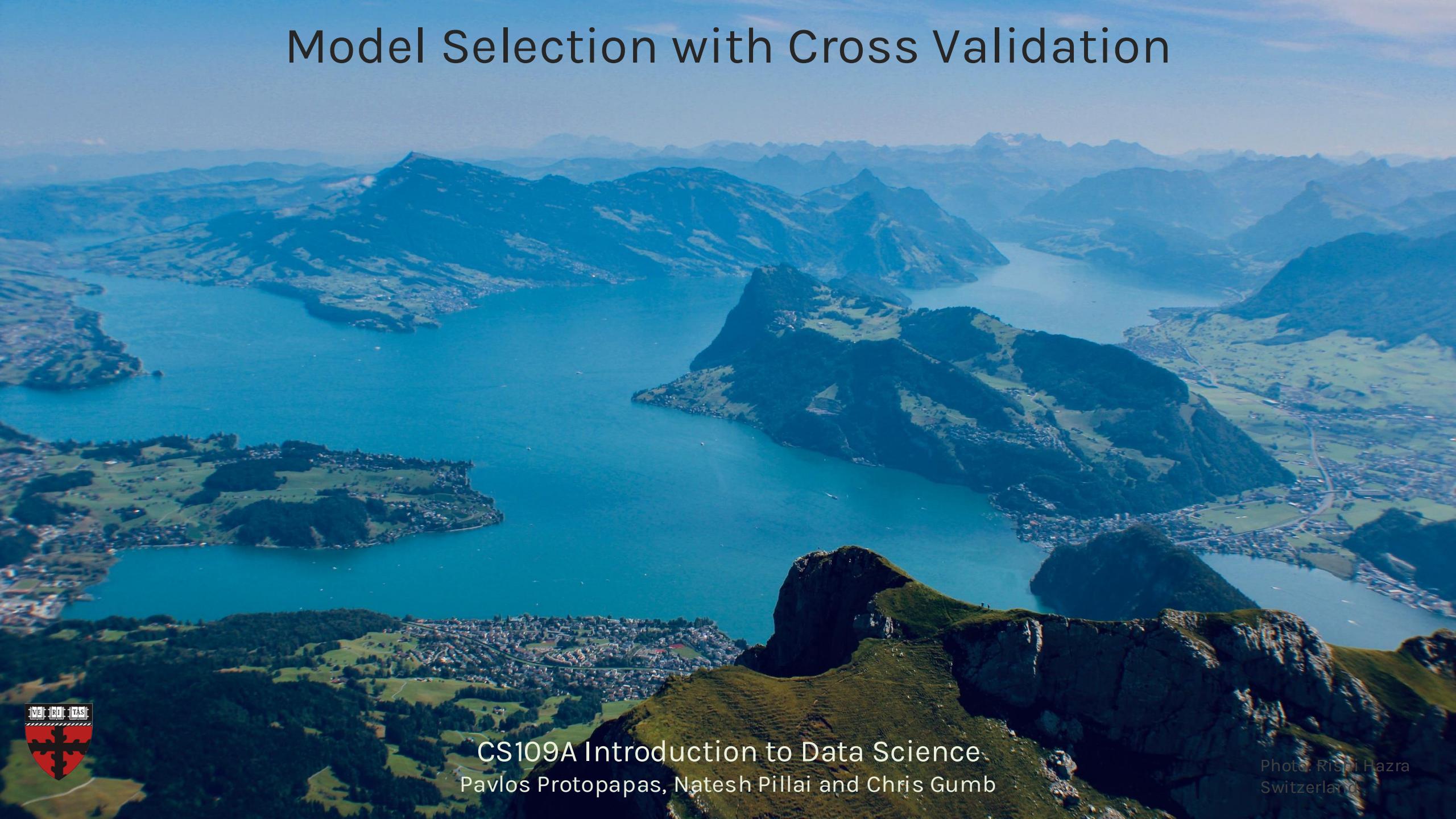
Overfitting occurs when a model corresponds too closely to the training set, and as a result, the model fails to fit additional data.

So far, we have seen that overfitting can happen when:

- too many parameters
- the degree of the polynomial is too large
- too many interaction terms

Soon, we will see other evidence of overfitting, which will point to a way of avoiding overfitting: **Ridge and Lasso regressions**.

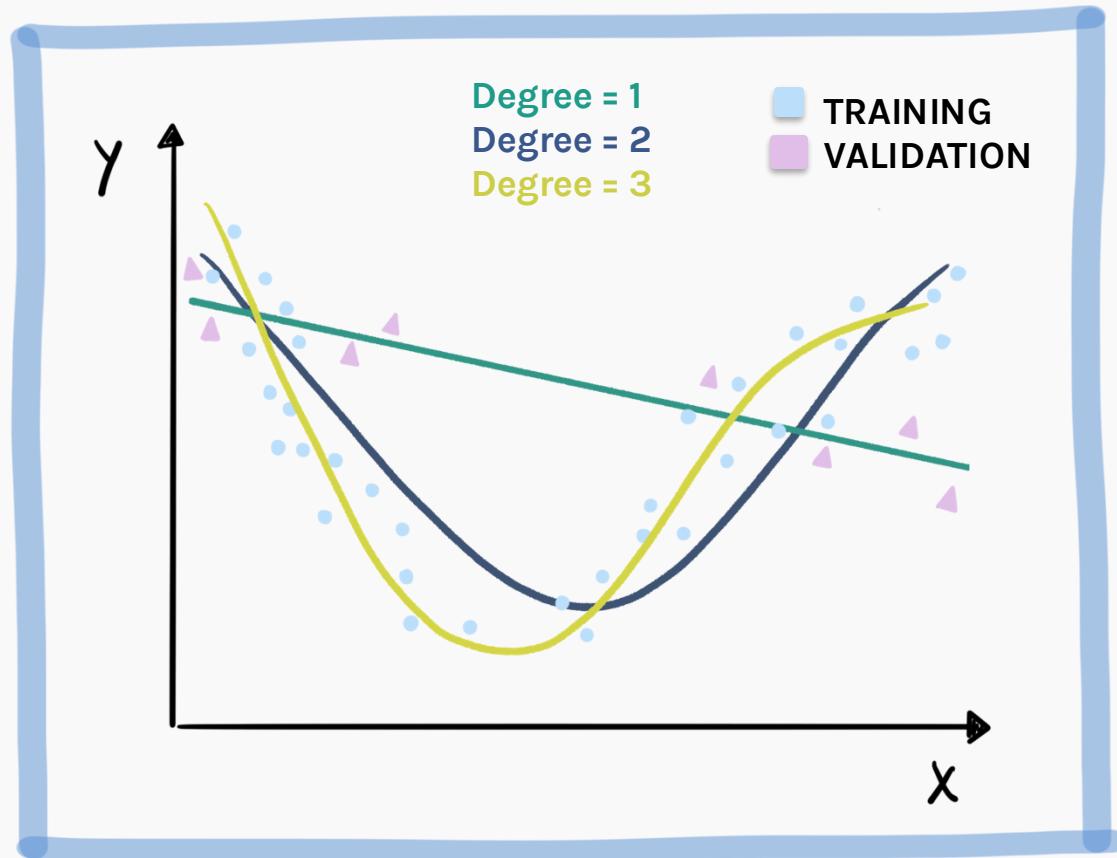
Model Selection with Cross Validation



CS109A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai and Chris Gumb

Photo: Rishi Hazra
Switzerland

Cross Validation: Motivation



It is obvious that degree=3 is the correct model, but the validation set by chance favors the linear model.

Using a **single validation set** to select amongst multiple models can be **problematic - there is the possibility of overfitting to the validation set.**

Cross Validation: Motivation

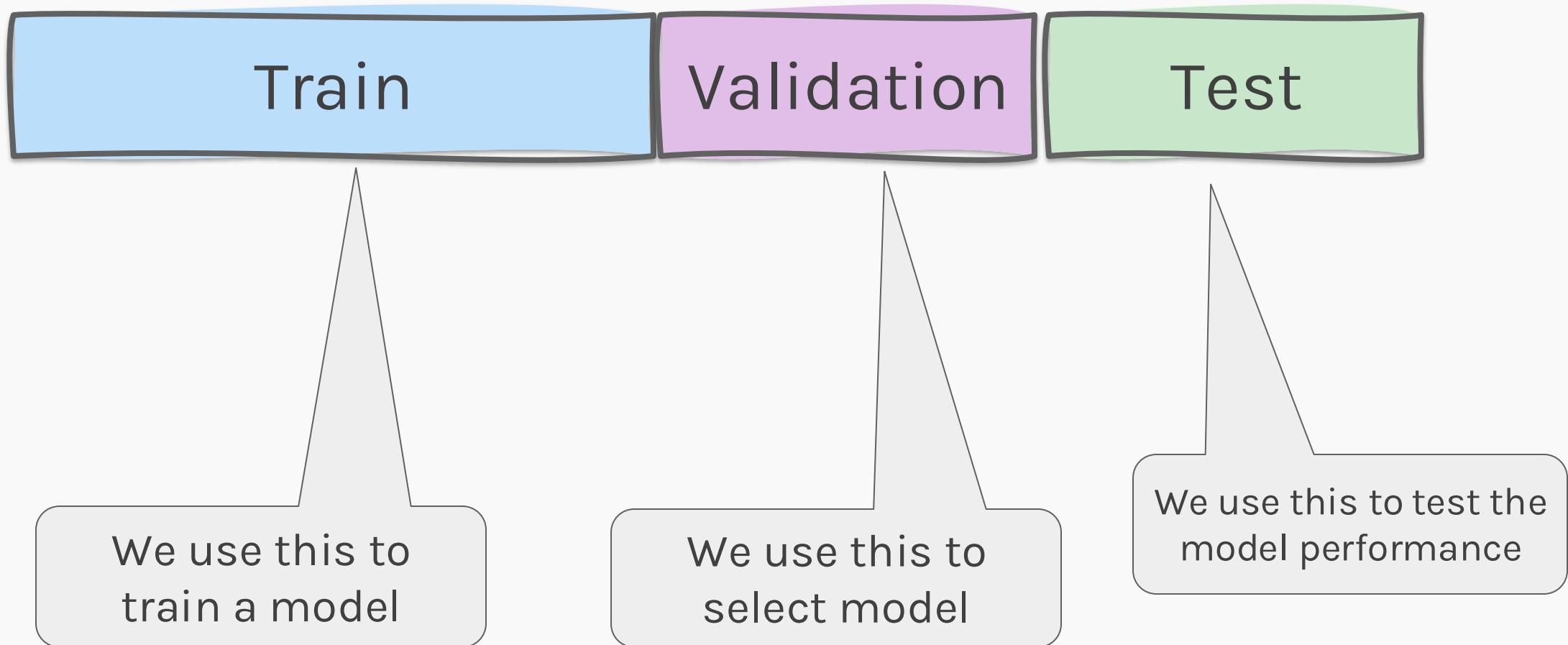
Using a **single validation set** to select amongst multiple models can be problematic - there is the possibility of overfitting to the validation set.

One solution to the problems raised by using a single validation set is to evaluate each model on **multiple** validation sets and **average** the validation performance.

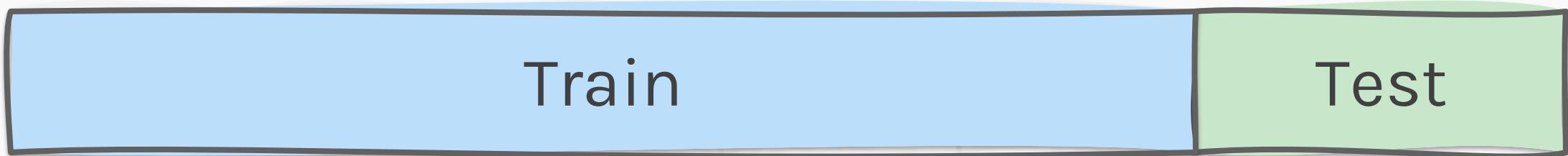
One can randomly split the training set into training and validation multiple times **but** randomly creating these sets can create the scenario where important features of the data never appear in our random draws.

Train-Validation-Test

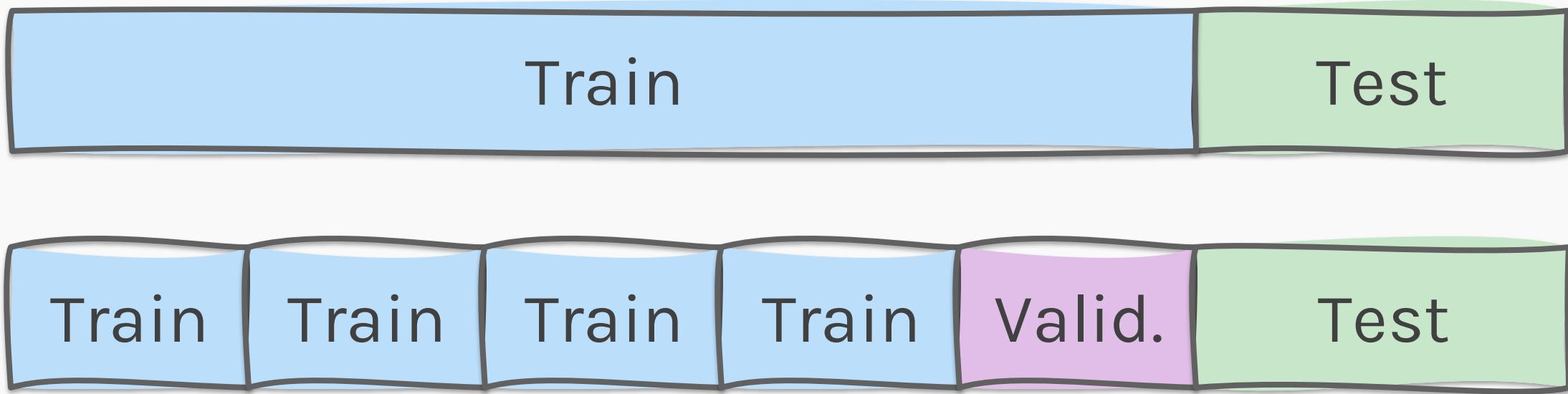
We introduced a different sub-set, which we called validation and we use it to select the model.



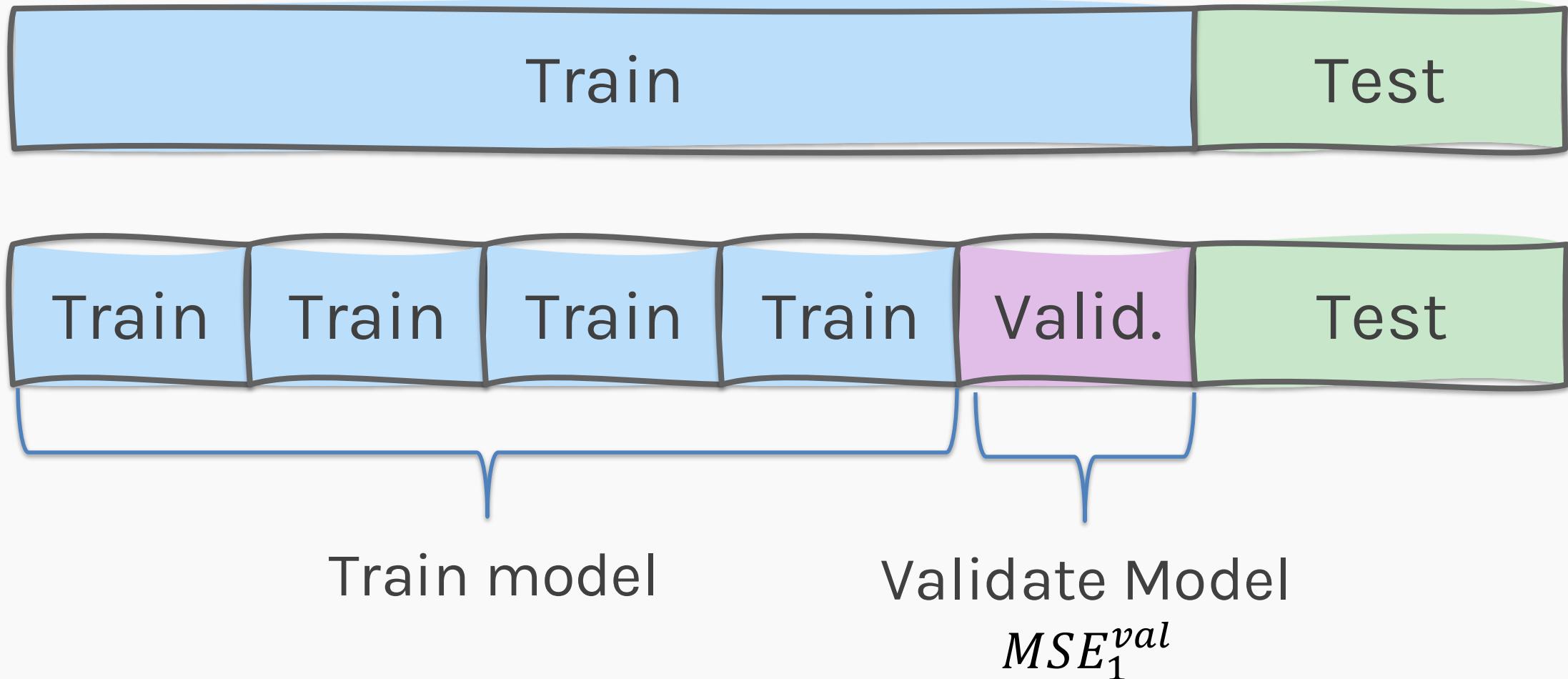
Cross Validation



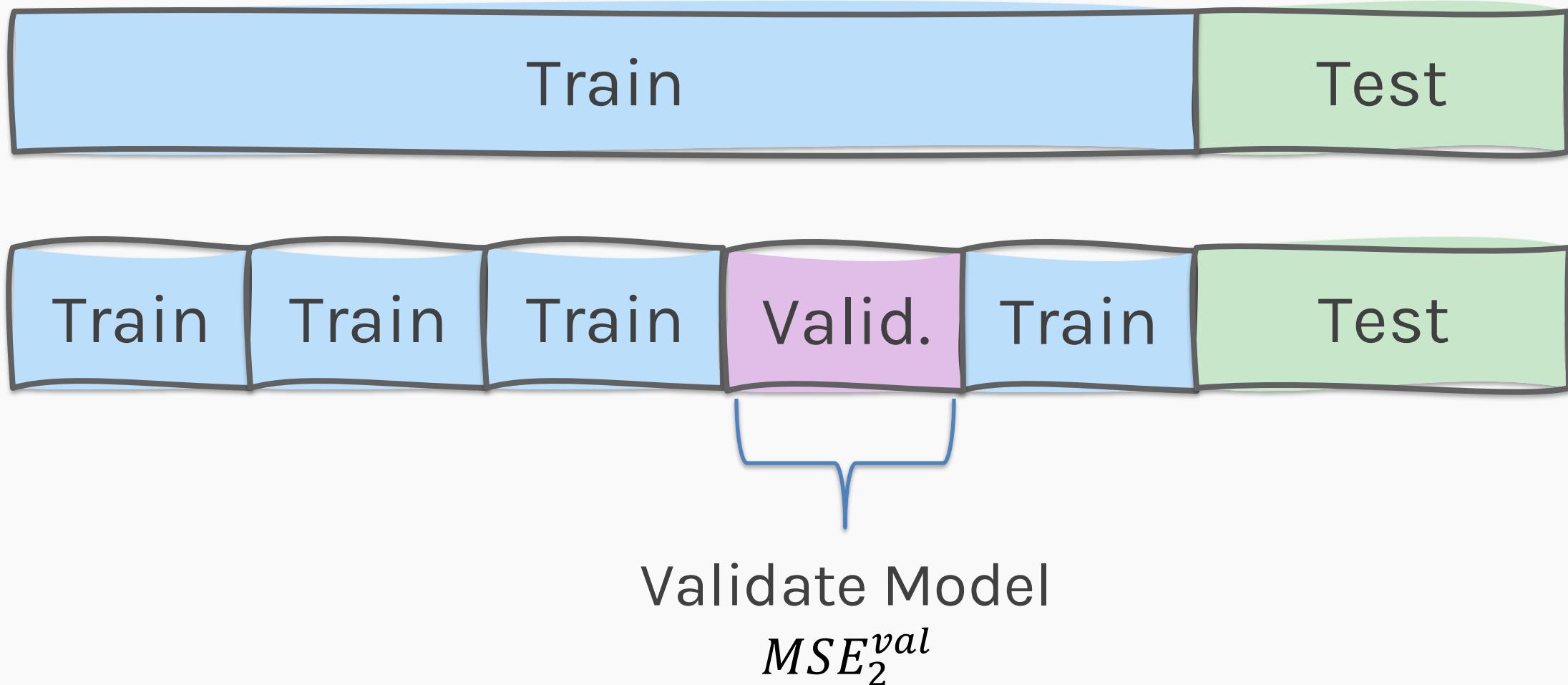
Cross Validation



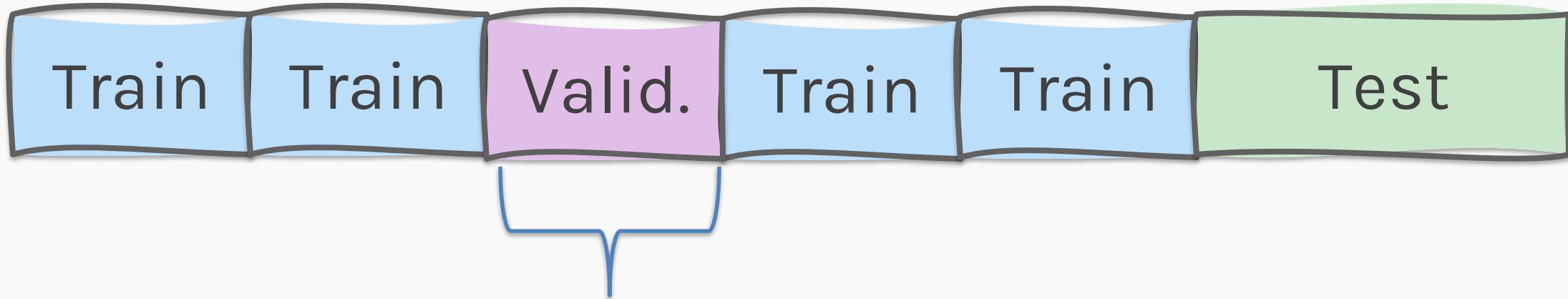
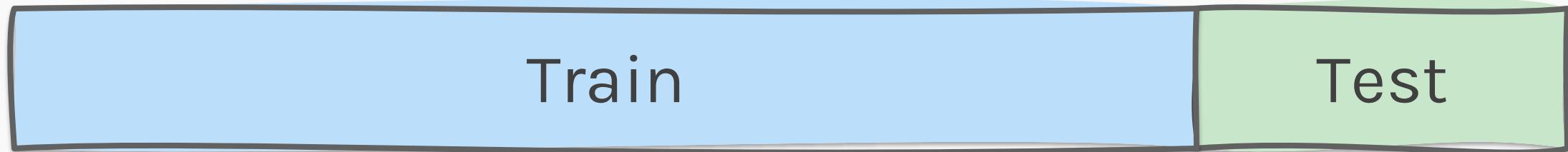
Cross Validation



Cross Validation



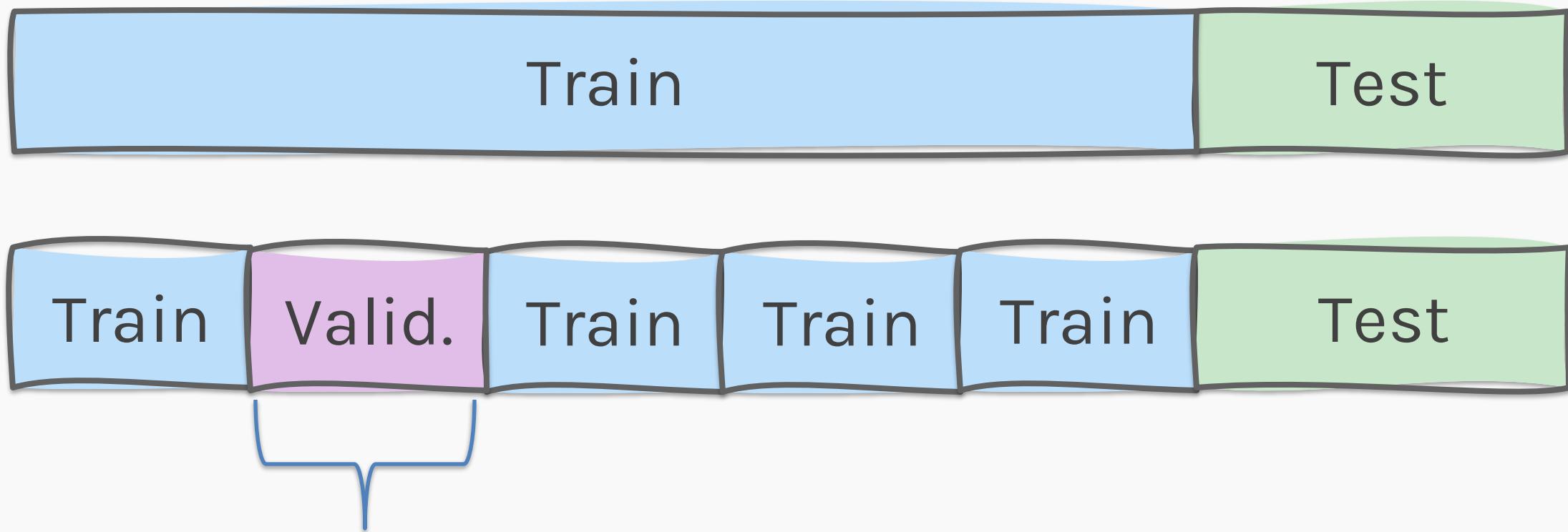
Cross Validation



Validate Model

$$MSE_3^{val}$$

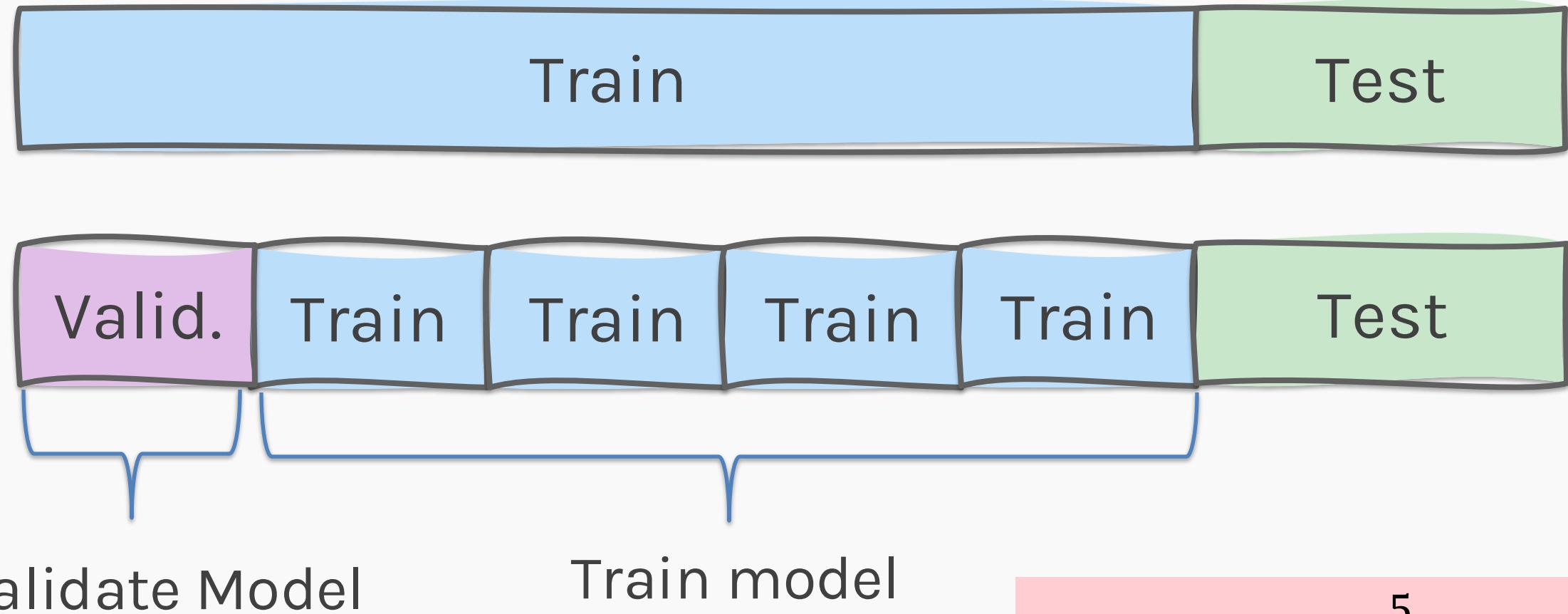
Cross Validation



Validate Model

$$MSE_4^{val}$$

Cross Validation



$$MSE_5^{val}$$

$$MSE^{val} = \frac{1}{5} \sum_{i=1}^5 MSE_i^{val}$$

K-Fold Cross Validation

Given a data set $\{X_1, \dots, X_n\}$, containing J features.

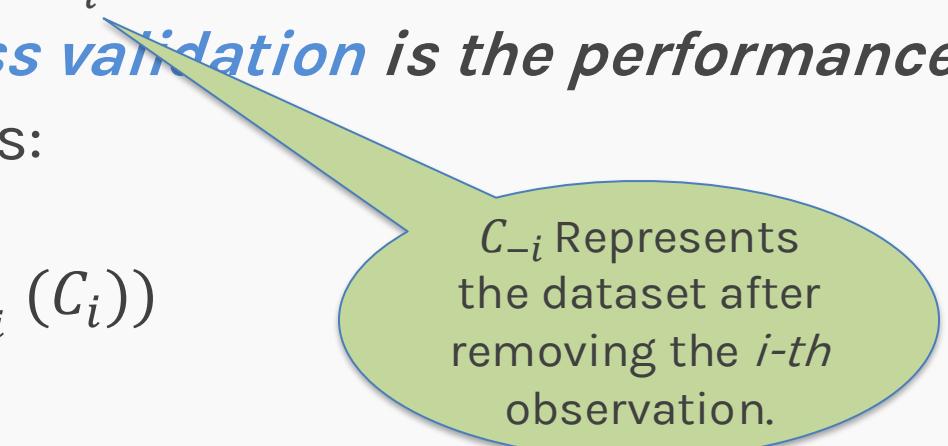
To ensure that every observation in the dataset is included in at least one training set and at least one validation set we use the ***K-fold validation***:

- split the data into K uniformly sized chunks, $\{C_1, \dots, C_K\}$
- we create K number of training/validation splits, using one of the K chunks for validation and the rest for training.

We fit the model on each training set, denoted $\hat{f}_{C_{-i}}$, and evaluate it on the corresponding validation set, $\hat{f}_{C_{-i}}(C_i)$. The ***cross validation is the performance*** of the model averaged across all validation sets:

$$CV(\text{Model}) = \frac{1}{K} \sum_{i=1}^K L(\hat{f}_{C_{-i}}(C_i))$$

where L is a **loss function**.



C_{-i} Represents the dataset after removing the i -th observation.

Leave-One-Out

Or using the **leave one out** method:

- validation set: $\{X_i\}$
- training set: $X_{-i} = \{X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_n\}$

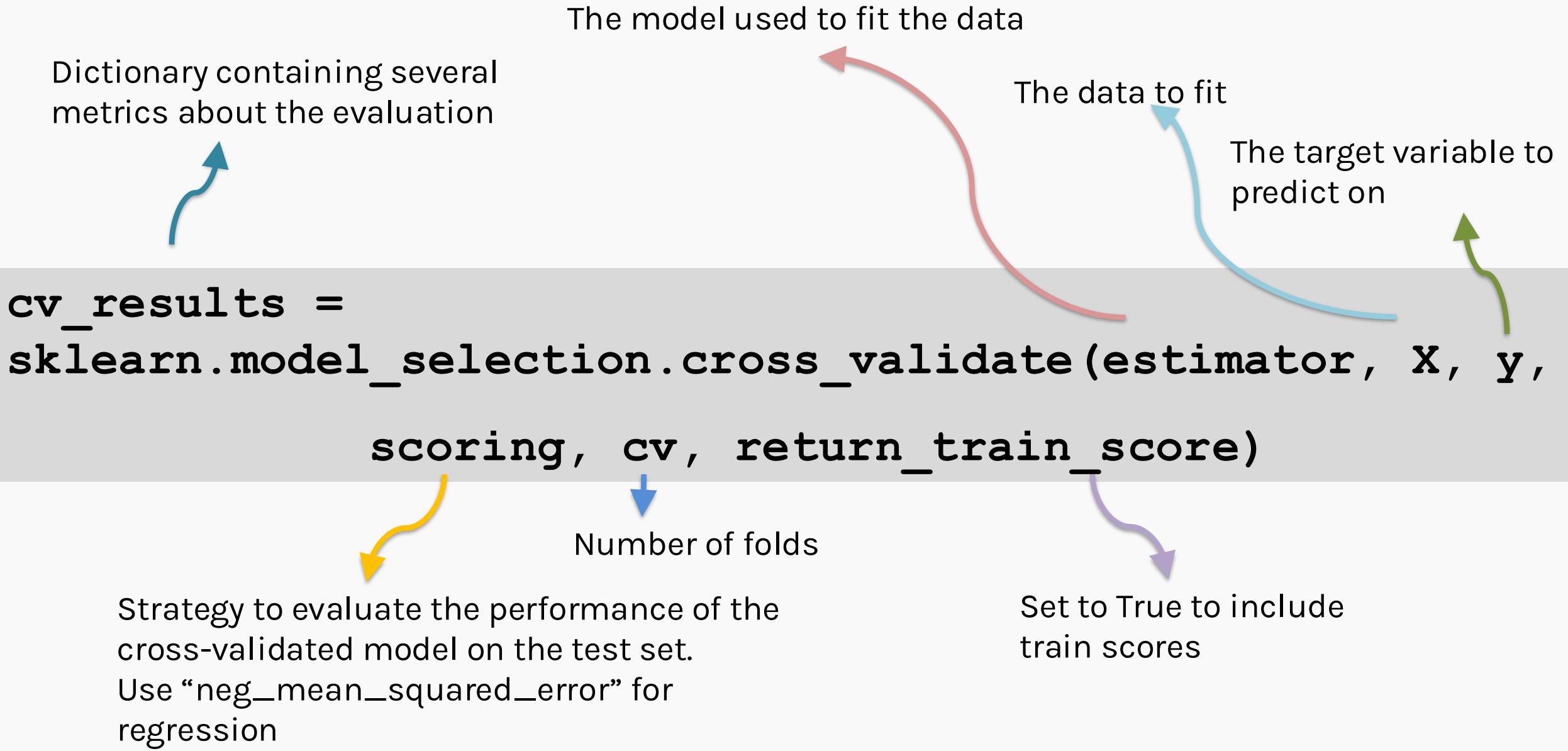
for $i = 1, \dots, n$:

We fit the model on each training set, denoted $\hat{f}_{X_{-i}}$, and evaluate it on the corresponding validation set, $\hat{f}_{X_{-i}}(X_i)$.

The **cross validation score** is the performance of the model averaged across all validation sets:

$$CV(\text{Model}) = \frac{1}{n} \sum_{i=1}^n L(\hat{f}_{X_{-i}}(X_i))$$

where L is a loss function.



Dictionary containing several metrics about the evaluation



```
cv_results =
```

```
sklearn.model_selection.cross_val_score(...,
```

```
scoring,
```

```
cv=5, return_train_score)
```

Number of folds

Strategy to evaluate the performance of the cross-validated model on the test set.



Use “`neg_mean_squared_error`” for regression

The model us

scikit-learn's cross-validation framework is designed to maximize a scoring function.

In the case of regression problems, however, we usually want to minimize the error (such as mean squared error or MSE), not maximize it.

able to



Set to True to include train scores





Model Selection

CS109A Introduction to Data Science

Pavlos Protopapas, Natesh Pillai and Chris Gumb



Lecture Outline

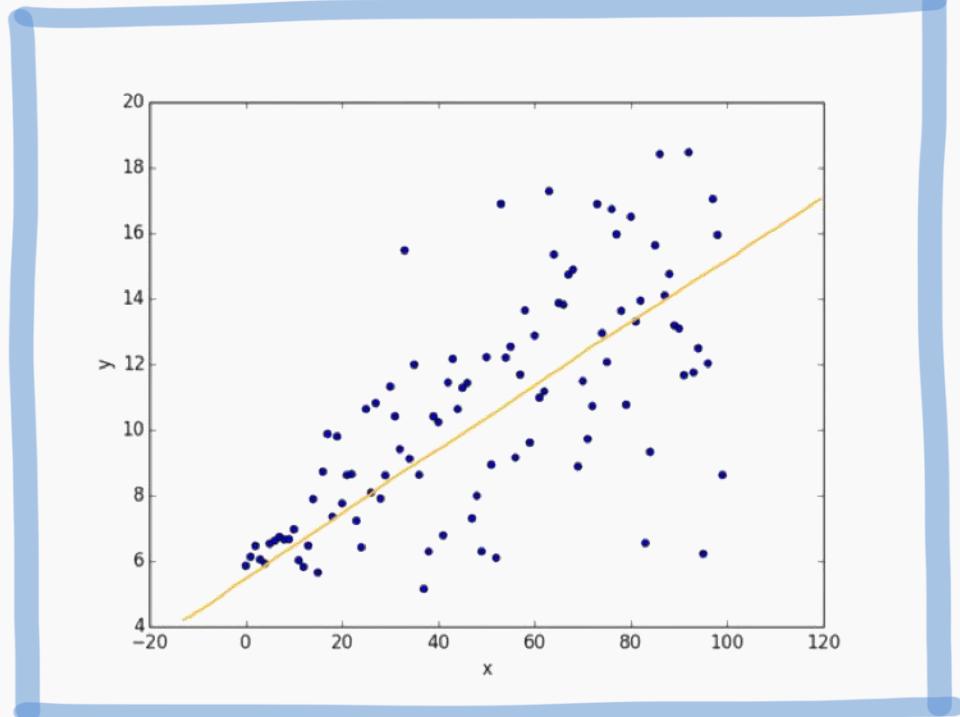
Interaction Effects in Regression Models

Polynomial Regression: Extending Linear Models

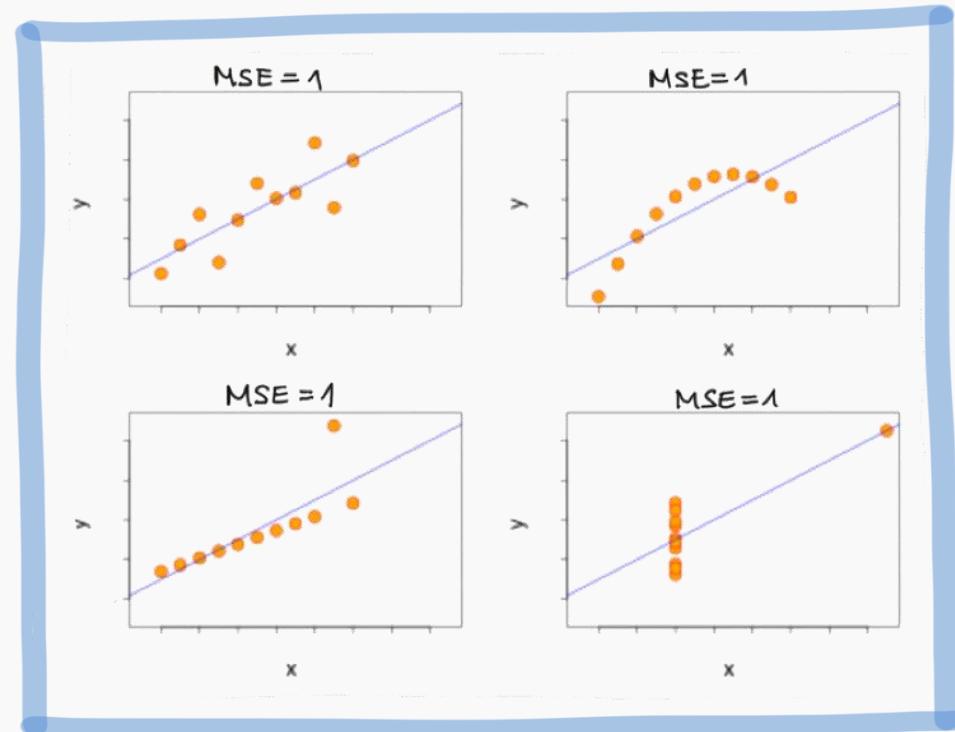
Model Selection Techniques: Focus on Cross-Validation

Evaluation: Training Error

Just because we found the model that minimizes the squared error it doesn't mean that it's a good model. We could investigate the R^2 but also:



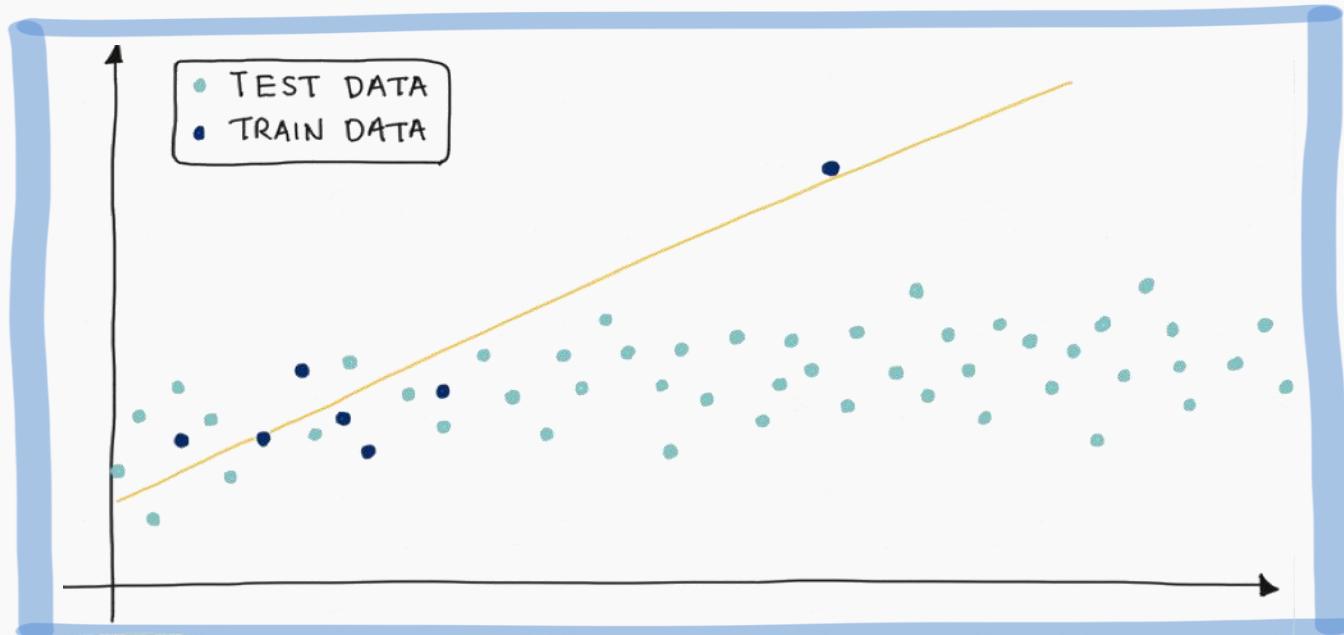
The MSE is high due to noise in the data.



The MSE is high in all four models, but the models are not equal.

Evaluation: Test Error

We need to evaluate the fitted model on new data, data that the model did not train on, the **test data**.



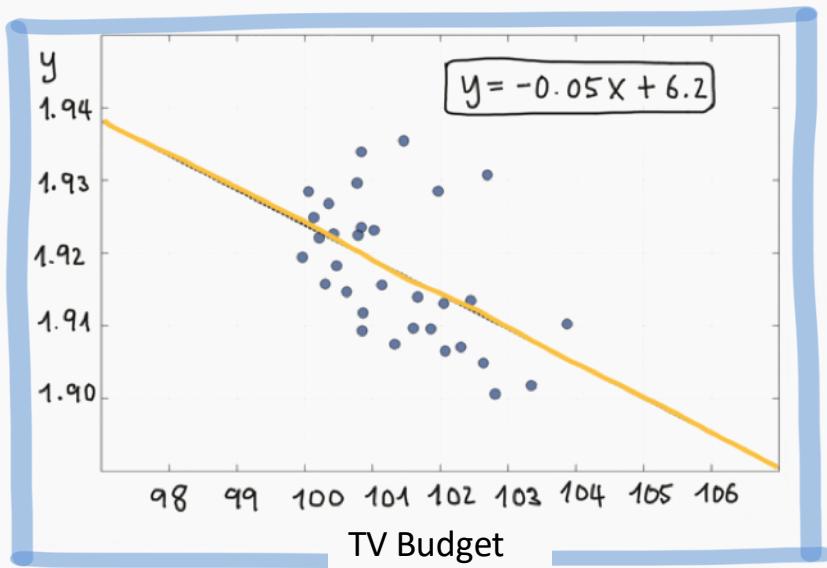
The **training** MSE here is 2.0 where the **test** MSE is 12.3.

The training data contains a strange point – an outlier – which confuses the model.

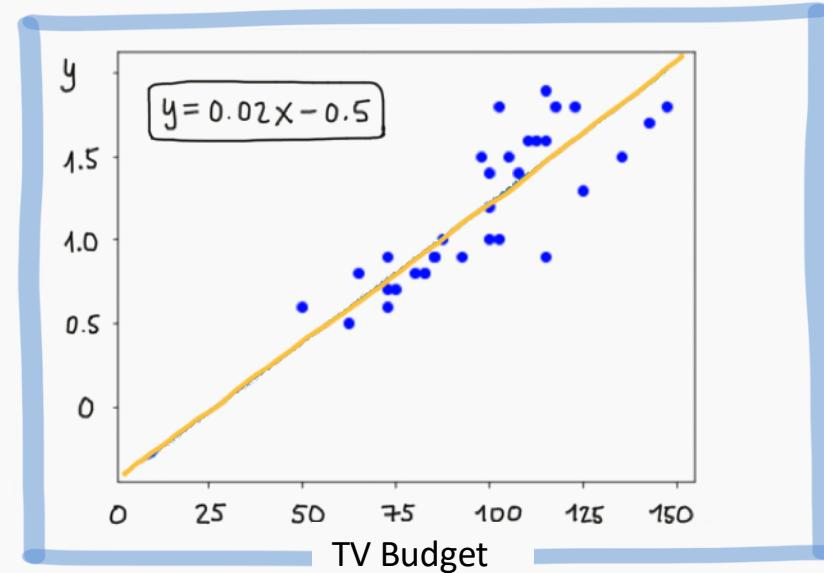
Fitting to meaningless patterns in the training is called **overfitting**.

Evaluation: Model Interpretation

For linear models it's important to interpret the parameters



The MSE of this model is very small. But the slope is -0.05. That means the larger the budget the less the sales.



The MSE is very small, but the intercept is -0.5 which means that for very small budget we will have negative sales.

Generalization Error

We know to evaluate the model on both train and test data, because models that do well on training data may do poorly on new data (overfitting).

The ability of models to do well on new data is called **generalization**.

The goal of **model selection** is to choose the model that generalizes the best.

Model Selection

Model selection is the application of a principled method to determine the complexity of the model, e.g., choosing a subset of predictors, choosing the degree of the polynomial model etc.

A strong motivation for performing model selection is to avoid **overfitting**, which we saw can happen when:

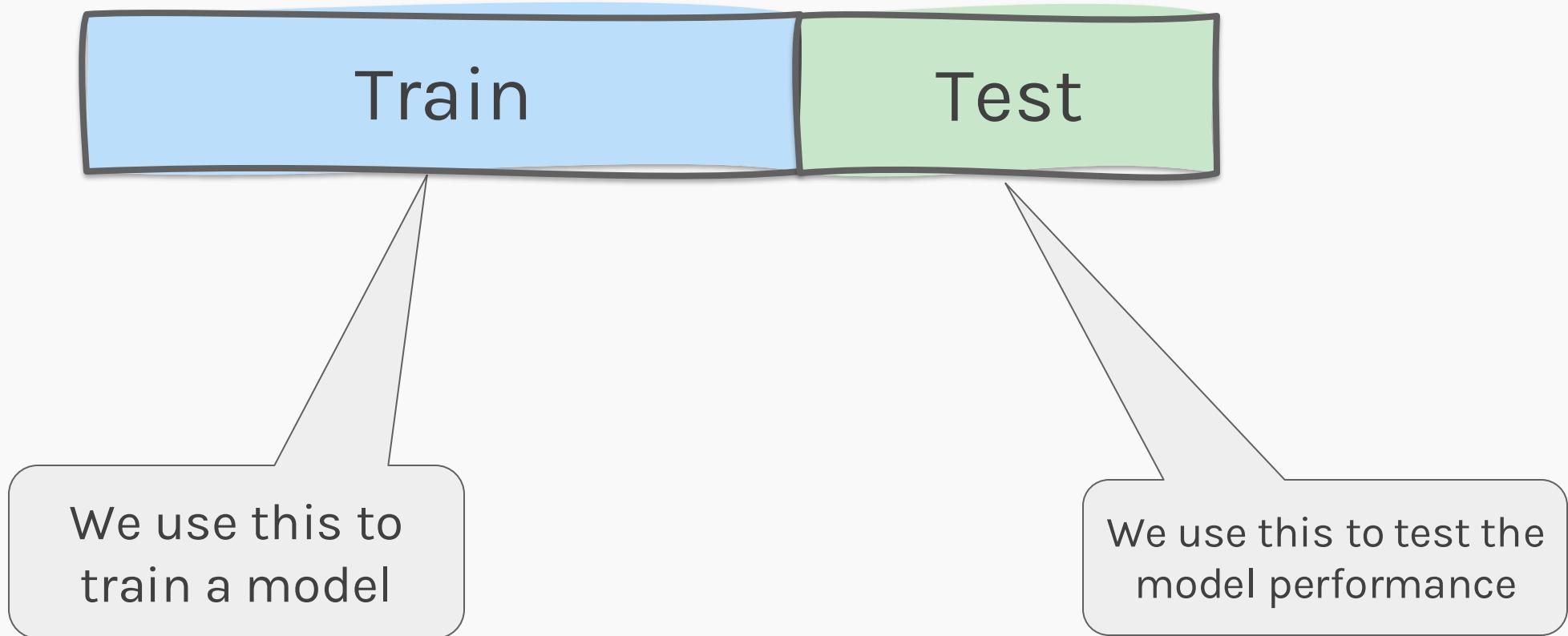
- there are too many predictors:
 - the feature space has high dimensionality
 - the polynomial degree is too high
 - too many cross terms are considered
- the coefficients values are too **extreme (we have not seen this yet)**

Train-Test split

How do we select a model?



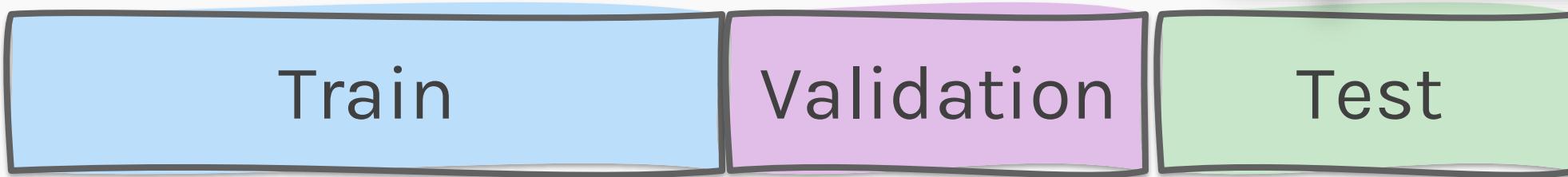
So far, we have been using train/test splits



Train-Validation-Test

We introduce a different sub-set, which we called validation set to select the model.

The test set should never be touched for model training or selection.



We use this to train a model

We use this to select model

We use this to test the model performance

Model Selection

Ways of model selection:

- Exhaustive search
- Greedy algorithms
- Fine tuning hyper-parameters
- Regularization

Model Selection

Ways of model selection:

- **Exhaustive search**
- Greedy algorithms
- Fine tuning hyper-parameters
- Regularization

Model Selection: How many models?

Can you prove this?



Question:

How many different models when considering J predictors (only linear terms) do we have?

Example: 3 predictors (X_1, X_2, X_3)

- Models with 0 predictor:

M0:

- Models with 1 predictor:

M1: X_1

M2: X_2

M3: X_3

- Models with 2 predictors:

M4: $\{X_1, X_2\}$

M5: $\{X_2, X_3\}$

M6: $\{X_3, X_1\}$

- Models with 3 predictors:

M7: $\{X_1, X_2, X_3\}$



2^J models

Model Selection

Ways of model selection:

- Exhaustive search
- **Greedy algorithms**
- Fine tuning hyper-parameters
- Regularization

Stepwise Variable Selection and Validation

Selecting optimal subsets of predictors (including choosing the degree of polynomial models) through:

- stepwise variable selection - **iteratively building** an optimal subset of **predictors** by optimizing a fixed model evaluation metric each time.
- **selecting** an optimal **model** by evaluating each model on validation set.

Stepwise Variable Selection: Forward method

In **forward selection**, we find an ‘optimal’ set of predictors by iterative building up our set.

1. Start Simple

Begin with the empty set P_0 , construct the null model M_0

2. Add Predictors One by One

For $k = 1, \dots, J$:

2.1 Use the best model so far, M_{k-1} , based on $k - 1$ predictors

2.2 Pick one predictor not yet P_{k-1} that improves the model the most according to validation MSE

2.3 Create a new model, M_k , using the selected predictors

3. Select the model M amongst $\{M_0, M_1, \dots, M_J\}$ that optimizes validation MSE

Stepwise Variable Selection Computational Complexity

How many models did we evaluate?

- 1st step, **J Models**
- 2nd step, **$J-1$ Models** (add 1 predictor out of $J-1$ possible)
- 3rd step, **$J-2$ Models** (add 1 predictor out of $J-2$ possible)
- ...

$$O(J^2) \ll 2^J \text{ for large } J$$

Model Selection

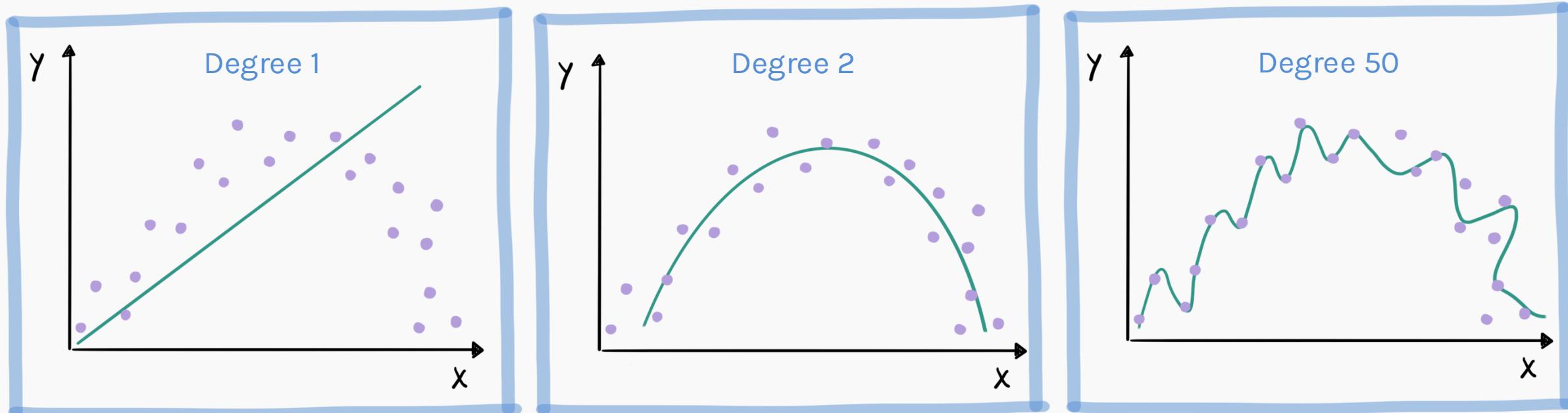
Ways of model selection:

- Exhaustive search
- Greedy algorithms
- **Fine tuning hyper-parameters**
- Regularization



Choosing the degree of the polynomial model

We turn model selection into choosing a **hyper-parameter**. For example, polynomial regression requires choosing a degree – this can be thought as model selection – and we select the model by tuning the hyper-parameter.

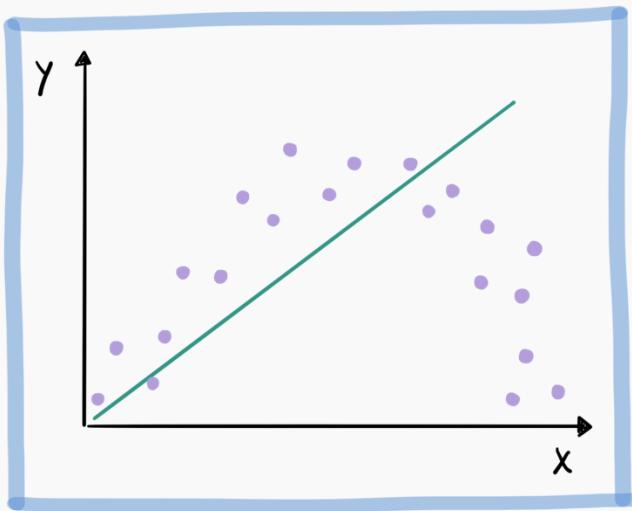


Underfitting: when the degree is too low, the model cannot fit the trend.

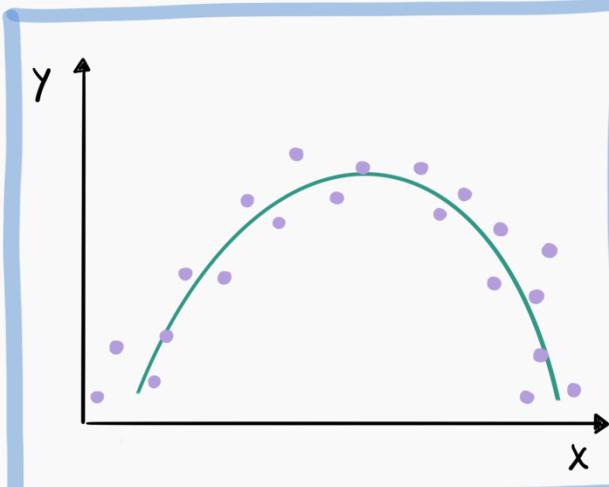
We want a model that fits the trend and ignores the noise.

Overfitting: when the degree is too high, the model fits all the noisy data points.

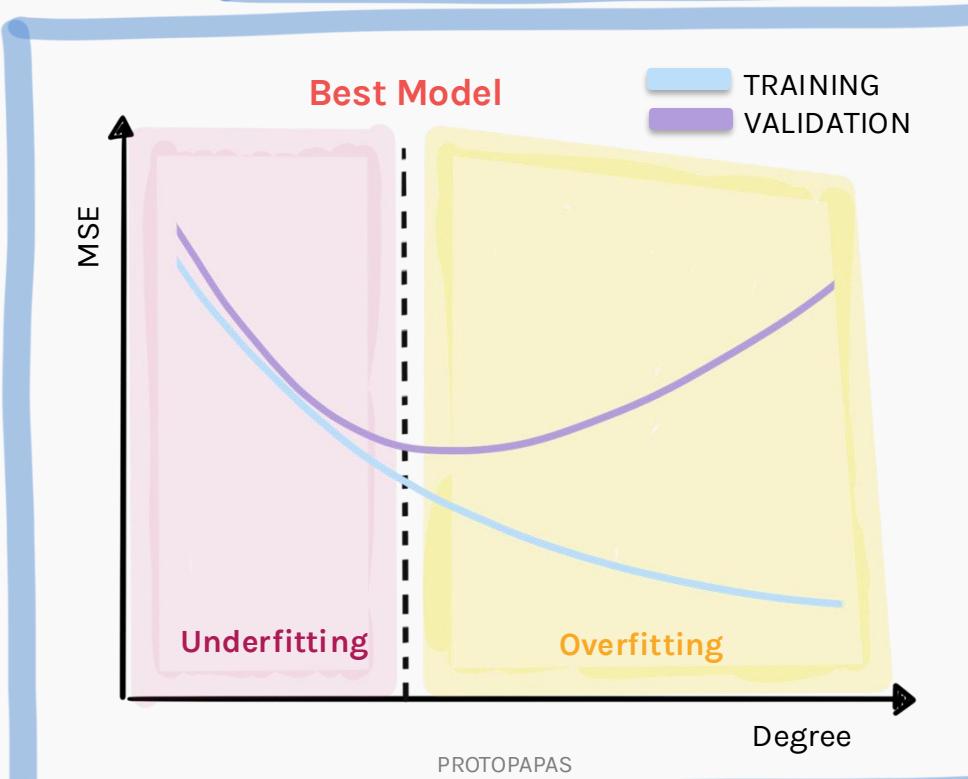
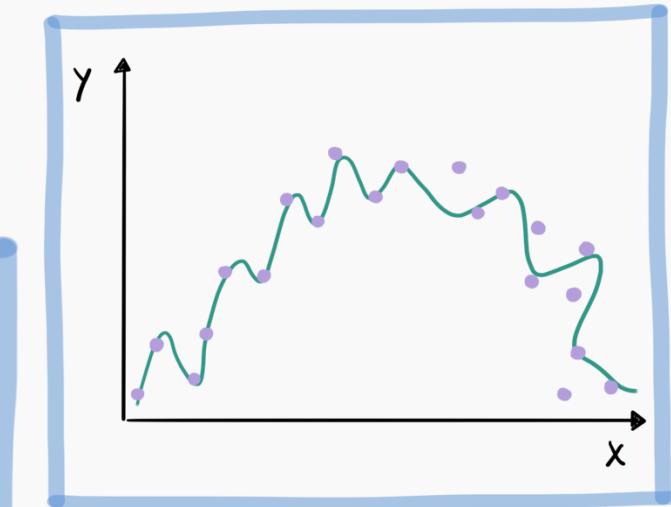
Underfitting: train and validation error is high.



Best model: validation error is minimum.



Overfitting: train error is low, validation error is high.



What are the parameters of the models and what are the hyperparameters?

Interaction Terms and Polynomial Regression

CS109A Introduction to Data Science

Pavlos Protopapas, Natesh Pillai and Chris Gumb



Lecture Outline

Interaction Effects in Regression Models

Polynomial Regression: Extending Linear Models

Model Selection Techniques: Focus on Cross-Validation

Too many predictors and collinearity leads to
OVERFITTING!

Game Time



If your model was a student, what would overfitting be like?

Options:

- A. Studying just the night before the test
- B. Memorizing every lecture, lab and OH word-for-word
- C. Only studying one chapter for all subjects
- D. Taking extensive notes but forgetting to actually understand the concepts

Game Time



If your model was a TF, what would overfitting be like?

Options:

- A. Grading papers while wearing 3D glasses to "see the errors in a new dimension"
- B. Using a "Magic 8-Ball" to decide students' grades
- C. Give everyone the first letter that comes on their name. Sorry Frank
- D. Subtract points for every answer that does not include the word overfitting

Too many predictors and collinearity and leads to
OVERFITTING!

Too many predictors and collinearity and leads to
OVERFITTING!

Overfitting occurs when a model learns the training data too well, including its noise and outliers, resulting in poor performance on new, unseen data.

Lecture Outline

Interaction Effects in Regression Models

Polynomial Regression: Extending Linear Models

Model Selection Techniques: Focus on Cross-Validation

Assumptions of Linear Regression

Linearity: Relationship between variables is linear.

$$f(x) = \beta_0 + \beta_1 x$$

Independence: No correlation between errors and predictors.

Homoscedasticity: Constant variance of residuals.

Normality of Residuals: Residuals are normally distributed.

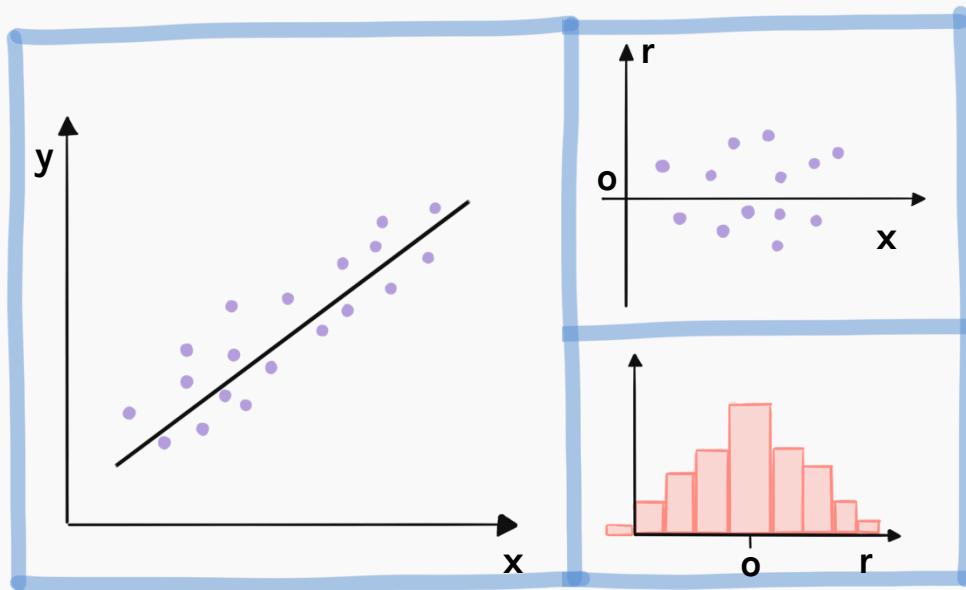
$$\begin{aligned}y &= f(x) + \epsilon \\L(\beta_0, \beta_1) &= MSE\end{aligned}$$

Other things to consider

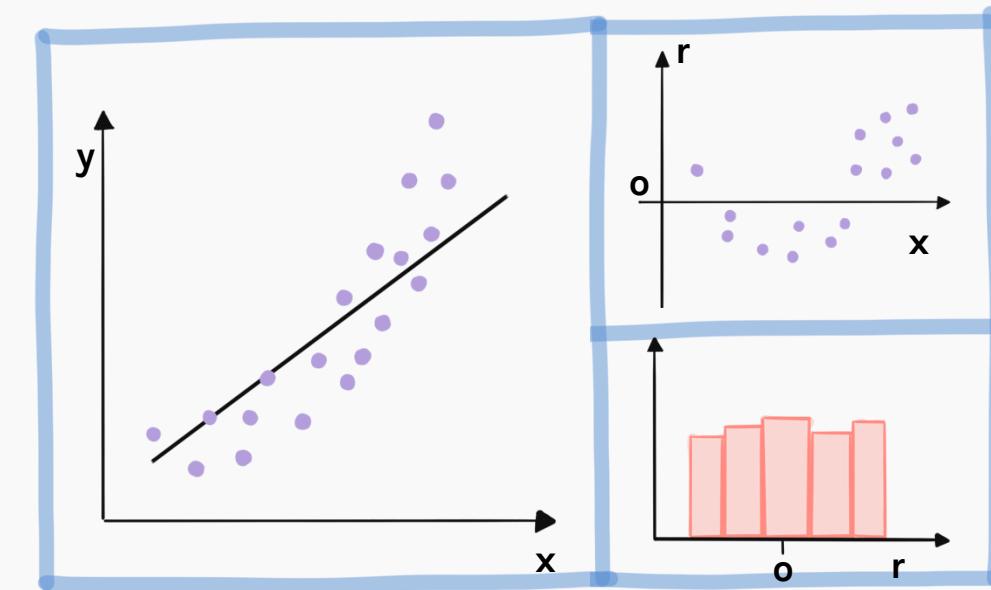
Fixed X: Independent variables are error-free.

No Multicollinearity: Low correlation between predictors.

Residual Analysis



Linear assumption is correct. There is no obvious relationship between residuals and x . Histogram of residuals is **symmetric** and **normally distributed**.



Linear assumption is incorrect. There is an obvious relationship between residuals and x . Histogram of residuals is symmetric but **not normally distributed**.

Note: For multi-regression, we plot the residuals vs predicted y, \hat{y} , since there are too many x 's and that could wash out the relationship.

Beyond linearity: synergy effect or interaction effect

We assumed that the average effect on *sales* of a one-unit increase in *TV*, is always β_1 regardless of the amount spent on *radio* or *newspaper*.

Synergy effect or **interaction effect** states that when an increase on the *radio budget* affects the effectiveness of the *TV* spending on *sales*.

To account for it, we simply add a term as:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2$$

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \boxed{\beta_3 X_1 X_2}$$

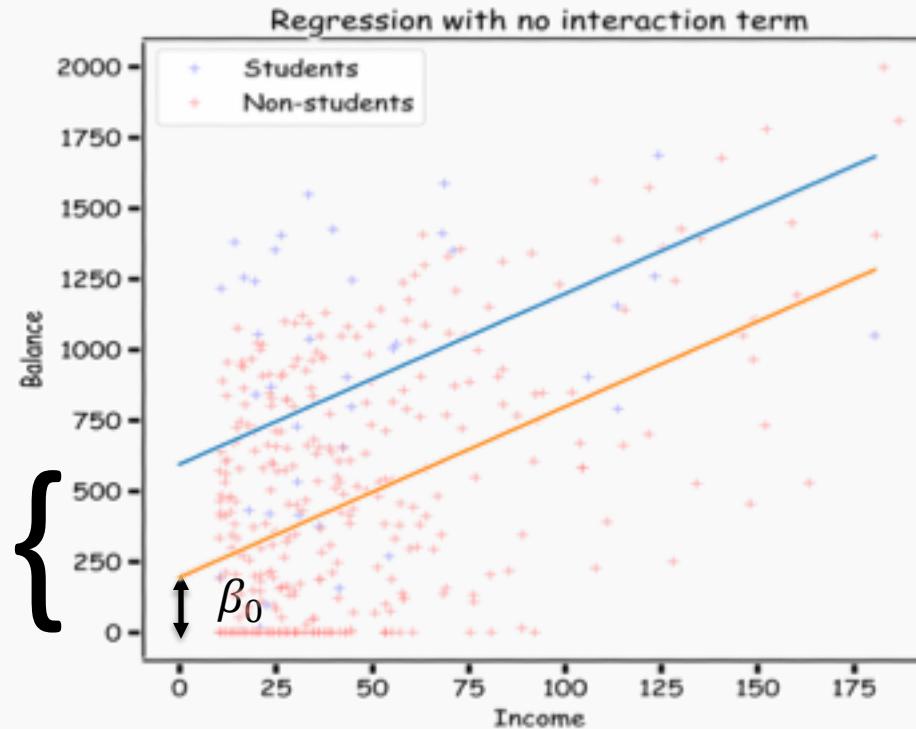
Not linear
term

What does it mean? First consider the case without the interaction term

$$\text{balance} = \beta_0 + \beta_1 \times \text{income} + \beta_2 \times \text{student}$$

$$\text{student} = \begin{cases} 0 & \text{balance} = \beta_0 + \beta_1 \times \text{income} \\ 1 & \text{balance} = \beta_0 + \beta_1 \times \text{income} + \beta_2 \rightarrow \text{balance} = (\beta_0 + \beta_2) + \underbrace{\beta_1 \times \text{income}}_{\text{slope}} \end{cases}$$

$$\beta_0 + \beta_2$$

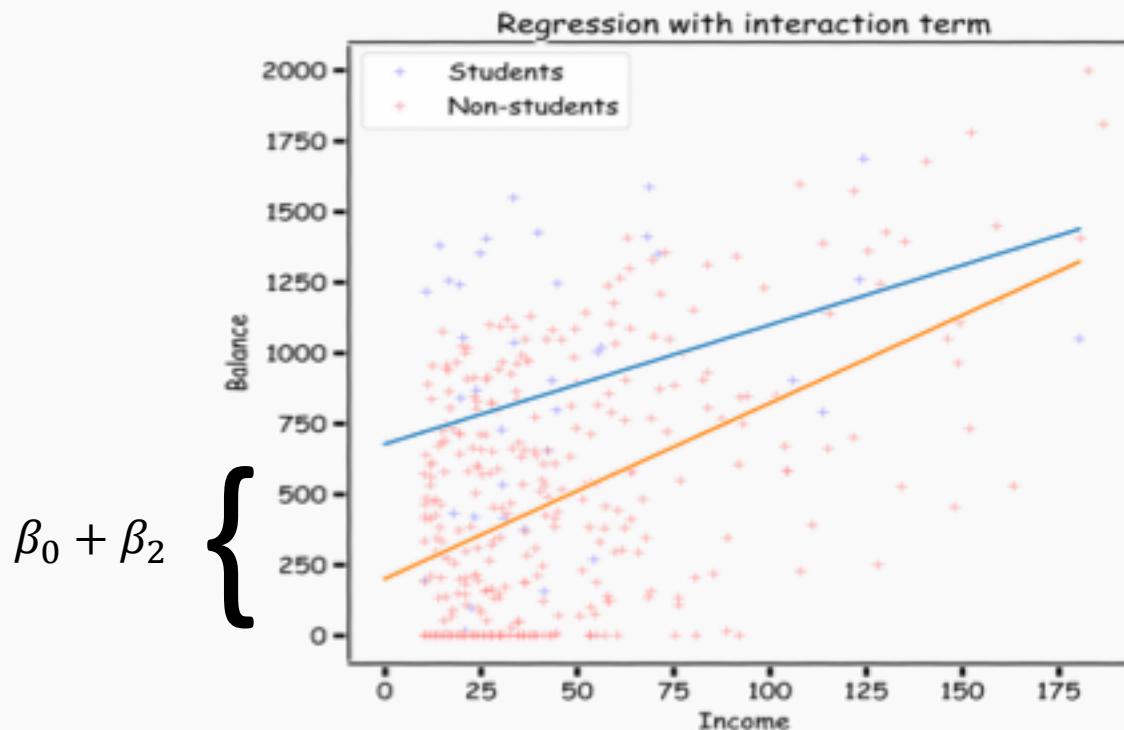


What does it mean? Next we consider the case with the interaction term

$$balance = \beta_0 + \beta_1 \times income + \beta_2 \times student + \beta_3 \times income \times student$$

$$student = \begin{cases} 0 & balance = \beta_0 + \beta_1 \times income \\ 1 & balance = \beta_0 + \beta_1 \times income + \beta_2 + \beta_3 \times income \end{cases}$$

$\rightarrow balance = (\underbrace{\beta_0 + \beta_2}_{\text{intercept}}) + (\underbrace{\beta_1 + \beta_3}_{\text{slope}}) \times income$





Digestion Time

Too many predictors, collinearity and too many
interaction terms leads to

Too many predictors, collinearity and too many
interaction terms leads to
OVERFITTING!

Lecture Outline

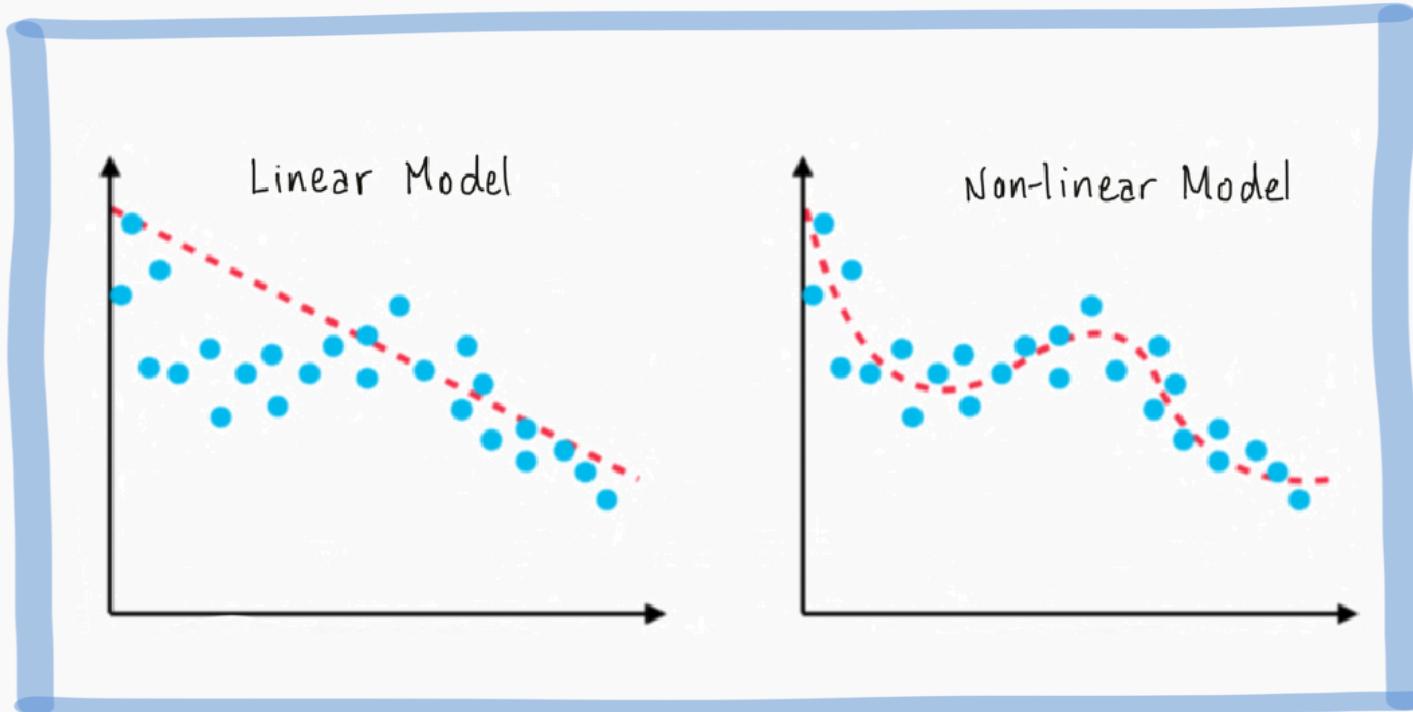
Interaction Effects in Regression Models

Polynomial Regression: Extending Linear Models

Model Selection Techniques: Focus on Cross-Validation

Fitting non-linear data

Multi-linear models can fit large datasets with many predictors. But the relationship between predictor and target isn't always linear.



We want a model:
 $y = f_{\beta}(x)$

Where f is a **non-linear** function and β is a vector of the **parameters** of f .

Polynomial Regression

The **simplest** non-linear model we can consider, for a response Y and a predictor x , is a polynomial model of degree M ,

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_M x^M$$

Just as in the case of multi-linear regression, **polynomial** regression is a **special case** of linear regression

HOW?

Polynomial Regression

The design matrix for a polynomial regression would be:

To the power of M

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} 1 & x_1^1 & \dots & x_1^M \\ 1 & x_2^1 & \dots & x_2^M \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n^1 & \dots & x_n^M \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_M \end{pmatrix}.$$

Polynomial Regression

The design matrix for a polynomial regression would be:

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} 1 & x_1^1 & \dots & x_1^M \\ 1 & x_2^1 & \dots & x_2^M \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^M \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_M \end{pmatrix}.$$

This looks a lot like [multi-linear regression](#) where the predictors are powers of x!

Multi-Regression

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_y \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} 1 & x_{1,1} & \dots & x_{1,J} \\ 1 & x_{2,1} & \dots & x_{2,J} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & \dots & x_{n,J} \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_J \end{pmatrix},$$

Model Training

Given a dataset $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, we want to fit a linear model:

$$y = \beta_0 + \beta_1 x + \epsilon$$

1. We transform the data by adding a column of 1's to the feature vector x :

$$\tilde{x} = [1, \hat{x}_1, \hat{x}_2, \dots, \hat{x}_n]^T$$

where $\hat{x}_k = x^k$

2. We find the parameter by minimizing the MSE using vector calculus yields, as in multi-linear regression

$$\hat{\beta} = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T y$$

`sklearn.linear_model.LinearRegression.fit()`

We can also perform multi-polynomial regression in the same way.

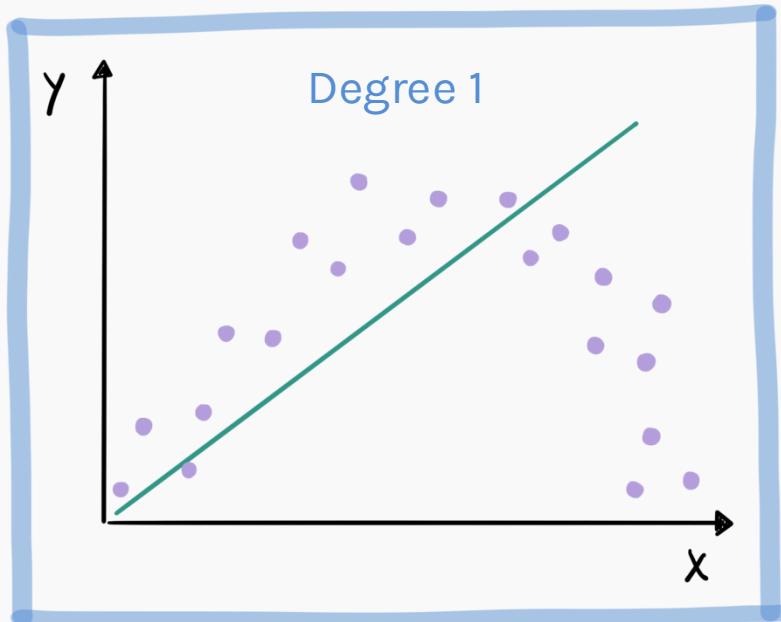
BUT be careful:

- A. Sklearn will include the interaction terms
- B. The new design matrix will include the 1 column so no need to fit for intercept

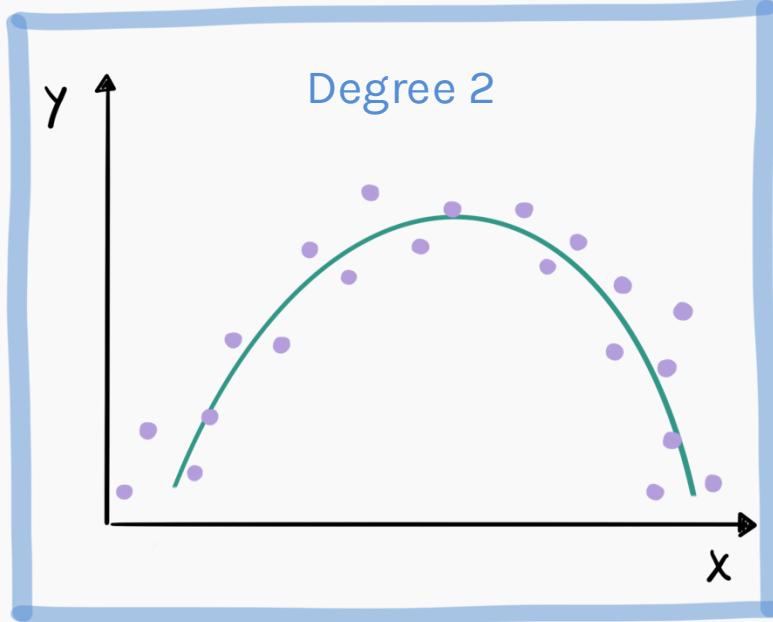
g:
oly
(?)

Polynomial Regression (cont.)

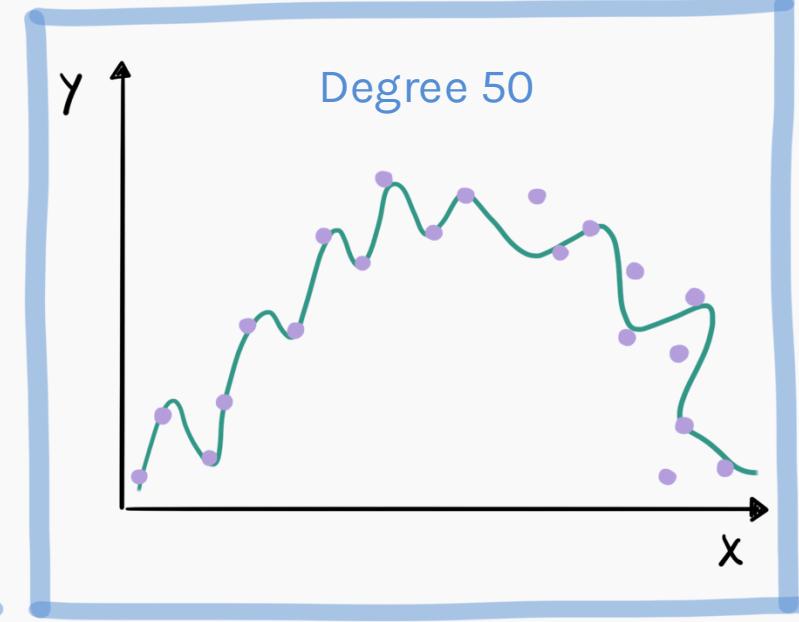
Fitting a polynomial model requires choosing a degree.



Underfitting: when the degree is too low, the model cannot fit the trend.



We want a model that fits the trend and ignores the noise.



Overfitting: when the degree is too high, the model fits all the noisy data points.

Feature Scaling

Do we need to scale out features for polynomial regression?

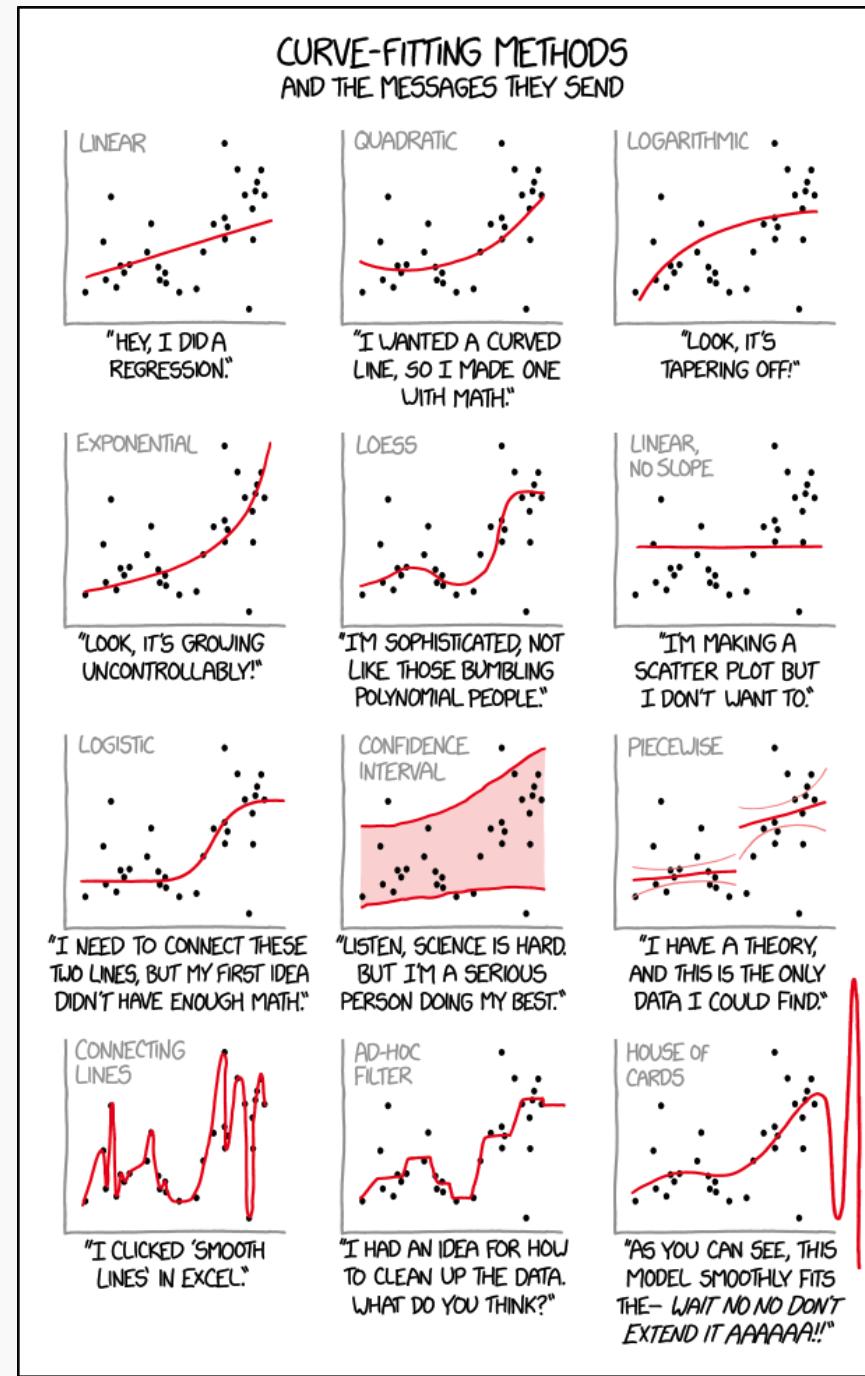
Linear regression, $Y = X\beta$, is [invariant under scaling](#). If X is multiplied by some number λ , then β will be scaled by $\frac{1}{\lambda}$ and MSE will be identical.

However, if the range of X is small or large, then we run into troubles. Consider a polynomial degree of 20 and the maximum or minimum value of any predictor is large or small. Those numbers to the 20th power will be [problematic](#).

It is always a good idea to [scale](#) X when considering [polynomial regression](#):

$$X^{norm} = \frac{X - \bar{X}}{\sigma_X}$$

Note: sklearn's StandardScaler() can do this.



Too many predictors, collinearity, too many interaction terms and high degree of polynomial leads to leads to

Too many predictors, collinearity, too many interaction terms and high degree of polynomial leads to leads to
OVERFITTING!

Too many predictors, collinearity, too many interaction terms and high degree of polynomial and model selection leads to

Too many predictors, collinearity, too many interaction terms and high degree of polynomial and model selection leads to

THESE ARE OVERFITTED STUDENTS!

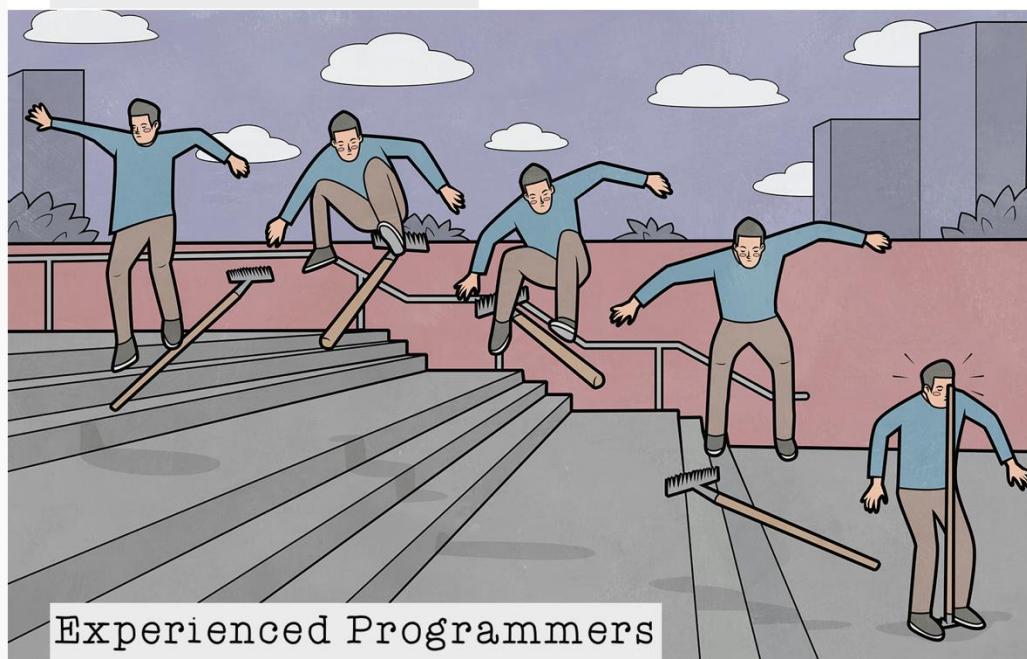
Multi-Linear Regression

CS109A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai and Chris Gumb





New Programmers



Experienced Programmers

Lecture Outline

Simple Linear Regression

Multi-linear Regression

Interpreting Model Parameters

Scaling

Collinearity

Qualitative Predictors



If you have to guess someone's height, would you rather be told

Options:

- A. Their weight, only
- B. Their weight and biological sex
- C. Their weight, biological sex, and income
- D. Their weight, biological sex, income, and favorite number

Multi-Linear Regression

Of course, you'd always **want as much data** about a person as possible. Even though height and favorite number may **not** be strongly related, at worst you could just **ignore** the information on favorite number.

We want our models to be able to take in lots of data as they make their predictions.

This approach brings up a few questions.

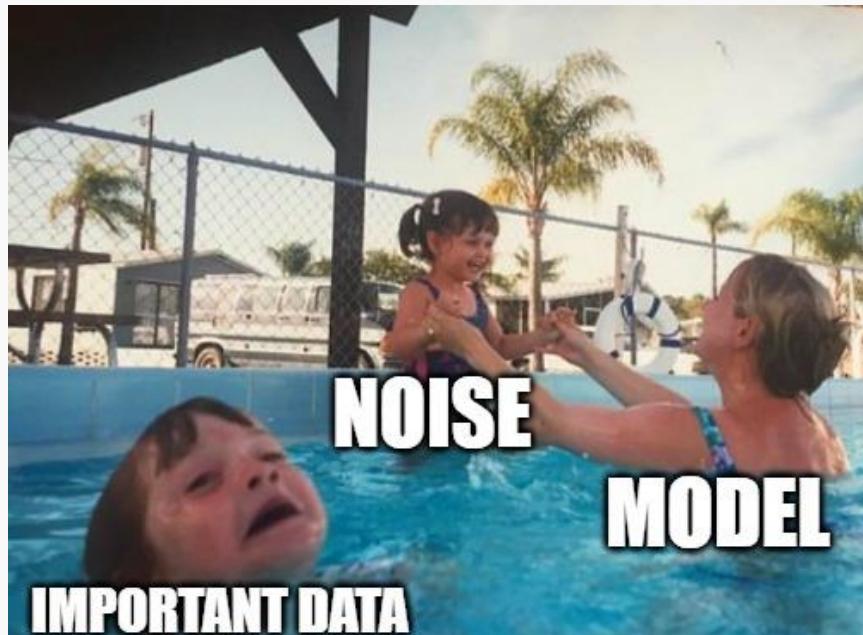


The model

Multi-Linear Regression

Data Noise

- Can too much irrelevant data introduce noise and make pattern detection difficult?



Ethical Considerations

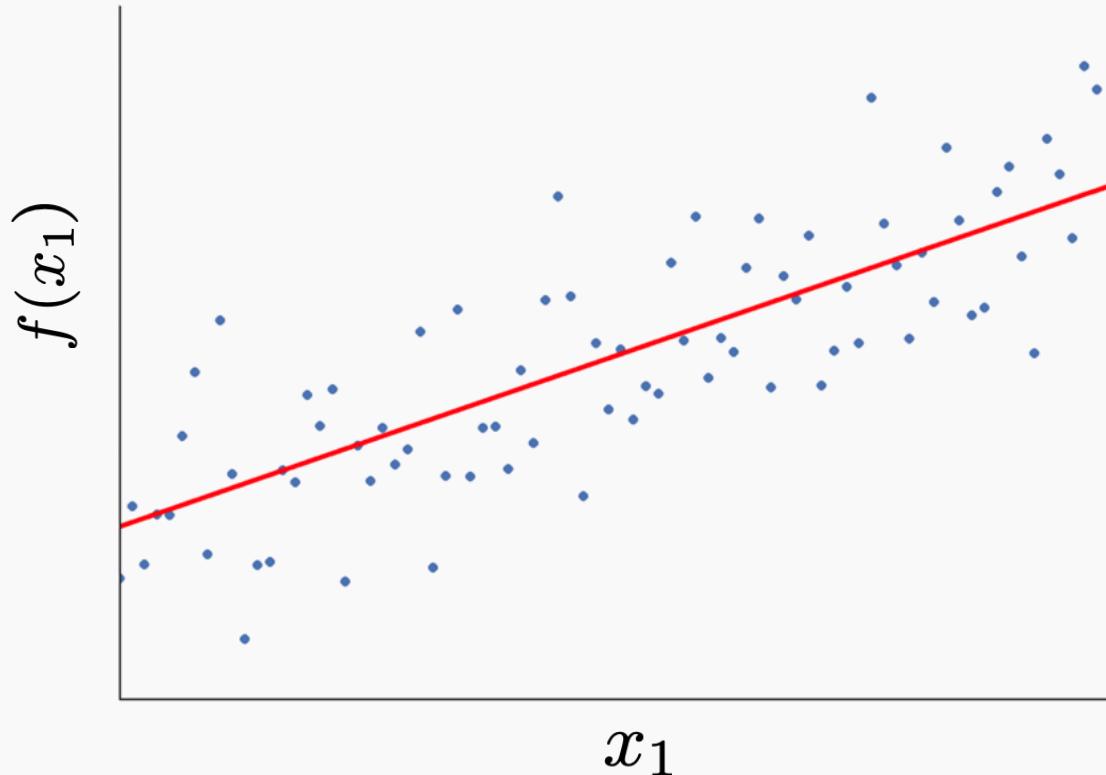
- Are there privacy concerns related to collecting more data than needed?



Simple Linear Regression

In simple linear regression, we assume a simple basic form for f :

$$f(x) = \beta_0 + \beta_1 x$$

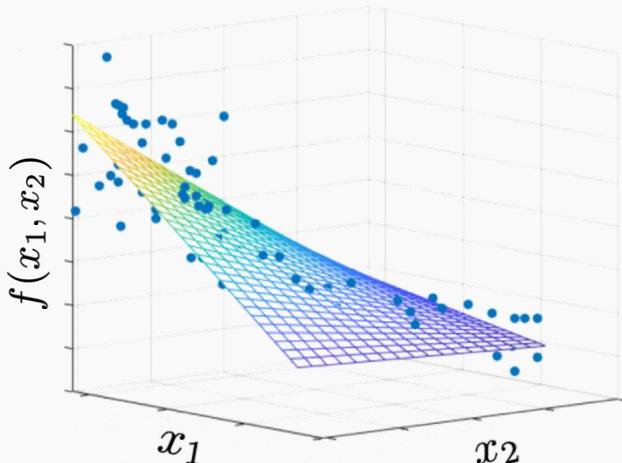


Linear Regression in n-D

In **practice**, it is unlikely that any response variable y depends solely on one predictor \mathbf{x} . Rather, we expect that y is a function of **multiple** predictors x_1, x_2, \dots, x_p .

Using the notation we introduced last part,

$$\mathbf{y} = y_1, \dots, y_n, \quad \mathbf{X} = \mathbf{x}_1, \dots, \mathbf{x}_p \quad \text{and} \quad \mathbf{x}_j = x_{1j}, \dots, x_{nj}$$



In **multiple** linear regression, we assume a **similar form** for f as in simple linear regression. We can assume a simple form for f a **multilinear** form:

$$f(\mathbf{x}_1, \dots, \mathbf{x}_p) = \beta_0 + \beta_1 \mathbf{x}_1 + \dots + \beta_p \mathbf{x}_p$$

Response vs. Predictor Variables

The Design Matrix

		x_1
n observations		
		TV
		230.1
		44.5
		17.2
		151.5
		180.8

y:
The response variable

		y
n observations		
		sales
		22.1
		10.4
		9.3
		18.5
		12.9

Response vs. Predictor Variables

The Design Matrix

n observations

TV	radio	newspaper
230.1	37.8	69.2
44.5	39.3	45.1
17.2	45.9	69.3
151.5	41.3	58.5
180.8	10.8	58.4

x_1 x_2 x_3

y:
The response variable

sales
22.1
10.4
9.3
18.5
12.9

Multi-Linear Regression, example

For our data

$$Sales = \beta_0 + \beta_1 \times TV + \beta_2 \times Radio + \beta_3 \times Newspaper$$

In linear algebra notation

$$Y = \begin{pmatrix} Sales_1 \\ \vdots \\ Sales_n \end{pmatrix}, X = \begin{pmatrix} 1 & TV_1 & Radio_1 & News_1 \\ \vdots & & \vdots & \vdots \\ 1 & TV_n & Radio_n & News_n \end{pmatrix}, \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_3 \end{pmatrix}$$

Multi-Linear Regression, example

For our data

$$Sales = \beta_0 + \beta_1 \times TV + \beta_2 \times Radio + \beta_3 \times Newspaper$$

In linear algebra notation

$$Y = \begin{pmatrix} Sales_1 \\ \vdots \\ Sales_n \end{pmatrix}, X = \begin{pmatrix} 1 & TV_1 & Radio_1 & News_1 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & TV_n & Radio_n & News_n \end{pmatrix}, \beta = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_3 \end{pmatrix}$$

Multi-Linear Regression, example

For our data

$$Sales = \beta_0 + \beta_1 \times TV + \beta_2 \times Radio + \beta_3 \times Newspaper$$

In linear algebra notation

$$Y = \begin{pmatrix} Sales_1 \\ \vdots \\ Sales_n \end{pmatrix}, X = \begin{pmatrix} 1 & TV_1 & Radio_1 & News_1 \\ \vdots & & \vdots & \vdots \\ 1 & TV_n & Radio_n & News_n \end{pmatrix}, \beta = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_3 \end{pmatrix}$$

$$Sales_1 = (1 \quad TV_1 \quad Radio_1 \quad News_1) \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_3 \end{pmatrix}$$

Multi-Linear Regression, example

$$Sales_1 = (1 \quad TV_1 \quad Radio_1 \quad News_1) \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_3 \end{pmatrix}$$



$$Sales = \beta_0 + \beta_1 \times TV + \beta_2 \times Radio + \beta_3 \times Newspaper$$



$$Y = X\beta$$

Multi-linear Regression - only consider 2 predictors

$$Y = X\beta$$

For simplicity we consider only two predictors

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \end{pmatrix}$$
$$X = \begin{pmatrix} 1 & x_{11} & x_{12} \\ \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} \end{pmatrix}$$
$$\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix}$$

RECAP: Transpose of a matrix

$$X = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ \dots & \dots \end{pmatrix}$$

In transpose, rows become columns and columns become rows.

$$X^T = \begin{pmatrix} x_{11} & x_{21} & \dots \\ x_{12} & x_{13} & \dots \end{pmatrix}$$

1	4
2	5
3	6

(n,2)

1	2	3
4	5	6

(2,n)

You can perform transpose over numpy objects by calling
np.transpose() or **ndarray.T**

RECAP: Inverse of a matrix

When we multiply a number by its reciprocal we get 1.

$$n * \frac{1}{n} = 1$$

When we multiply a matrix by its inverse, we get the Identity Matrix

$$A A^{-1} = I$$

```
In [16]: x = np.array([[1,2],[3,4]])
....:
....: #Inverse array x
....: invX = np.linalg.inv(x)
....: print(invX)
....:
....: #Verifying
....: print(np.dot(x, invX))
[[ -2.    1. ]
 [  1.5   -0.5]]
[[1.0000000e+00  1.11022302e-16]
 [0.0000000e+00  1.0000000e+00]]
```

`numpy.linalg.inv()` is used to calculate the inverse of a matrix
(if it exists!)

Multi-Linear Regression

The model takes a simple algebraic form: $Y = X\beta$

We will again choose the **MSE** as our loss function, which can be expressed in vector notation as

$$MSE(\beta) = \frac{1}{n} \|Y - X\beta\|^2$$

$$MSE(\beta) = \frac{1}{n} \sum_{i=1}^n (y_i - \beta_0 - x_{i1}\beta_1 - x_{i2}\beta_2)^2$$

Multi-Linear Regression

The model takes a simple algebraic form: $Y = X\beta$

This means
 $(Y - X\beta)^T(Y - X\beta)$

We will again choose the **MSE** as our loss function, which can be expressed in vector notation as

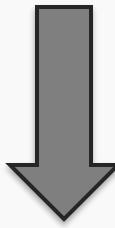
$$MSE(\beta) = \frac{1}{n} \|Y - X\beta\|_2^2$$

For simplicity again we consider only two predictors

$$MSE(\beta) = \frac{1}{n} \sum_{i=1}^n (y_i - \beta_0 - x_{i1}\beta_1 - x_{i2}\beta_2)^2$$

MSE minimization in 3D

$$MSE(\beta) = \frac{1}{n} \sum_{i=1}^n (y_i - \beta_0 - x_{i1}\beta_1 - x_{i2}\beta_2)^2$$



Dropping $\frac{1}{n}$ because it won't change the results.

$$MSE(\beta) = \frac{1}{n} \{(y_1 - \beta_0 - x_{11}\beta_1 - x_{12}\beta_2)^2 + (y_2 - \beta_0 - x_{21}\beta_1 - x_{22}\beta_2)^2 + \dots\}$$

MSE minimization in 3D

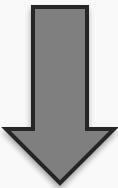
$$MSE(\beta) = \frac{1}{n} \sum_{i=1}^n (y_i - \beta_0 - x_{i1}\beta_1 - x_{i2}\beta_2)^2$$



$$\begin{aligned} MSE(\beta) = & (y_1 - \beta_0 - x_{11}\beta_1 - x_{12}\beta_2)^2 + \\ & (y_2 - \beta_0 - x_{21}\beta_1 - x_{22}\beta_2)^2 + \dots \end{aligned}$$

MSE minimization in 3D

$$MSE(\beta) = (y_1 - \beta_0 - x_{11}\beta_1 - x_{12}\beta_2)^2 + \\ (y_2 - \beta_0 - x_{21}\beta_1 - x_{22}\beta_2)^2 + \dots$$



$$\frac{\partial L}{\partial \beta_0} = -2(y_1 - \beta_0 - x_{11}\beta_1 - x_{12}\beta_2) \dots$$

$$\frac{\partial L}{\partial \beta_1} = -2x_{11}(y_1 - \beta_0 - x_{11}\beta_1 - x_{12}\beta_2) \dots$$

$$\frac{\partial L}{\partial \beta_2} = -2x_{12}(y_1 - \beta_0 - x_{11}\beta_1 - x_{12}\beta_2) \dots$$

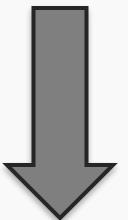
MSE minimization in 3D

$$\begin{pmatrix} \frac{\partial L}{\partial \beta_0} \\ \frac{\partial L}{\partial \beta_1} \\ \frac{\partial L}{\partial \beta_2} \end{pmatrix} = \begin{pmatrix} -2(y_1 - \beta_0 - x_{11}\beta_1 - x_{12}\beta_2) \dots \\ -2x_{11}(y_1 - \beta_0 - x_{11}\beta_1 - x_{12}\beta_2) \dots \\ -2x_{12}(y_1 - \beta_0 - x_{11}\beta_1 - x_{12}\beta_2) \dots \end{pmatrix}$$

$$\begin{pmatrix} \frac{\partial L}{\partial \beta_0} \\ \frac{\partial L}{\partial \beta_1} \\ \frac{\partial L}{\partial \beta_2} \end{pmatrix} = -2 \begin{pmatrix} 1 & 1 & \dots \\ x_{11} & x_{21} & \dots \\ x_{12} & x_{22} & \dots \end{pmatrix} \begin{pmatrix} (y_1 - \beta_0 - x_{11}\beta_1 - x_{12}\beta_2) \\ (y_2 - \beta_0 - x_{21}\beta_1 - x_{22}\beta_2) \\ \dots \end{pmatrix}$$

MSE minimization in 3D

$$\begin{pmatrix} \frac{\partial L}{\partial \beta_0} \\ \frac{\partial L}{\partial \beta_1} \\ \frac{\partial L}{\partial \beta_2} \end{pmatrix} = -2 \begin{pmatrix} 1 & 1 & \dots \\ x_{11} & x_{21} & \dots \\ x_{12} & x_{22} & \dots \end{pmatrix} \begin{pmatrix} (y_1 - \beta_0 - x_{11}\beta_1 - x_{12}\beta_2) \\ (y_2 - \beta_0 - x_{21}\beta_1 - x_{22}\beta_2) \\ \dots \end{pmatrix}$$



$$\begin{pmatrix} \frac{\partial L}{\partial \beta_0} \\ \frac{\partial L}{\partial \beta_1} \\ \frac{\partial L}{\partial \beta_2} \end{pmatrix} = -2 \begin{pmatrix} 1 & 1 & \dots \\ x_{11} & x_{21} & \dots \\ x_{12} & x_{22} & \dots \end{pmatrix} \left[\begin{pmatrix} y_1 \\ y_2 \\ \dots \end{pmatrix} - \begin{pmatrix} 1 & x_{11} & x_{12} \\ \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix} \right]$$

MSE minimization in 3D

$$\begin{pmatrix} \frac{\partial L}{\partial \beta_0} \\ \frac{\partial L}{\partial \beta_1} \\ \frac{\partial L}{\partial \beta_2} \end{pmatrix} = -2 \begin{pmatrix} 1 & 1 & \dots \\ x_{11} & x_{21} & \dots \\ x_{12} & x_{22} & \dots \end{pmatrix} \left[\begin{pmatrix} y_1 \\ y_2 \\ \dots \end{pmatrix} - \begin{pmatrix} 1 & x_{11} & x_{12} \\ \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix} \right]$$

$$\begin{pmatrix} \frac{\partial L}{\partial \boldsymbol{\beta}} \end{pmatrix} = -2X^T(Y - X\boldsymbol{\beta})$$

For optimization, we set the values of the partial derivative to zero, i.e., $\begin{pmatrix} \frac{\partial L}{\partial \boldsymbol{\beta}} \end{pmatrix} = 0$

MSE minimization in 3D

$$\begin{pmatrix} \frac{\partial L}{\partial \beta_0} \\ \frac{\partial L}{\partial \beta_1} \\ \frac{\partial L}{\partial \beta_2} \end{pmatrix} = -2 \begin{pmatrix} 1 & 1 & \dots \\ x_{11} & x_{21} & \dots \\ x_{12} & x_{22} & \dots \end{pmatrix} \left[\begin{pmatrix} y_1 \\ y_2 \\ \dots \end{pmatrix} - \begin{pmatrix} 1 & x_{11} & x_{12} \\ \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix} \right]$$

$$\begin{pmatrix} \frac{\partial L}{\partial \boldsymbol{\beta}} \end{pmatrix} = 0 \quad \Rightarrow \quad -2X^T(Y - X\boldsymbol{\beta}) = 0$$
$$\qquad \qquad \qquad \Rightarrow \qquad \qquad \qquad X^T(Y - X\boldsymbol{\beta}) = 0$$

MSE minimization in 3D

$$X^T(Y - X\beta) = 0$$

Which gives us,

$$X^T Y - X^T X \beta = 0$$

Multiplying on both sides with $(X^T X)^{-1}$

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

$$\Rightarrow \beta = (X^T X)^{-1} X^T Y$$



RECAP: Multi-Linear Regression

The model takes a simple algebraic form: $\mathbf{Y} = \mathbf{X}\beta + \epsilon$

We will again choose the **MSE** as our loss function, which can be expressed in vector notation as

$$\text{MSE}(\beta) = \frac{1}{n} \|\mathbf{Y} - \mathbf{X}\beta\|^2$$

Minimizing the MSE using vector calculus yields,

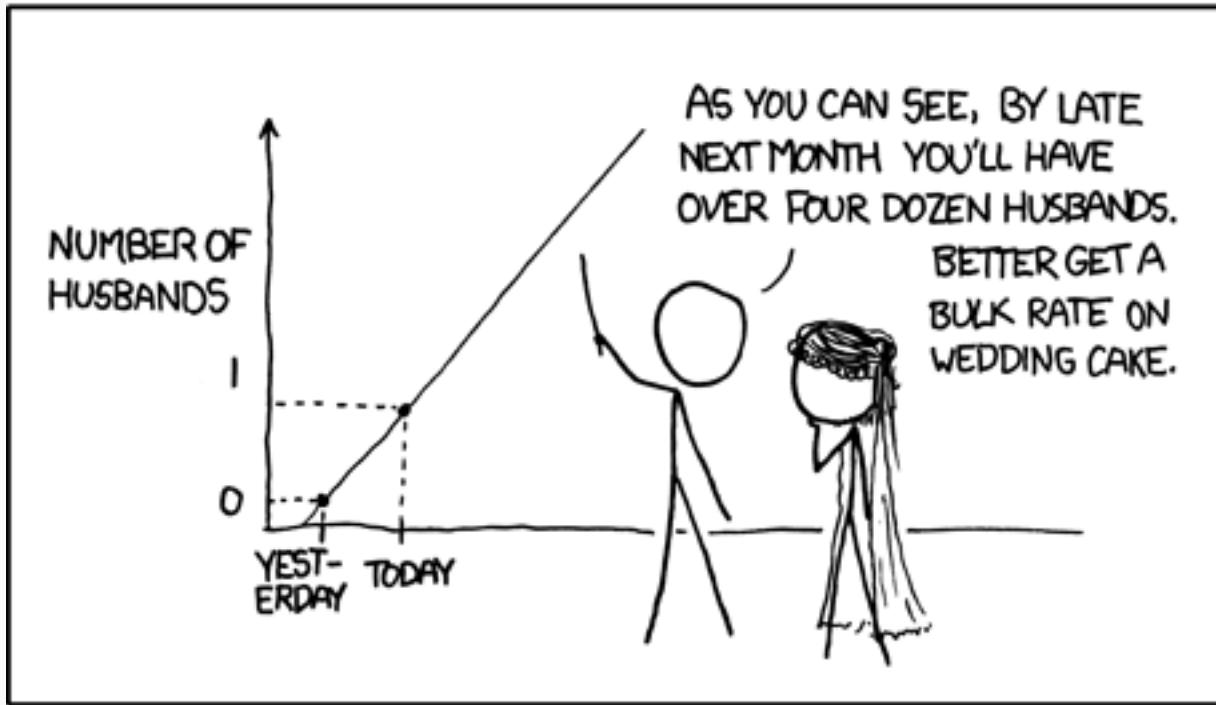
$$\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} = \underset{\beta}{\operatorname{argmin}} \text{MSE}(\beta).$$

Multi-Linear Regression

$$\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} = \underset{\beta}{\operatorname{argmin}} \text{MSE}(\beta).$$

```
>>> import numpy as np
>>> X = ...
>>> y = ...
>>> X_sq = X.T @ X
>>> X_inv = np.linalg.inv(X_sq)
>>> beta_hat = X_inv @ (X.T @ y)
```

MY HOBBY: EXTRAPOLATING



Digestion Time

Interpreting Model Parameters

Lecture Outline

Simple Linear Regression

Multi Linear Regression

Interpreting Model Parameters

Scaling

Collinearity

Qualitative Predictors



In a simple linear regression model, you have the equation $Y=5+3X$. What does the coefficient 3 represent?

Options

- A. The predicted value of Y when X=0
- B. The change in Y for a one-unit change in X
- C. The amount by which Y varies randomly around the line
- D. None of the above

Interpreting Model Parameters in Simple Linear Regression

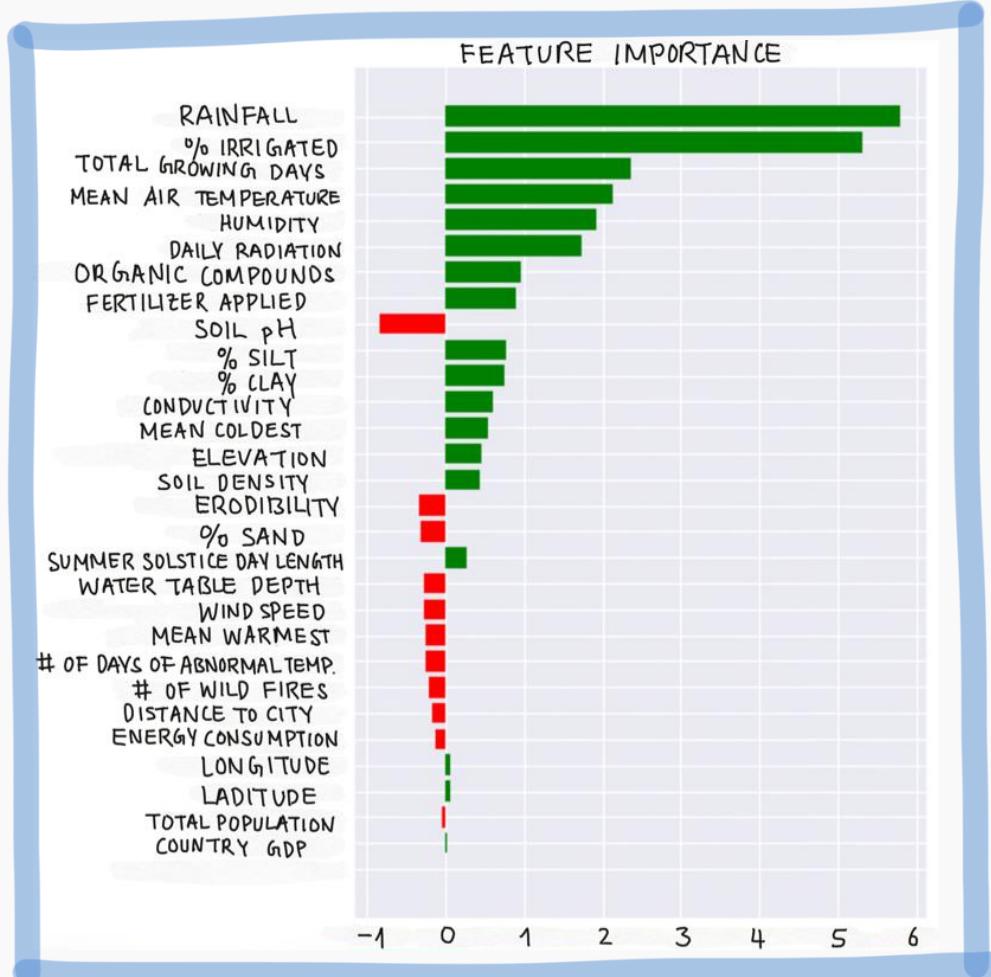
In the case of simple linear models, interpreting the model parameters is straightforward.

Interpretation

- β_0 : Predicted value of Y when $X=0$
- β_1 : Change in Y for a one-unit change in X

Interpreting multi-linear regression

In the case of simple linear models, interpreting the model parameters is straightforward.



When we have a large number of predictors: X_1, \dots, X_J , there will be a large number of model parameters, $\beta_1, \beta_2, \dots, \beta_J$.

Looking at the values of β 's is impractical, so we visualize these values in a feature importance graph.

The feature importance graph shows which predictors has the most impact on the model's prediction.



In a multiple linear regression model, how does scaling the predictor variables affect the interpretation of feature importance based on the β coefficients?

Options

- A. Scaling the predictors makes it easier to directly compare the importance of each feature based on their β coefficients.
- B. Scaling the predictors makes all the features equally important.
- C. Scaling the predictors increases the magnitude of β coefficients for less important features.
- D. Scaling the predictors eliminates the need for β coefficients for feature importance.

Scaling

Understanding Scaling: Standardization & Normalization

Scaling transforms your data so that it fits within a specific range or distribution.

Standardization (Z-Score)

Transforms data to have mean = 0 and standard deviation = 1

$$\frac{X - \text{mean}}{\text{std}}$$

Normalization (Min-Max Scaling)

Rescales data to range between 0 and 1

$$\frac{X - \min}{\max - \min}$$

Why Scale?

- Makes algorithms sensitive to feature scales perform better
- Facilitates easier interpretation and analysis



For More In-depth check my notes and examples on EdStem!

Collinearity

Lecture Outline

Simple Linear Regression

Multi-linear Regression

Interpreting Model Parameters

Scaling

Collinearity

Qualitative Predictors

Collinearity

We will discuss the assumptions of linear regression

Collinearity refers to a situation where two or more predictors in a regression model are highly correlated with each other.

While collinearity **doesn't violate the assumptions** of linear regression, it can make it difficult to determine the individual effect of each predictor on the response variable.

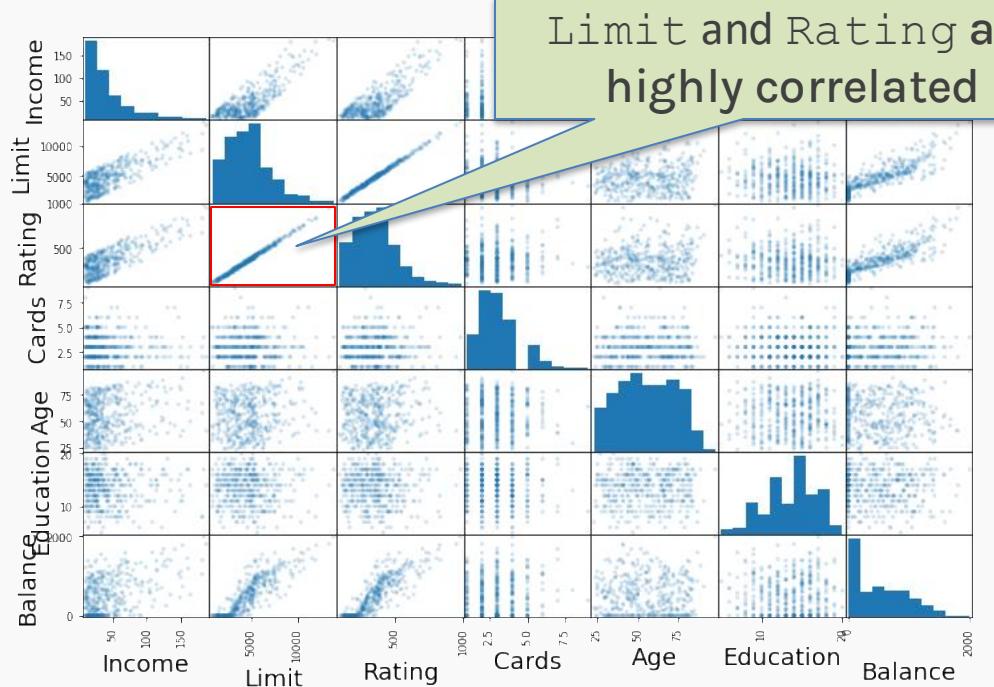
We will discuss confidence estimation of the parameters

Collinearity affects our **confidence in the estimated coefficients**, making it challenging to assess the importance of individual predictors.

Delve? ChatGPT took over my slides!

We will **delve deeper** into the implications of collinearity in the context of overfitting in our next lecture.

Collinearity



Cards do not seem to be correlated with anything yet, but its coefficient value changed significantly. WHY?

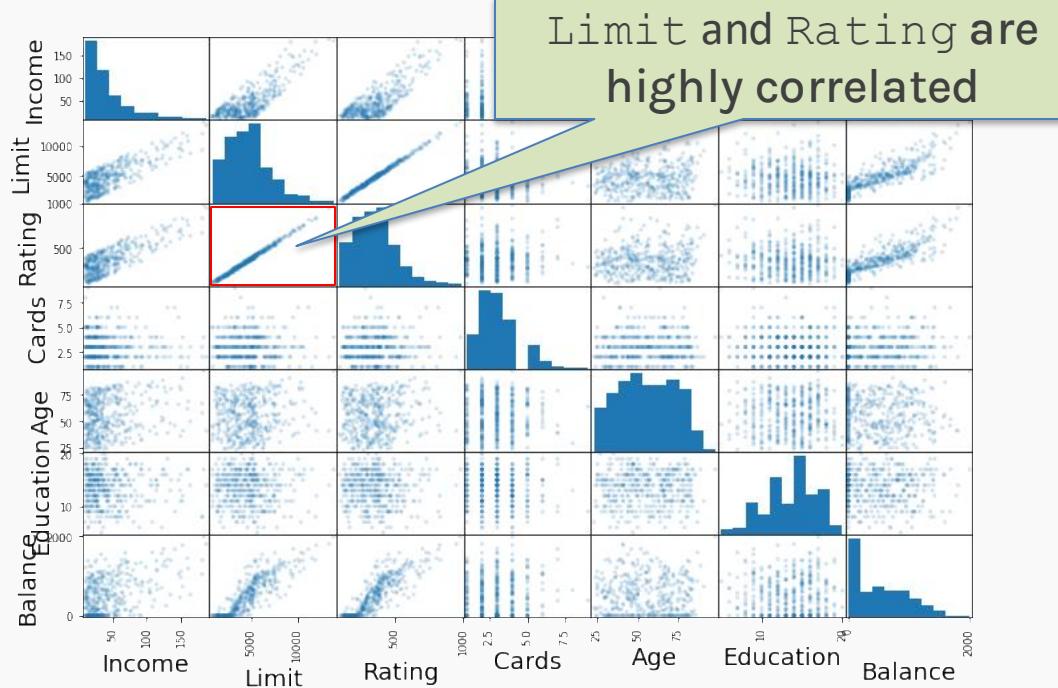
Columns	Coefficients
0 Income	-7.802001
1 Limit	0.193077
2 Rating	1.102269
3 Cards	17.923274
4 Age	-0.634677
5 Education	-1.115028
6 Gender	10.406651
7 Student	426.469192
8 Married	-7.019100

Column	Coefficients
0 Income	7.70915
1 Rating	0.976119
2 Cards	4.031215
3 Age	-0.669308
4 Education	-0.375954
5 Gender	10.368840
6 Student	417.417484
7 Married	-13.265344

Non-unique regression coefficients reduce model **interpretability** due to feature influence.

Positive coefficients for both limit and rating create **ambiguity** in attributing balance changes. Removing limit maintains model performance but alters coefficients.

Collinearity



Non-unique regression coefficients reduce model **interpretability** due to feature influence.

Re-run: It was a mistake

Columns	Coefficients
0 Income	-7.802001
1 Limit	0.193077
2 Rating	1.102269
3 Cards	17.923274
4 Age	-0.634677
5 Education	-1.115028
6 Gender	10.406651
7 Student	426.469192
8 Married	-7.019100

Columns	Coefficients
0 Income	7.70915
1 Rating	3.976119
2 Cards	14.031214
3 Age	-0.669308
4 Education	-0.375954
5 Gender	10.368840
6 Student	417.417484
7 Married	-13.265344

Positive coefficients for both limit and rating create **ambiguity** in attributing balance changes. Removing limit maintains model performance but alters coefficients.

Qualitative Predictors

Qualitative Predictors

So far, we have assumed that all variables are **quantitative**. But in practice, often some predictors are **qualitative**.

Example: The *credit data set* contains information about balance, age, cards, education, income, limit, and rating for a number of potential customers.

Income	Limit	Rating	Cards	Age	Education	Sex	Student	Married	Ethnicity	Balance
14.890	3606	283	2	34	11	Male	No	Yes	Caucasian	333
106.02	6645	483	3	82	15	Female	Yes	Yes	Asian	903
104.59	7075	514	4	71	11	Male	No	No	Asian	580
148.92	9504	681	3	36	11	Female	No	No	Asian	964
55.882	4897	357	2	68	16	Male	No	Yes	Caucasian	331

Qualitative Predictors

So far, we have assumed that all variables are **quantitative**. But in practice, often some predictors are **qualitative**.

Example: The *credit data set* contains information about balance, age, cards, education, income, limit, and rating for a number of potential customers.

Income	Limit	Rating	Cards	Age	Education	Sex	Student	Married	Ethnicity	Balance
14.890	3606	283	2	34	11	Male	No	Yes	Caucasian	333
106.02	6645	483	3	82	15	Female	Yes	Yes	Asian	903
104.59	7075	514	4	71	11	Male	No	No	Asian	580
148.92	9504	681	3	36	11	Female	No	No	Asian	964
55.882	4897	357	2	68	16	Male	No	Yes	Caucasian	331



You have a dataset with a column named 'Student' containing values 'Yes' and 'No'. How would you encode this column as a binary variable?

Options

- A. Replace 'No' with 0 and 'Yes' with 1
- B. Replace 'No' with 1 and 'Yes' with 2
- C. Replace 'No' with 1 and 'Yes' with 0
- D. Replace 'No' with 'N' and 'Yes' with 'Y'

Qualitative Predictors

If the predictor takes only two values, then we create an **indicator** or **dummy variable** that takes on two possible numerical values.

For example, for the sex column, we create a new variable:

$$x_i = \begin{cases} 1 & \text{if } i \text{ th person is female} \\ 0 & \text{if } i \text{ th person is male} \end{cases}$$

We then use this variable as a predictor in the regression equation.

$$y_i = \beta_0 + \beta_1 x_i = \begin{cases} \beta_0 + \beta_1 & \text{if } i \text{ th person is female} \\ \beta_0 & \text{if } i \text{ th person is male} \end{cases}$$



What is interpretation of β_0 and β_1 ?

Select all that apply.

$$x_i = \begin{cases} 1 & \text{if } i \text{ th person is female} \\ 0 & \text{if } i \text{ th person is male} \end{cases}$$

$$y_i = \beta_0 + \beta_1 x_i$$

Options

- A. β_0 represents the expected value of **balance** for **males**.
- B. β_0 represents the difference in **balance** between **males** and **females**.
- C. $\beta_0 + \beta_1$ represents the expected in **balance** for **females**.
- D. β_1 the average **difference** in **balance** between **females** and **males**.

More than two levels: One hot encoding

Why?

Often, the qualitative predictor takes more than two values (e.g. ethnicity in the credit data).

In this situation, a single dummy variable cannot represent all possible values.

We create **additional** dummy variable as:

$$x_{i,1} = \begin{cases} 1 & \text{if } i\text{ th person is Asian} \\ 0 & \text{if } i\text{ th person is not Asian} \end{cases}$$

$$x_{i,2} = \begin{cases} 1 & \text{if } i\text{ th person is Caucasian} \\ 0 & \text{if } i\text{ th person is not Caucasian} \end{cases}$$

More than two levels: One hot encoding

We then use these variables as predictors, the regression equation becomes:

$$y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} = \begin{cases} \beta_0 + \beta_1 & \text{if } i \text{ th person is Asian} \\ \beta_0 + \beta_2 & \text{if } i \text{ th person is Caucasian} \\ \beta_0 & \text{if } i \text{ th person is AfricanAmerican} \end{cases}$$

Question: What is the interpretation of $\beta_0, \beta_1, \beta_2$?

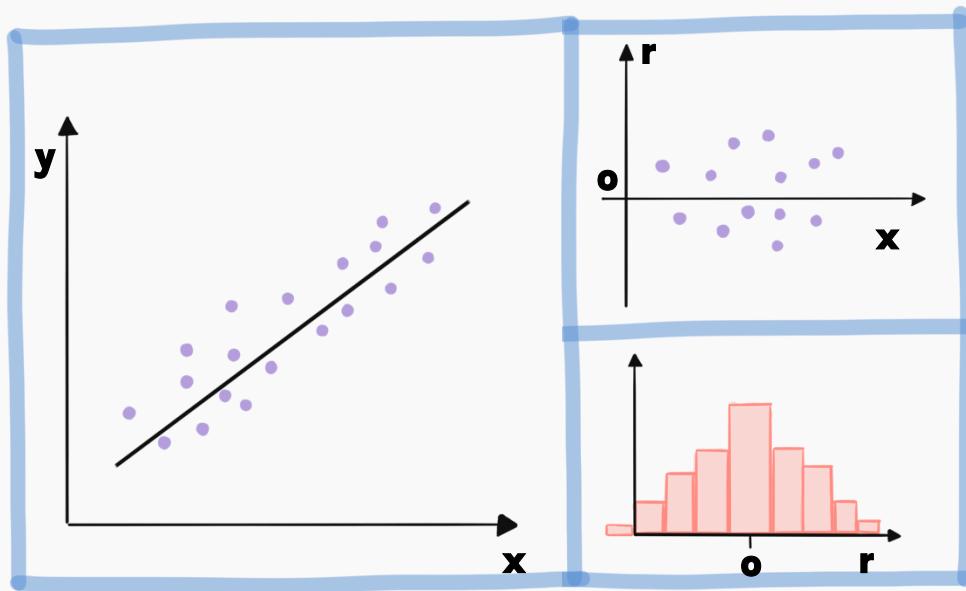
Beyond linearity

So far, we assumed:

- linear relationship between X and Y
- the residuals $r_i = y_i - \hat{y}_i$ were **uncorrelated** (taking the average of the square residuals to calculate the MSE implicitly assumed uncorrelated residuals)

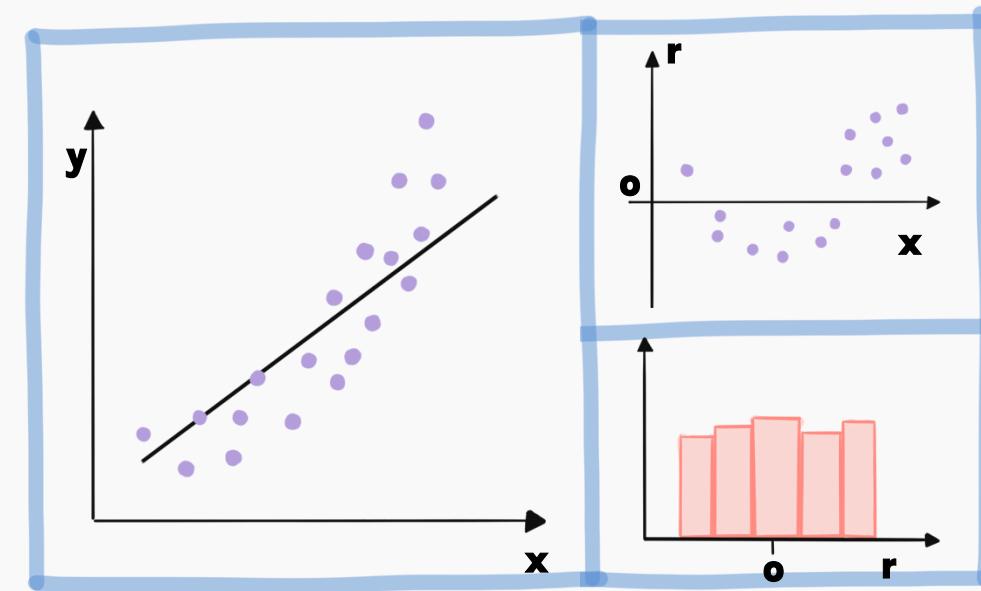
These assumptions need to be verified using the data. This is often done by **visually inspecting the residuals**.

Residual Analysis



Linear assumption is correct. There is no obvious relationship between residuals and x . Histogram of residuals is **symmetric** and **normally distributed**.

Note: For multi-regression, we plot the residuals vs predicted, \hat{y} , since there are too many x 's and that could wash out the relationship.



Linear assumption is incorrect. There is an obvious relationship between residuals and x . Histogram of residuals is **symmetric** but **not normally distributed**.



Introduction to Linear Regression

CS109A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai and Chris Gumb



Lecture Outline

Simple Linear Regression

Multi-linear Regression

Interpreting Model Parameters

Scaling

Collinearity

Qualitative Predictors

Lecture Outline

Simple Linear Regression

Multi-linear Regression

Interpreting Model Parameters

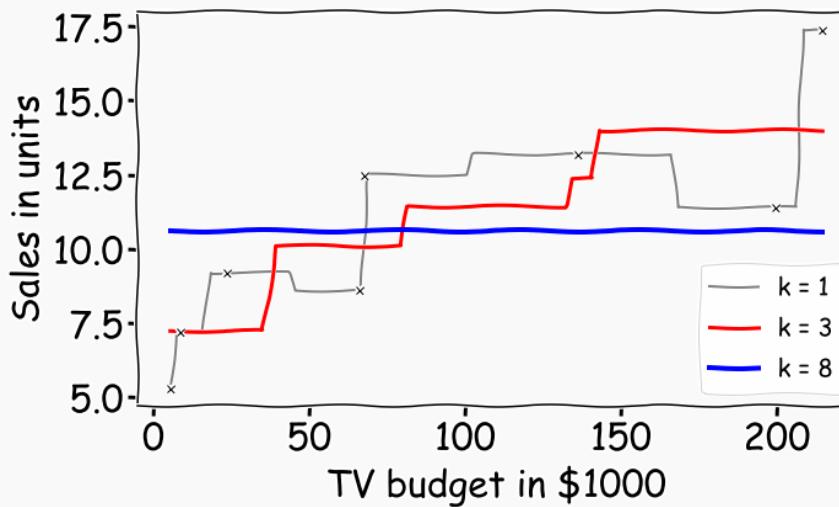
Scaling

Collinearity

Qualitative Predictors

Linear Models

kNN model



Note that when building our kNN model for prediction, which is non-parametric, we did not compute a closed-form solution for \hat{f} . So, what happens when we pose the question

How much more in sales can we expect if we double the TV advertising budget?

Linear Regression

Linear Models

We can build a model by first assuming a simple form of f :

$$f(x) = \beta_0 + \beta_1 X$$

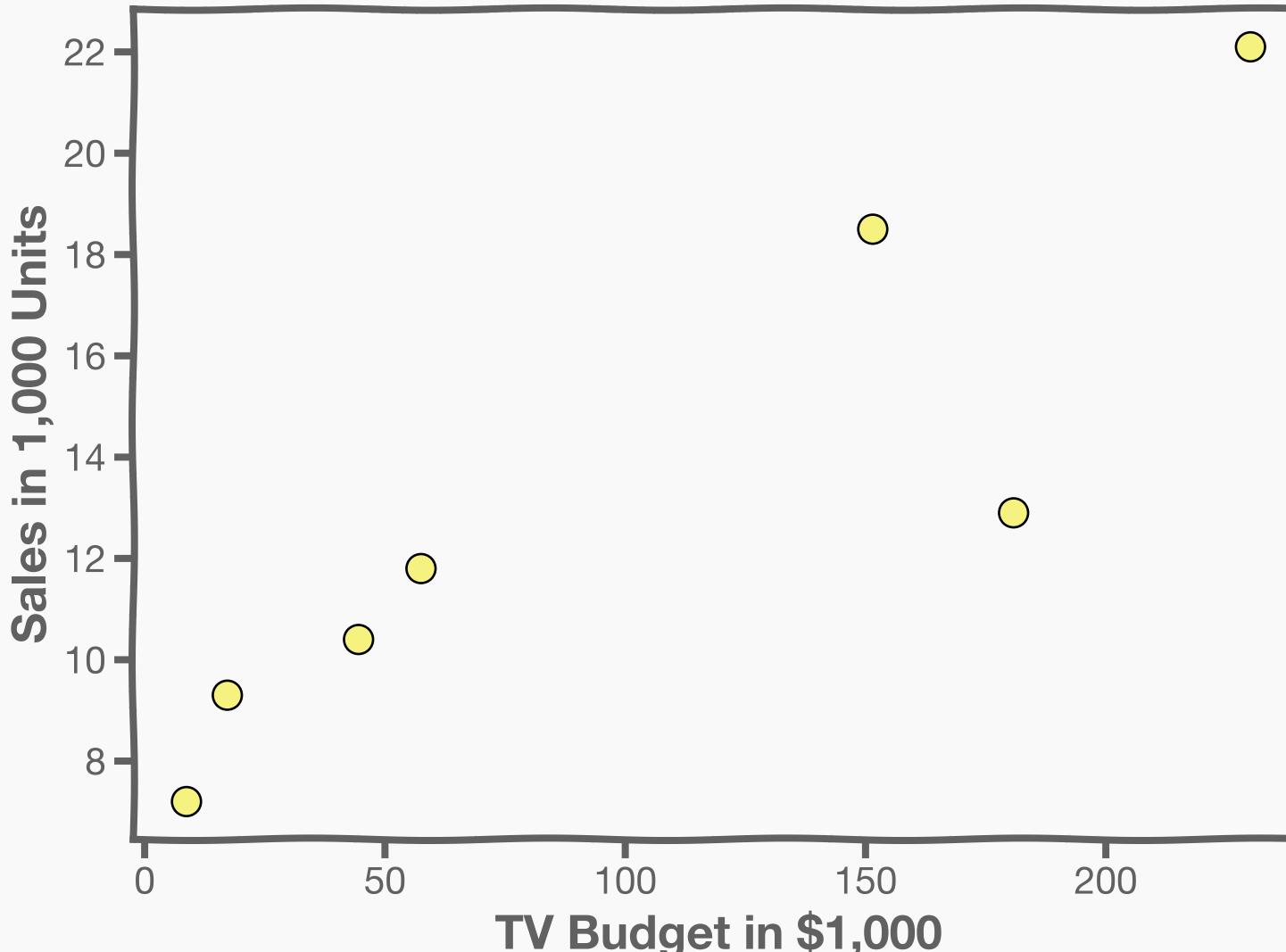
... then it follows that our estimate is:

$$\hat{Y} = \hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 X$$

where $\hat{\beta}_1$ and $\hat{\beta}_0$ are **estimates** of β_1 and β_0 respectively, that we compute using observations.

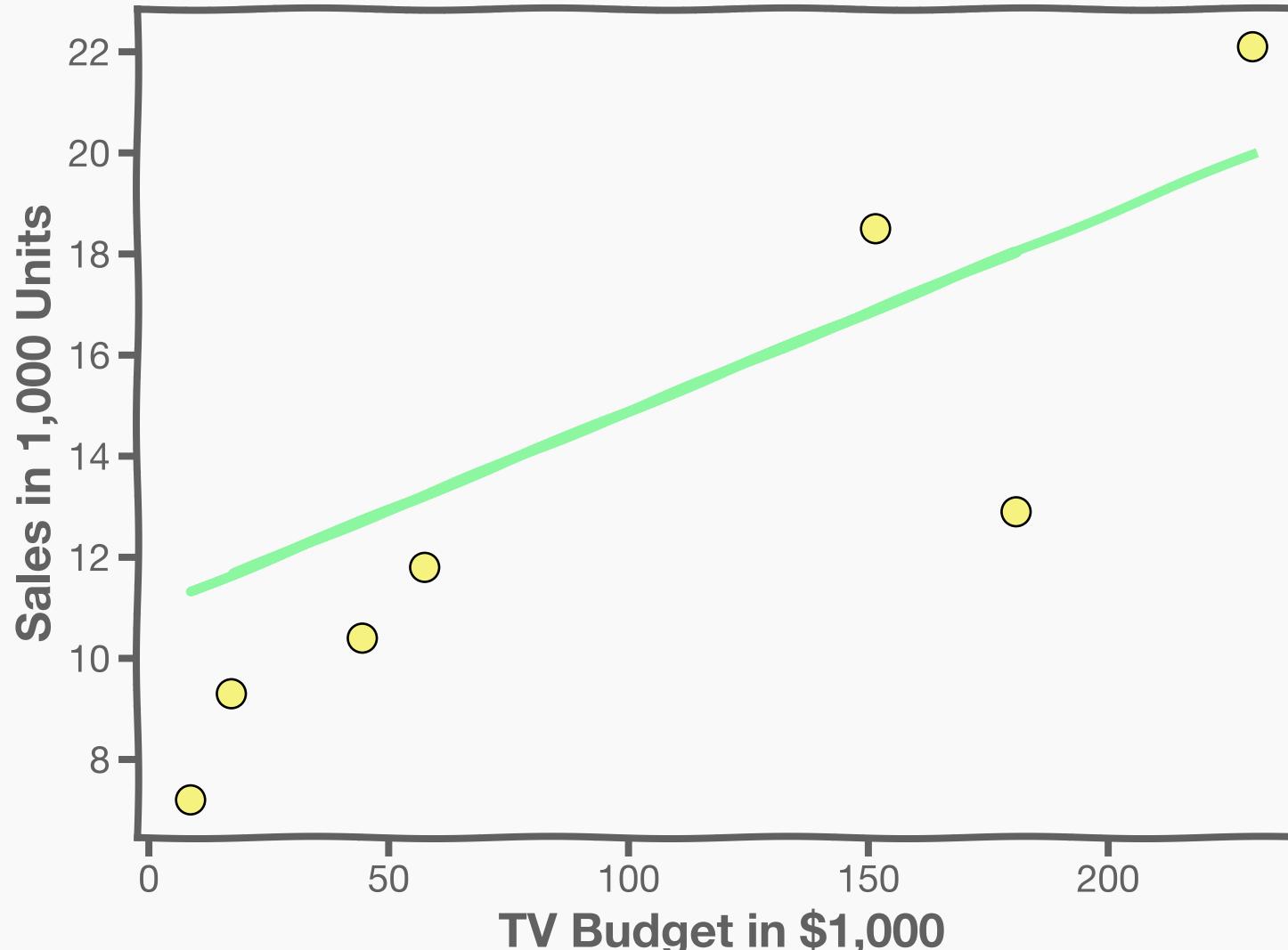
Estimate of the regression coefficients

For a given data set



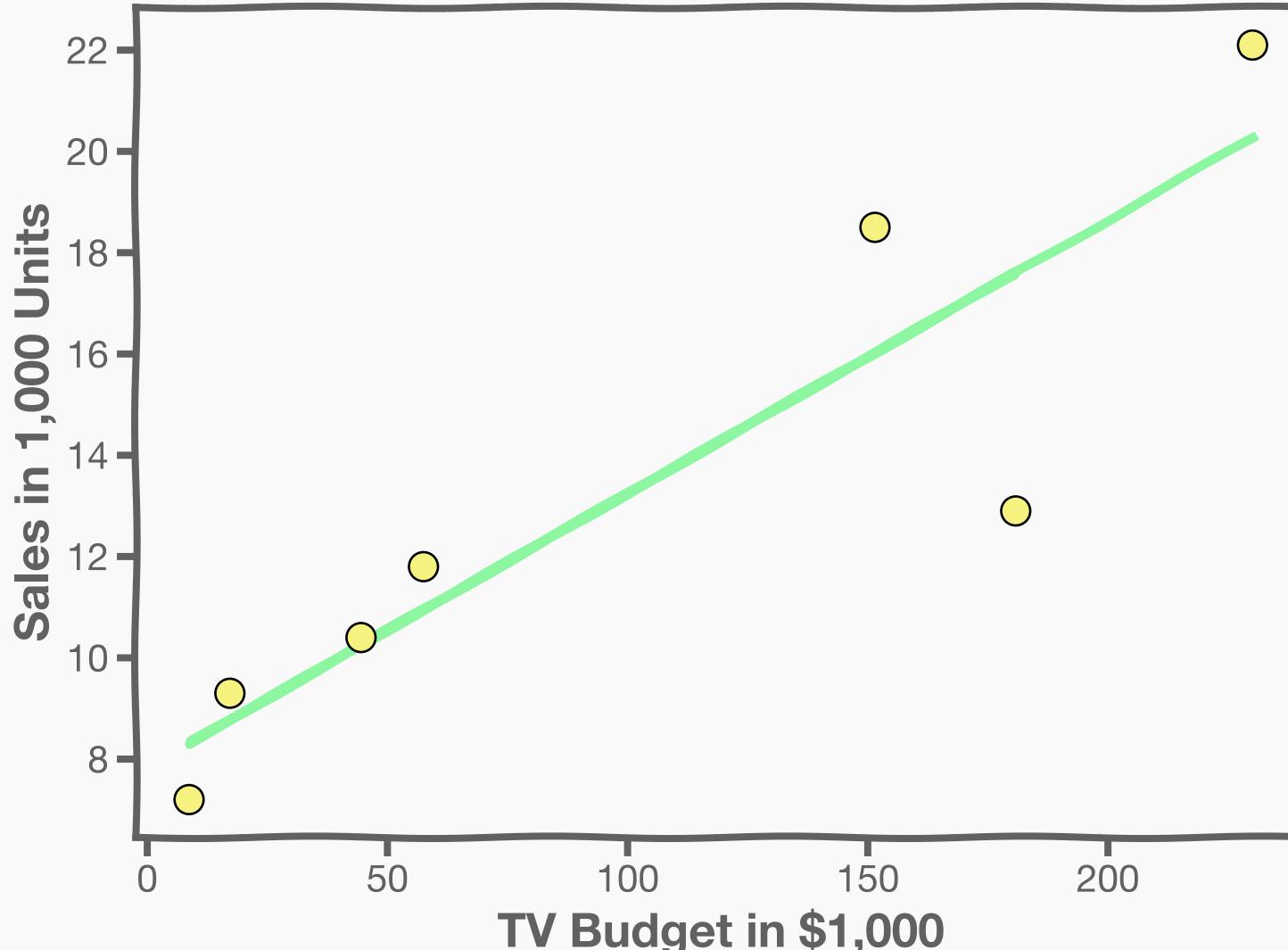
Estimate of the regression coefficients (cont)

Is this line good?



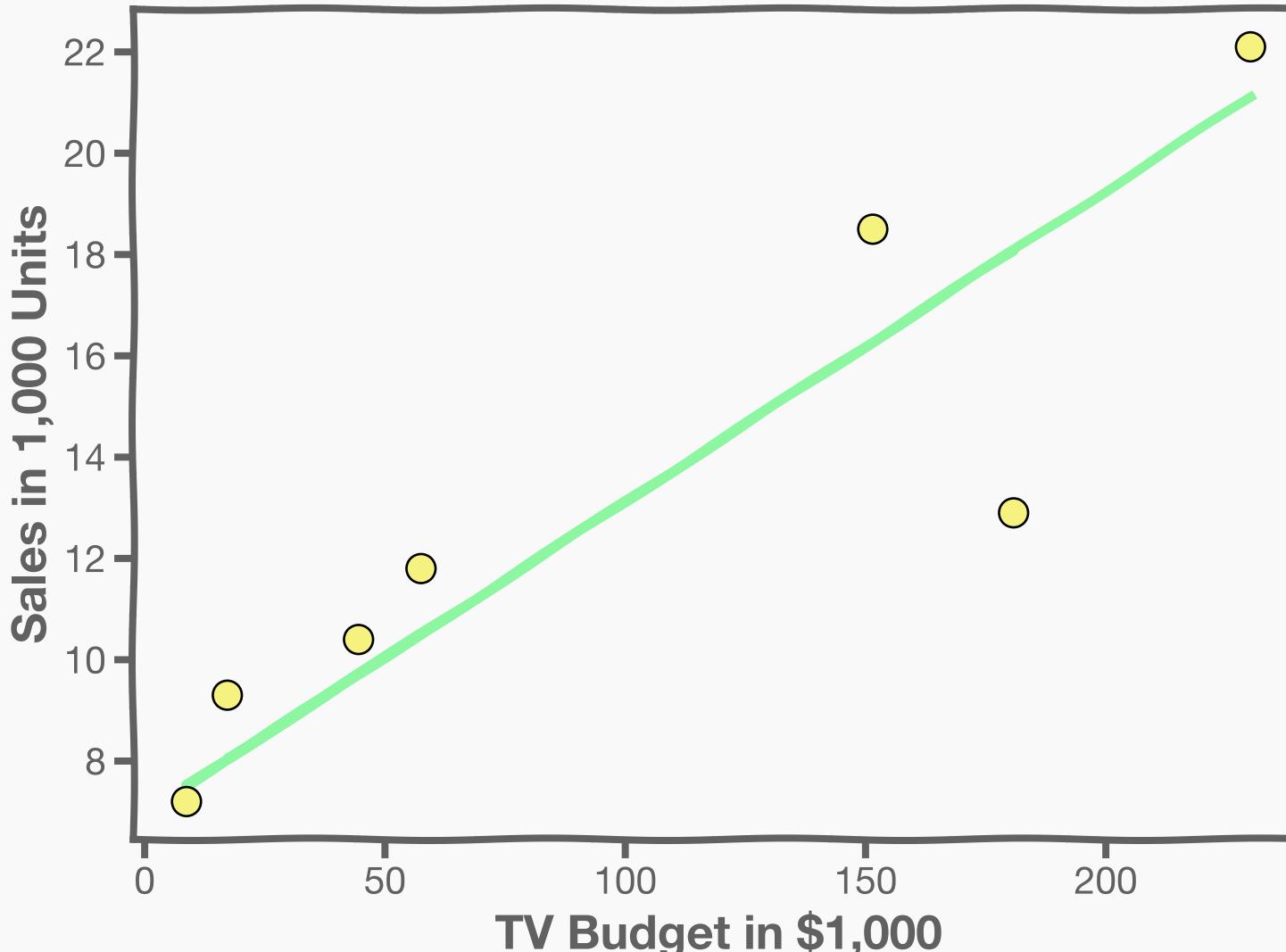
Estimate of the regression coefficients (cont)

Maybe this one?



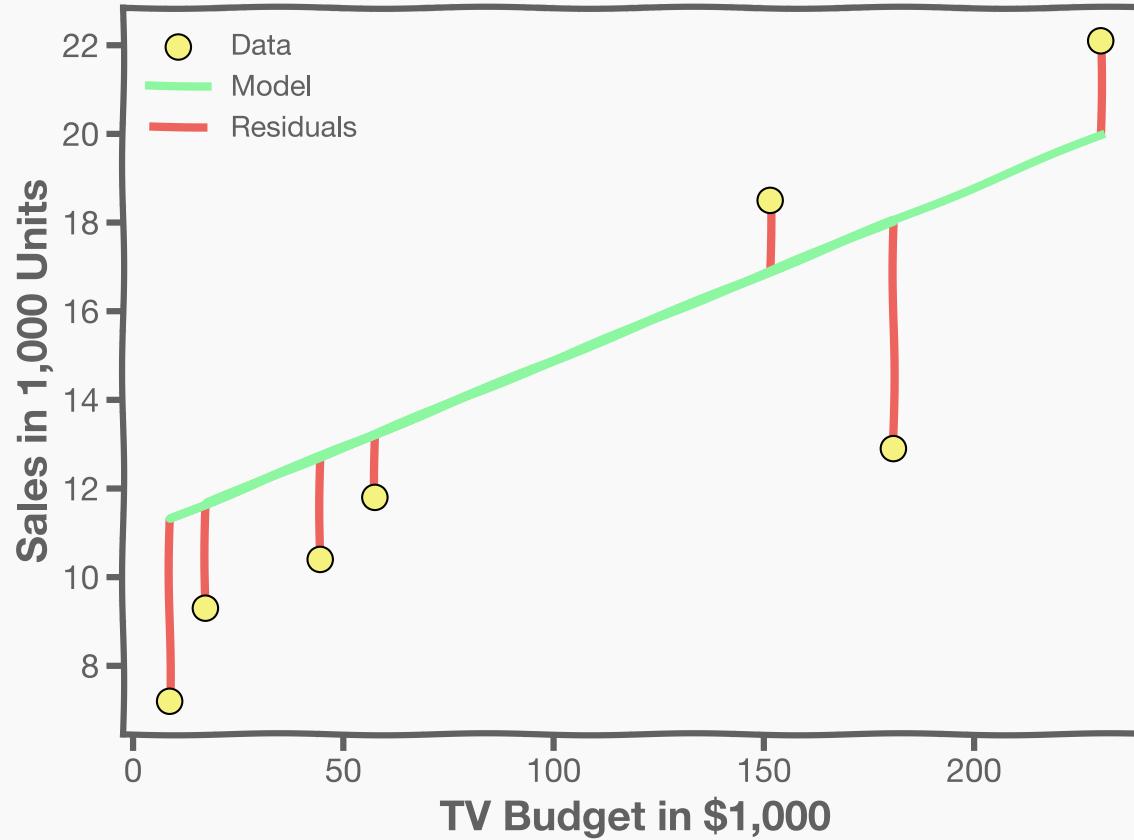
Estimate of the regression coefficients (cont)

Or this one?



Estimate of the regression coefficients (cont.)

Question: Which line is the best?



As before, for each observation (x_n, y_n) , the **absolute residuals**, $r_i = |y_i - \hat{y}_i|$ quantify the error at each observation.

Estimate of the regression coefficients (cont.)

AGAIN, we use the **MSE** as our **loss function**,

$$L(\beta_0, \beta_1) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

FIND THE VALUES
OF β_0 AND β_1 THAT
YIELD THE
SMALLEST VALUE OF
 L

We choose β_1 and β_0 that minimizes the prediction errors made by our model, i.e., minimize our loss function.

Then the optimal values, $\hat{\beta}_0$ and $\hat{\beta}_1$, should be:

$$\hat{\beta}_0, \hat{\beta}_1 = \operatorname{argmin}_{\beta_0, \beta_1} L(\beta_0, \beta_1).$$

WE CALL THIS
FITTING OR
TRAINING THE
MODEL

Introducing...



SK-Learn

Import sklearn's linear model
LinearRegression

```
>>> from sklearn.linear_model import LinearRegression
>>> df = pd.read_csv('Advertising.csv')
>>> X= df[['TV']].values
>>> y = df['Sales'].values
```

SK-Learn

```
>>> from sklearn.linear_model import LinearRegression  
>>> df = pd.read_csv('Advertising.csv')  
>>> X= df[['TV']].values  
>>> y = df['Sales'].values  
>>> reg = LinearRegression()  
>>> reg.fit(X, y)
```

Instantiate the model

Use the method `fit()` from the model `LinearRegression`. This method finds the values of β_0 and β_1

SK-Learn

```
>>> from sklearn.linear_model import LinearRegression  
>>> df = pd.read_csv('Advertising.csv')  
>>> X= df[['TV']].values  
>>> y = df['Sales'].values  
>>> reg = LinearRegression()  
>>> reg.fit(X, y)  
>>> reg.coef_  
array([[0.04665056]])  
>>> reg.intercept_  
array([7.08543108])  
>>> reg.predict(np.array([[100]]))  
array([[11.75048733]])
```

Use the fitted model (i.e. uses the values of β_0 and β_1 found in the `.fit()` to predict y .
$$y = \beta_0 + \beta_1 x$$

```
>>> reg.fit(X, y)
```



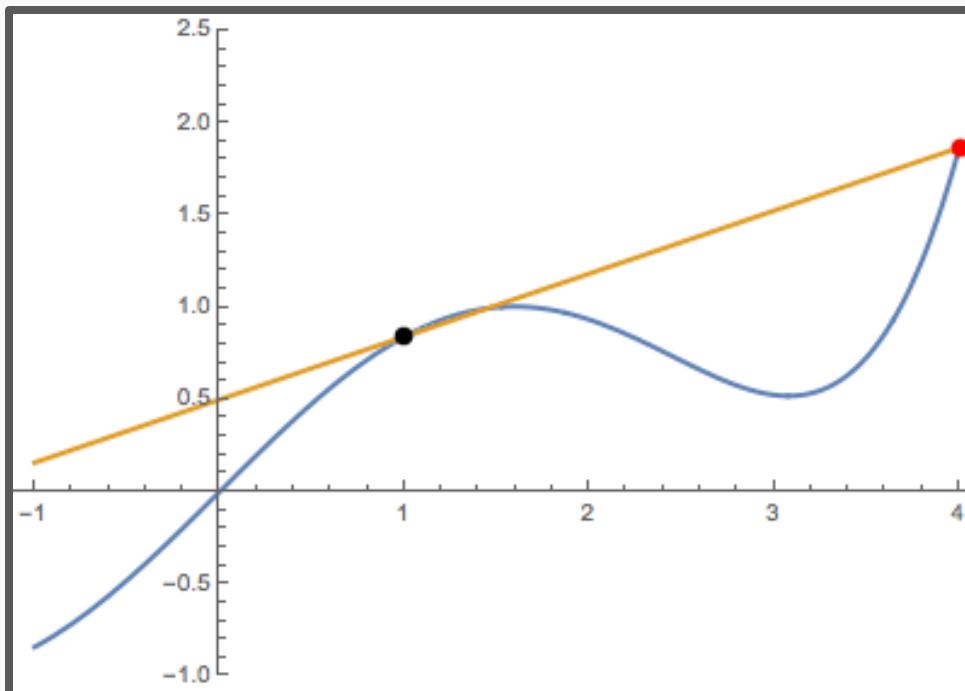
Што се случи



Derivative definition

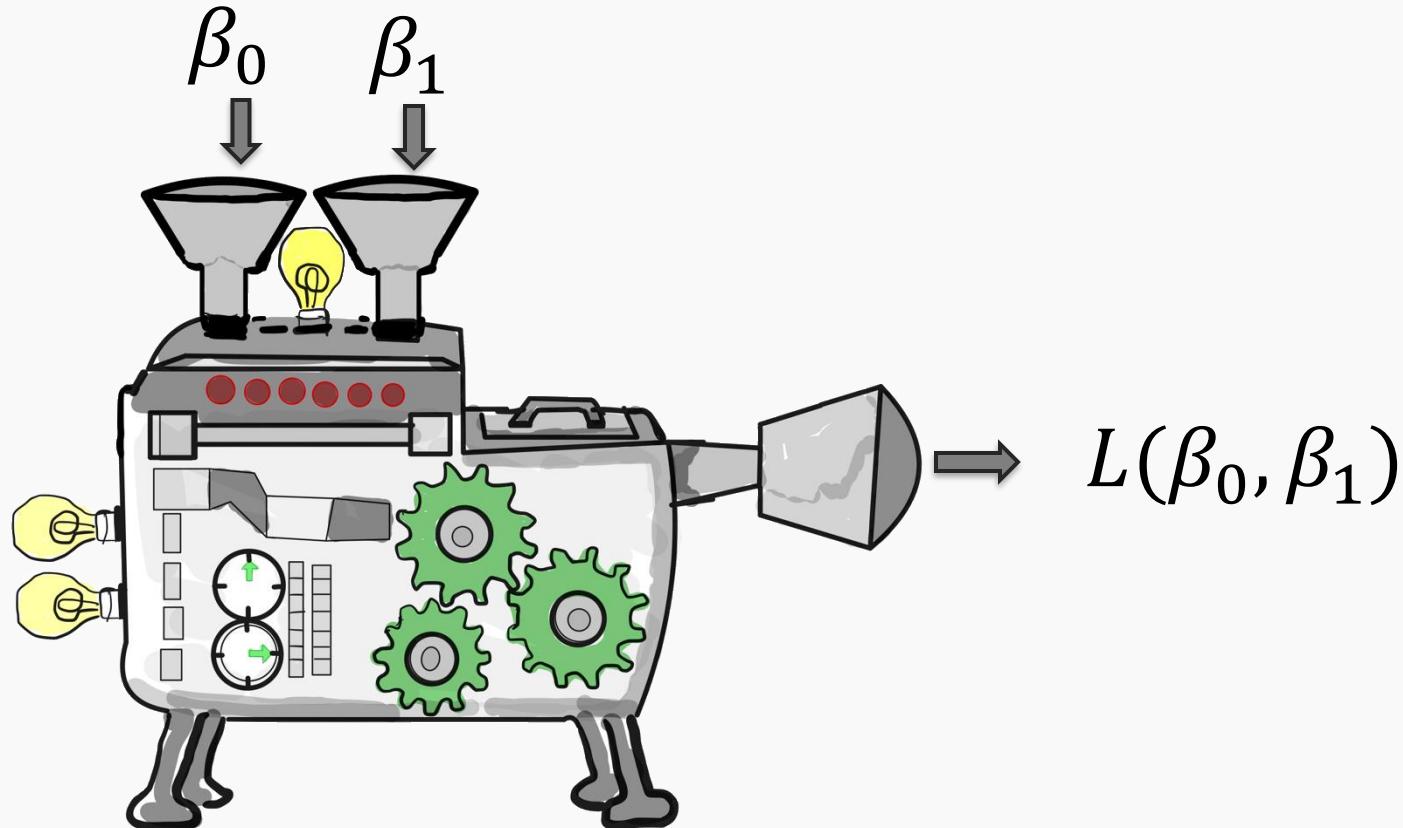
A **derivative** is the instantaneous rate of change of a single valued function. Given a function $f(x)$ the derivative can be defined as:

$$f'(x) = \frac{df}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h)-f(x)}{h}$$



Partial derivatives

For a loss function L that depends on β_0, β_1 we need the **partial derivatives**, $\frac{\partial L}{\partial \beta_i}$. Partial derivatives indicate the rate of change of the function with respect to one variable while keeping the others fixed.



$$\begin{array}{c} L(\beta_0, \beta_1) \\ \downarrow \qquad \downarrow \\ \frac{\partial L}{\partial \beta_0} \qquad \frac{\partial L}{\partial \beta_1} \end{array}$$

Partial derivative example

If $L(\beta_0, \beta_1) = (y - (\beta_1 x + \beta_0))^2$ then what is $\frac{\partial L}{\partial \beta_0}$?

Looks like we're going
to need the chain rule.
But what is it? I forgot



Partial derivative example

If $L(\beta_0, \beta_1) = (y - (\beta_1 x + \beta_0))^2$ then what is $\frac{\partial L}{\partial \beta_0}$?

$$\frac{\partial L(f(\beta_0))}{\partial \beta_0} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial \beta_0}$$



Partial derivative $\frac{\partial L}{\partial \beta_0}$

If $L(\beta_0, \beta_1) = (y - (\beta_1 x + \beta_0))^2$ then what is $\frac{\partial L}{\partial \beta_0}$?

$$L = \underbrace{(y - \beta_1 x - \beta_0)^2}_{f(L(\beta_0))f^2}$$

$$\frac{\partial L}{\partial \beta_0} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial \beta_0} \quad L = f^2 \Rightarrow \frac{\partial L}{\partial f} = 2f \quad f = y - \beta_1 x - \beta_0 \Rightarrow \frac{\partial f}{\partial \beta_0} = -1$$

$$\frac{\partial L}{\partial \beta_0} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial \beta_0} = -2f = -2(y - \beta_1 x - \beta_0)$$

Partial derivative $\frac{\partial L}{\partial \beta_1}$

If $L(\beta_0, \beta_1) = (y - (\beta_1 x + \beta_0))^2$ then what is $\frac{\partial L}{\partial \beta_1}$?

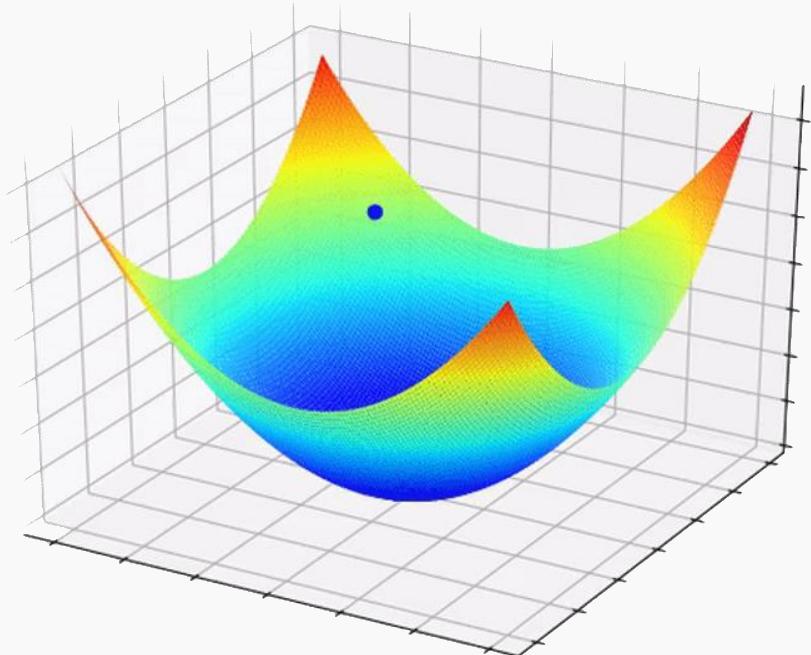
$$L = (y - \beta_1 x - \beta_0)^2$$

$$\frac{\partial L}{\partial \beta_1} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial \beta_1} \quad L = f^2 \Rightarrow \frac{\partial L}{\partial f} = 2f \quad f = y - \beta_1 x - \beta_0 \Rightarrow \frac{\partial f}{\partial \beta_1} = -x$$

$$\frac{\partial L}{\partial \beta_1} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial \beta_1} = -2x \cdot 2f = -2x(y - \beta_1 x - \beta_0)$$

Optimization

How does one minimize a loss function?



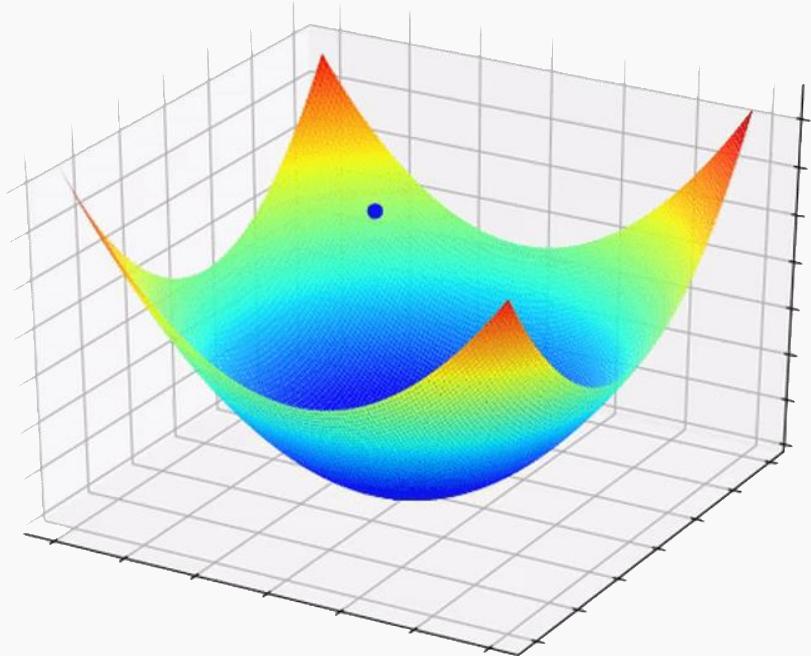
The global minima or maxima of $L(\beta_0, \beta_1)$ must occur at a point where the **gradient** (slope) is:

$$\nabla L = \left[\frac{\partial L}{\partial \beta_0}, \frac{\partial L}{\partial \beta_1} \right] = 0$$

- **Brute Force:** Try every combination
- **Closed-form Solution:** Solve the above equation for β_0, β_1
- **Greedy Algorithm:** Gradient Descent

Optimization

How does one minimize a loss function?



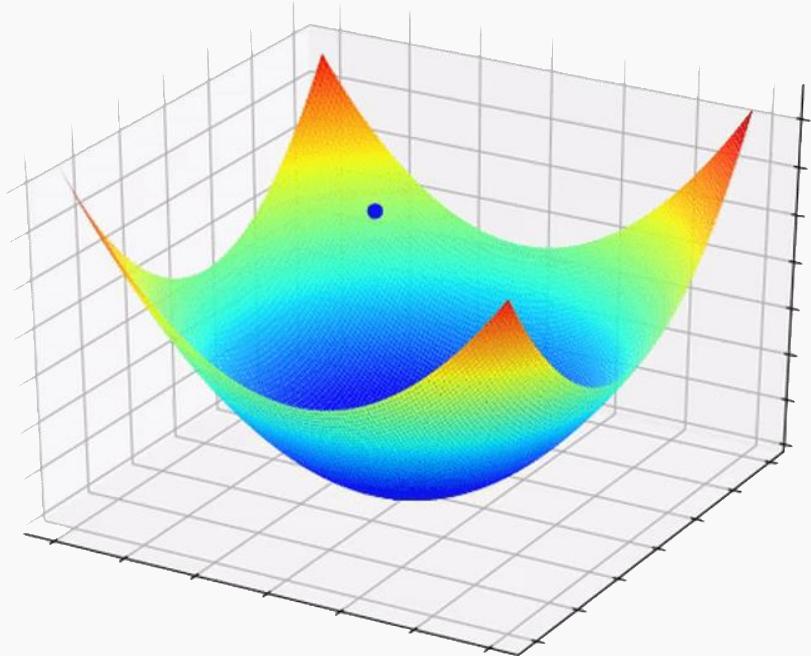
The global minima or maxima of $L(\beta_0, \beta_1)$ must occur at a point where the **gradient** (slope) is:

$$\nabla L = \left[\frac{\partial L}{\partial \beta_0}, \frac{\partial L}{\partial \beta_1} \right] = 0$$

- **Brute Force:** Try every combination
- **Closed-form Solution:** Solve the above equation for β_0, β_1
- **Greedy Algorithm:** Gradient Descent

Optimization

How does one minimize a loss function?



The gradient is a vector that contains all the partial derivatives of the function with respect to its variables. The nabla symbol (∇) is used to denote the gradient operation

The global minimum of $L(\beta_0, \beta_1)$ must occur at a point where the gradient (slope) is:

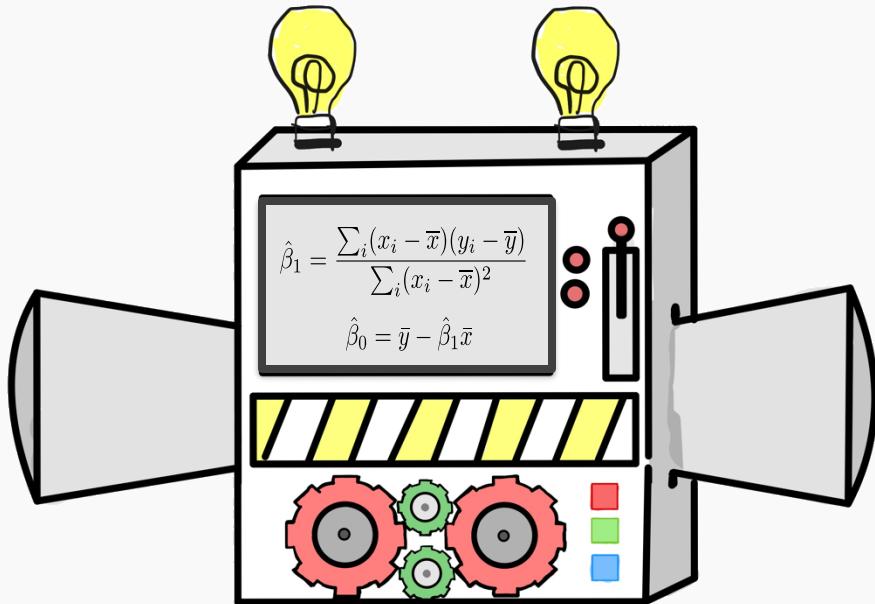
$$\nabla L = \left[\frac{\partial L}{\partial \beta_0}, \frac{\partial L}{\partial \beta_1} \right] = 0$$

- **Brute Force:** Try every combination
- **Closed-form Solution:** Solve the above equation for β_0, β_1
- **Greedy Algorithm:** Gradient Descent

Optimization

$$\nabla L = \left[\frac{\partial L}{\partial \beta_0}, \frac{\partial L}{\partial \beta_1} \right] = 0$$

$$\frac{\partial L}{\partial \beta_0} = -2(y - \beta_1 x - \beta_0) = 0$$



$$\frac{\partial L}{\partial \beta_1} = -2x(y - \beta_1 x - \beta_0) = 0$$

Optimization

Sum over the data Data: predictors values Average value of x Average value of y

$$\hat{\beta}_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Summary: Estimate of the regression coefficients

We use MSE as our loss function,

$$L(\beta_0, \beta_1) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \frac{1}{n} \sum_{i=1}^n [y_i - (\beta_0 + \beta_1 x_i)]^2$$

FIND THE VALUES
OF β_0 AND β_1 THAT
YIELD THE
SMALLEST VALUE OF
 L

We choose $\hat{\beta}_1$ and $\hat{\beta}_0$ in order to minimize the predictive errors made by our model, i.e. minimize our loss function.

Then the optimal values for $\hat{\beta}_0$ and $\hat{\beta}_1$ should be:

$$\hat{\beta}_0, \hat{\beta}_1 = \operatorname{argmin}_{\beta_0, \beta_1} L(\beta_0, \beta_1).$$

WE CALL THIS
FITTING OR
TRAINING THE
MODEL

Estimate of the regression coefficients: analytical solution

Take the gradient of the loss function and find gradient is zero: $\nabla L = \left[\frac{\partial L}{\partial \beta_0}, \frac{\partial L}{\partial \beta_1} \right] = 0$

Finding the exact solution only works for rare cases. Linear regression is one of such rare cases.

$$\hat{\beta}_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

where \bar{y} and \bar{x} are sample means.

The line:
is called the **regression line**.

$$\hat{Y} = \hat{\beta}_1 X + \hat{\beta}_0$$



Introduction to Regression

Part A - kNN



Pavlos Protopapas, Natesh Pillai and Chris Gumb



Lecture Outline

Part A: Statistical Modeling

k-Nearest Neighbors (kNN)

Part B: Model Fitness

How does the model perform predicting?

How do we choose from two different models?

Response and Predictor Variables

Predicting a Variable

Let's consider a scenario in which we aim to **predict** the value of one variable based on another variable or a set of other variables.

Examples:

Predicting the number of views that a **TikTok** video will receive next week, based on factors such as **video length**, **posting date**, and **previous view count**.



Predicting a Variable

Let's consider a scenario in which we aim to **predict** the value of one variable based on another variable or a set of other variables.

Examples:

Forecasting **which movies**, a **Netflix** user is likely to rate highly, considering their **previous movie ratings** and **demographic data**.



Working example

The [Advertising dataset](#) contains sales (in 1000 units) data for a specific product across 200 different markets. It also includes advertising budgets in \$1000 allocated to three different media channels: *TV, radio, and newspaper*, for each of those markets.

TV	radio	newspaper	sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9

Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani "

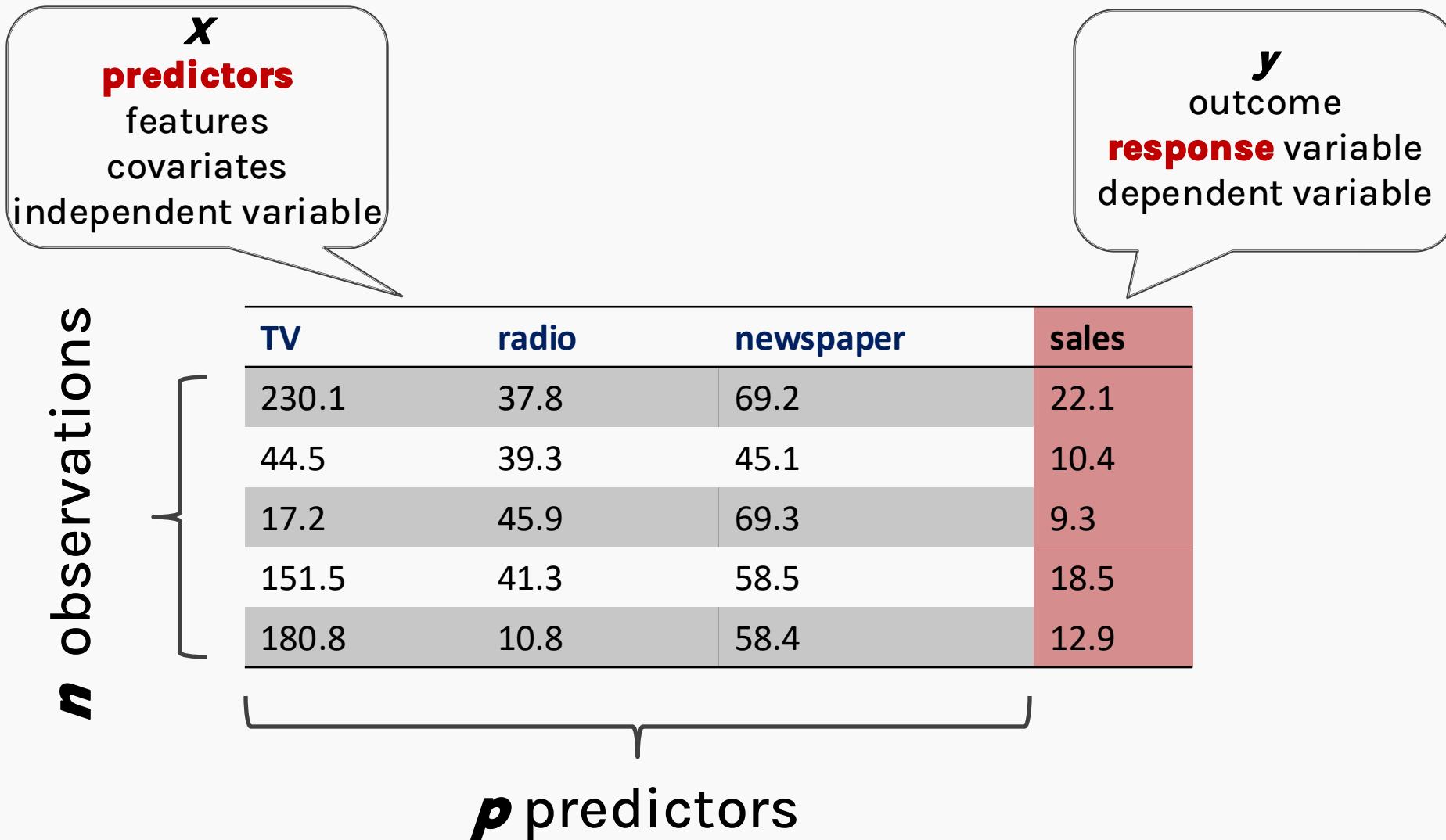
Response vs. Predictor Variables

Many of these problems exhibit an **asymmetry**: the variable we aim to predict may be **harder to measure**, more **significant**, or **directly or indirectly influenced** by other variables.

Therefore, we can classify variables into two categories:

- Variables whose values we aim to **predict**
- Variables **used** as inputs to inform our prediction

Response vs. Predictor Variables



X
predictors
features
covariates
independent variable

y
outcome
response variable
dependent variable

TV	radio	newspaper	sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9

n observations

p predictors

Response vs. Predictor Variables

$X = X_1, \dots, X_p$
 $X_j = x_{1j}, \dots, x_{ij}, \dots, x_{nj}$
predictors
features
covariates
independent variable

$y = y_1, \dots, y_i, \dots, y_n$
outcome
response variable
dependent variable

n observations

p predictors

TV	radio	newspaper	sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9

Response vs. Predictor Variables

This is called X : a.k.a.
The Design Matrix

n observations

TV	radio	newspaper
230.1	37.8	69.2
44.5	39.3	45.1
17.2	45.9	69.3
151.5	41.3	58.5
180.8	10.8	58.4

y :
The response variable

sales
22.1
10.4
9.3
18.5
12.9

Response vs. Predictor Variables

This is called X : a.k.a.
The Design Matrix

n observations

TV	radio	newspaper
230.1	37.8	69.2
44.5	39.3	45.1
17.2	45.9	69.3
151.5	41.3	58.5
180.8	10.8	58.4

Capital letters mean **matrices**,

y :
The response variable

sales
22.1
10.4
9.3
18.5
12.9

Response vs. Predictor Variables

This is called X : a.k.a.
The Design Matrix

n observations

TV	radio	newspaper
230.1	37.8	69.2
44.5	39.3	45.1
17.2	45.9	69.3
151.5	41.3	58.5
180.8	10.8	58.4

y
The response variable

sales
22.1
10.4
9.3
18.5
12.9

Capital letters mean **matrices**, lower case letters mean **vectors**

Sklearn expects certain dimensions

```
>>> X.shape  
(n, p)
```

n observations

TV	radio	newspaper
230.1	37.8	69.2
44.5	39.3	45.1
17.2	45.9	69.3
151.5	41.3	58.5
180.8	10.8	58.4

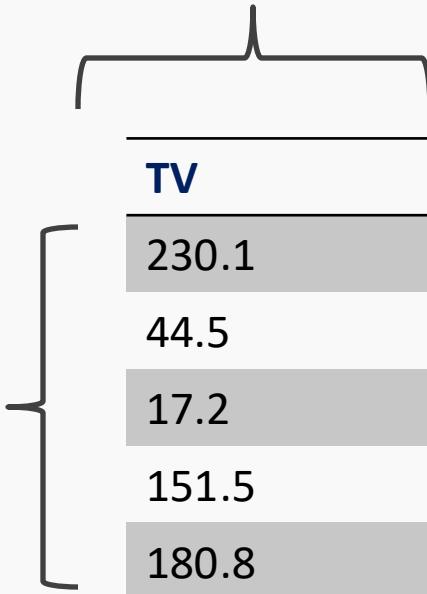
```
>>> y.shape  
(n,) OR (n, 1)
```

sales
22.1
10.4
9.3
18.5
12.9

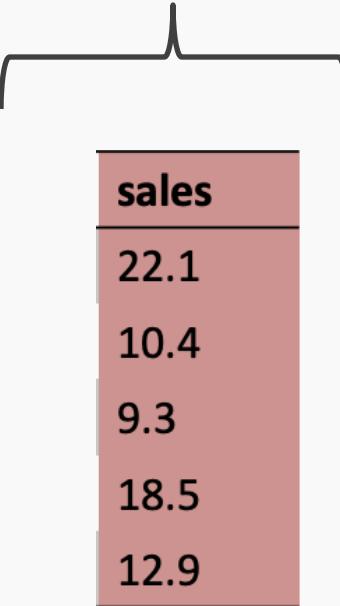
Sklearn expects certain dimensions

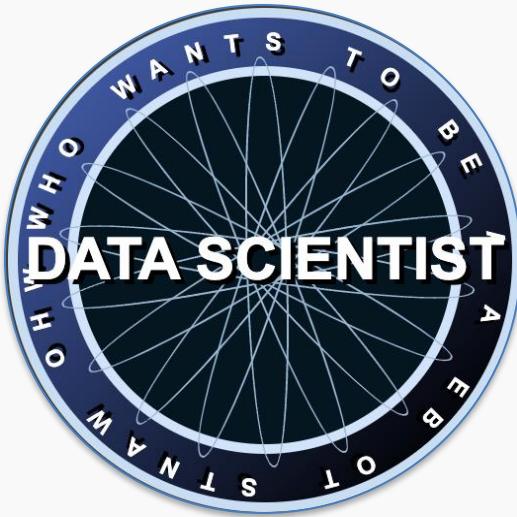
```
>>> X.shape  
(n,) OR (n, 1)
```

n observations



```
>>> y.shape  
(n,) OR (n, 1)
```





CS109A

GAME Time

Game time: Pandas Review



`df[['x']]` vs `df['x']`

Which of the statements below is correct?

Options:

- A. `df[['x']]` returns a `pd.Series` object whereas `df['x']` returns a `pd.DataFrame`.
- B. `df[['x']]` is invalid operation.
- C. `df[['x']]` returns a `pd.DataFrame` whereas `df['x']` returns a `pd.Series` object.
- D. `df['x']` is invalid operation.

Statistical Model

True vs. Statistical Model

- Imagine an ice cream cone so perfect that it captures every flavor, topping, and swirl of deliciousness. That is what a *true model* is.



True vs. Statistical Model

- Imagine an ice cream cone so perfect that it captures every flavor, topping, and swirl of deliciousness. That is what a *true model* is.
- But reality is like an ice cream shop with infinite flavors and toppings. Trying to fit all of them into one cone is **impossible**.



True vs. Statistical Model

- Imagine an ice cream cone so perfect that it captures every flavor, topping, and swirl of deliciousness. That is what a *true model* is.
- But reality is like an ice cream shop with infinite flavors and toppings. Trying to fit all of them into one cone is **impossible**.
- This is why we use *statistical models*: instead of trying to scoop the impossible sundae, we craft a tasty treat from the flavors we have.



True vs. Statistical Model

We assume that the response variable, Y , is related to the predictor variables, X , through an **unknown function** which can be generally expressed as:

$$Y = f(X) + \varepsilon$$

Here, f represents the unknown function expressing an underlying rule for relating Y to X . ε represents the random amount (unrelated to X) that Y differs from the rule $f(X)$.

A **statistical model** is any algorithm used to estimate f . We denote the estimated function as \hat{f} .

Prediction vs. Estimation

Inference Problems:

- The primary focus is on **obtaining \hat{f}** , which is an estimate of the true function f
- Objective: Understand the form and characteristics of \hat{f} .



Prediction Problems

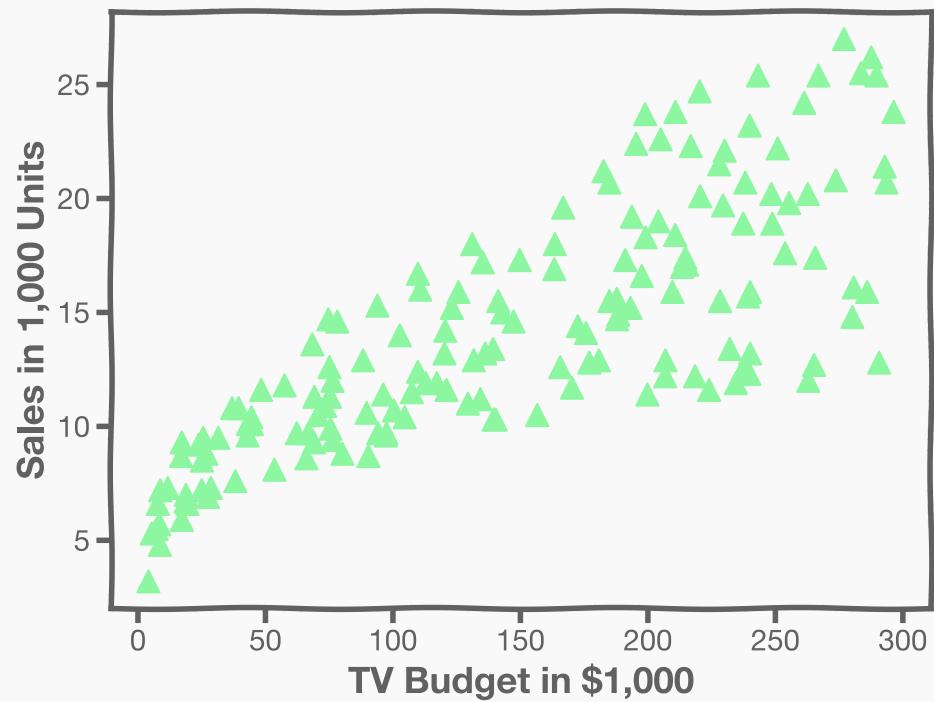
- The specific form of \hat{f} is less important than the **accuracy** of the predictions.
- Objective: Minimize the difference between predicted values \hat{y} and observed values y .



Example: predicting sales

Motivation: Predict Sales

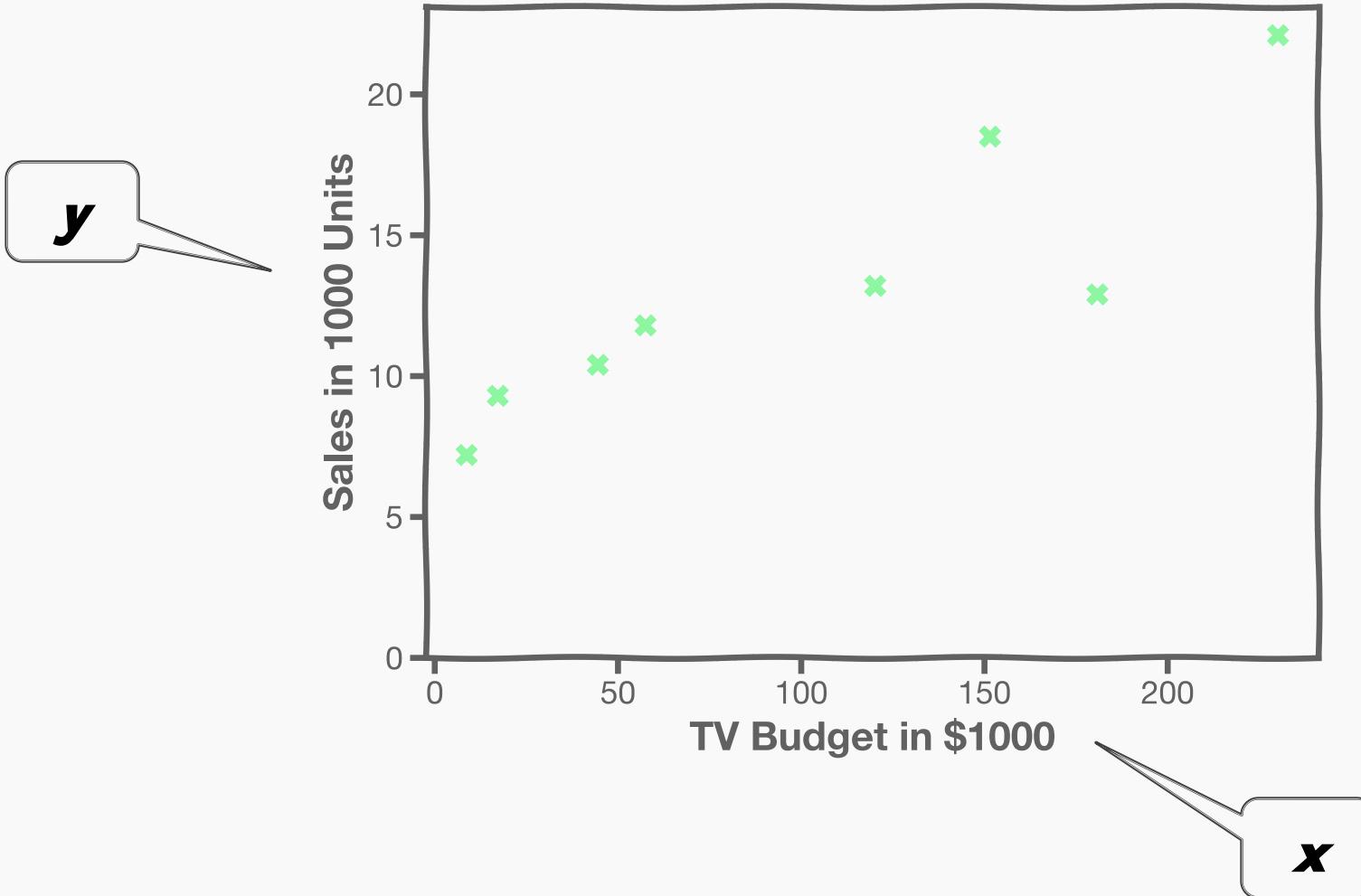
Build a model to **predict** sales based on TV budget



The response, y , is the sales.

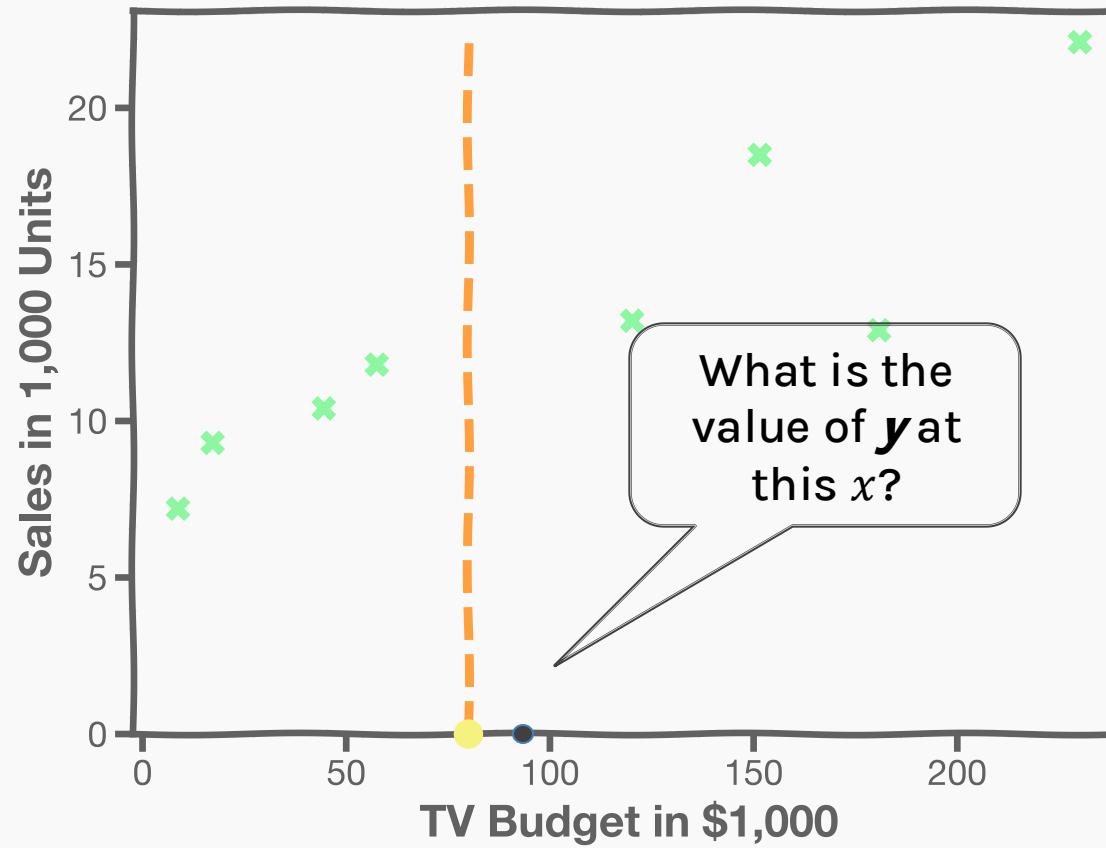
The predictor, x , is TV budget.

Statistical Model



Statistical Model

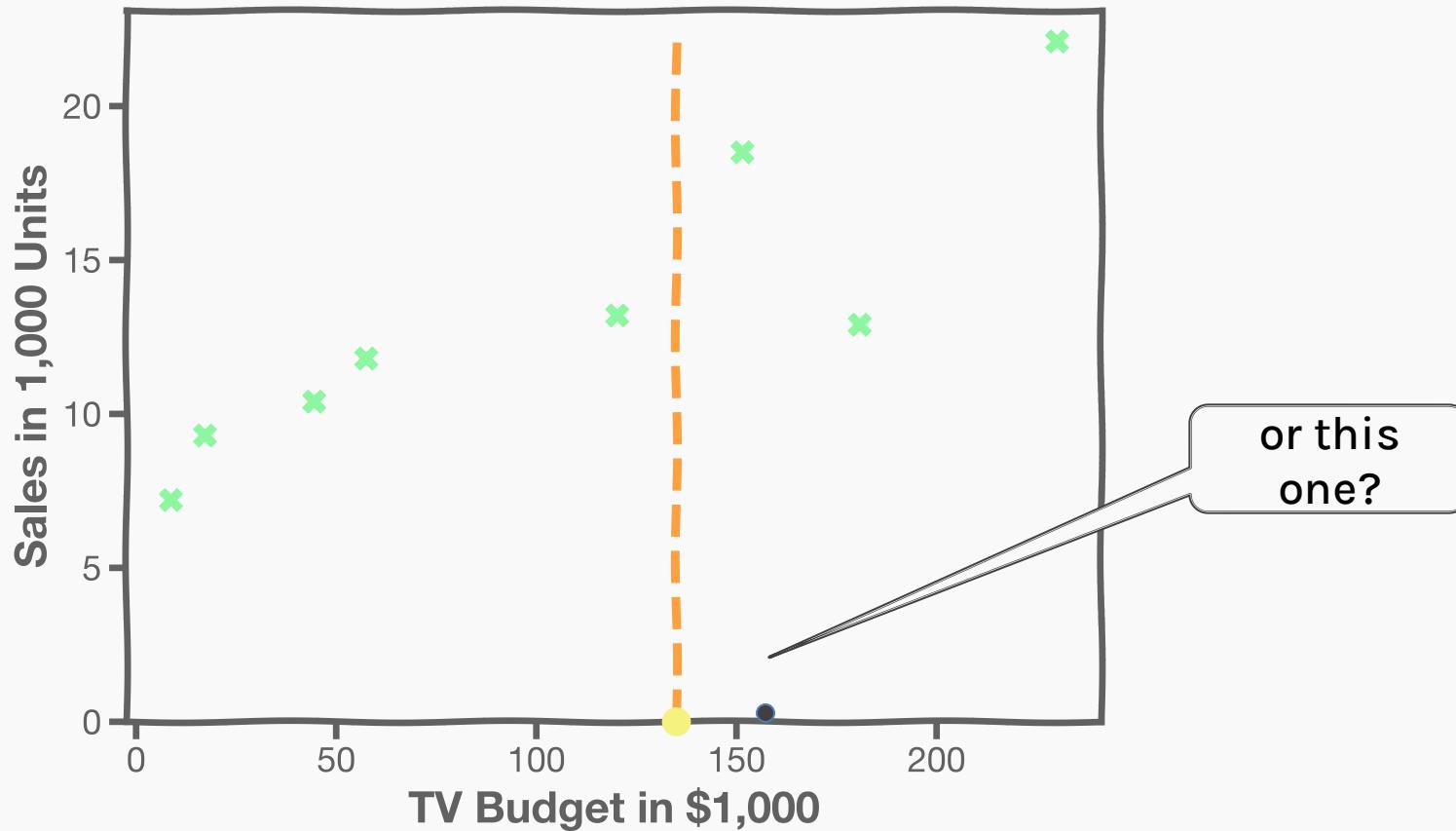
How do we predict y for some x ?





Statistical Model

How do we predict y for some x ?



Game time: Choices for model



Which of the following 5 methods could be used to predict the value of y given x ?

Options:

- A. Utilize a Convolutional Neural Network (CNN).
- B. Use a Linear Regression Model with a slope of 3 and an intercept of 2.
- C. Identify examples that closely resemble the input data point.
- D. Consult a TF during office hours for the answer.
- E. Calculate the average value of y from the available data points.

Game time: Choices for model



Which of the following 5 methods could be used to predict the value of y given x ?

Options:

- A. Utilize a Convolutional Neural Network (CNN).
- B. Use a Linear Regression Model with a slope of 3 and an intercept of 2.
- C. Identify examples that closely resemble the input data point.
- D. Consult a TF during office hours for the answer.
- E. Calculate the average value of y from the available data points.

Game time: Choices for model



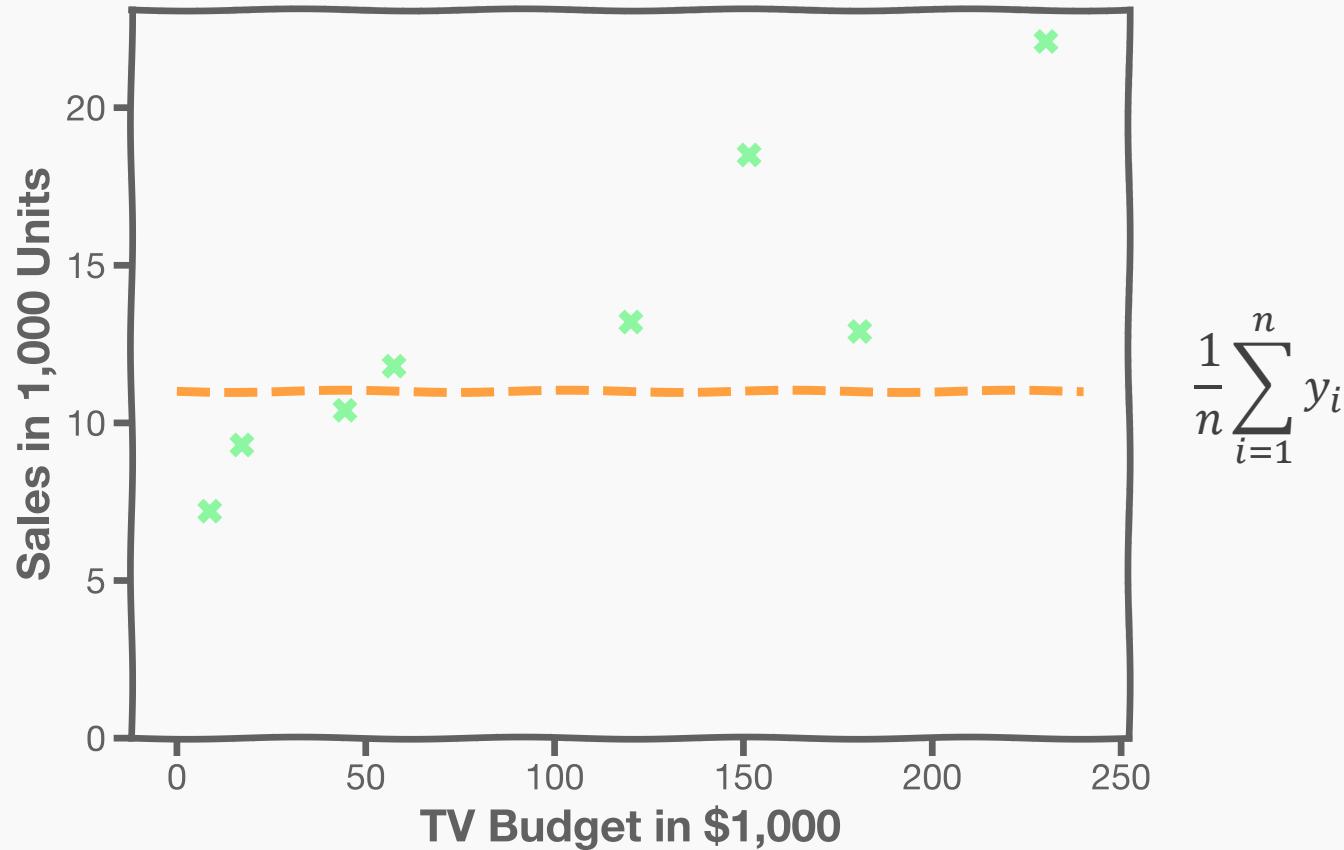
Which of the following 5 methods could be used to predict the value of y given x ?

Options:

- A. Utilize a Convolutional Neural Network (CNN).
- B. Use a Linear Regression Model with a slope of 3 and an intercept of 2.
- C. Identify examples that closely resemble the input data point.
- D. Consult a TF during office hours for the answer.
- E. Calculate the average value of y from the available data points.

Statistical Model

A simple idea is to take the mean of all y 's: $\frac{1}{n} \sum_{i=1}^n y_i$



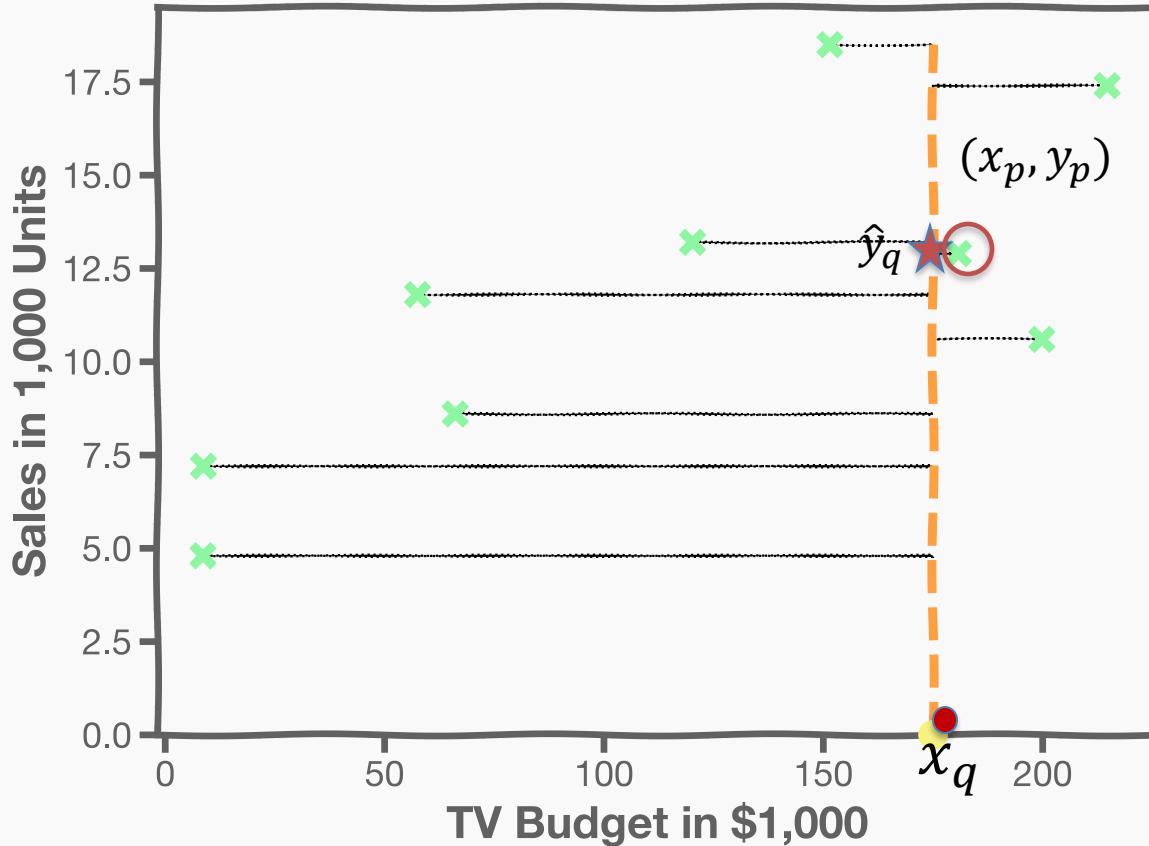
K-Nearest Neighbors

k-Nearest Neighbors – kNN



X = new patient

k-Nearest Neighbors – kNN



What is \hat{y}_q at some x_q ?

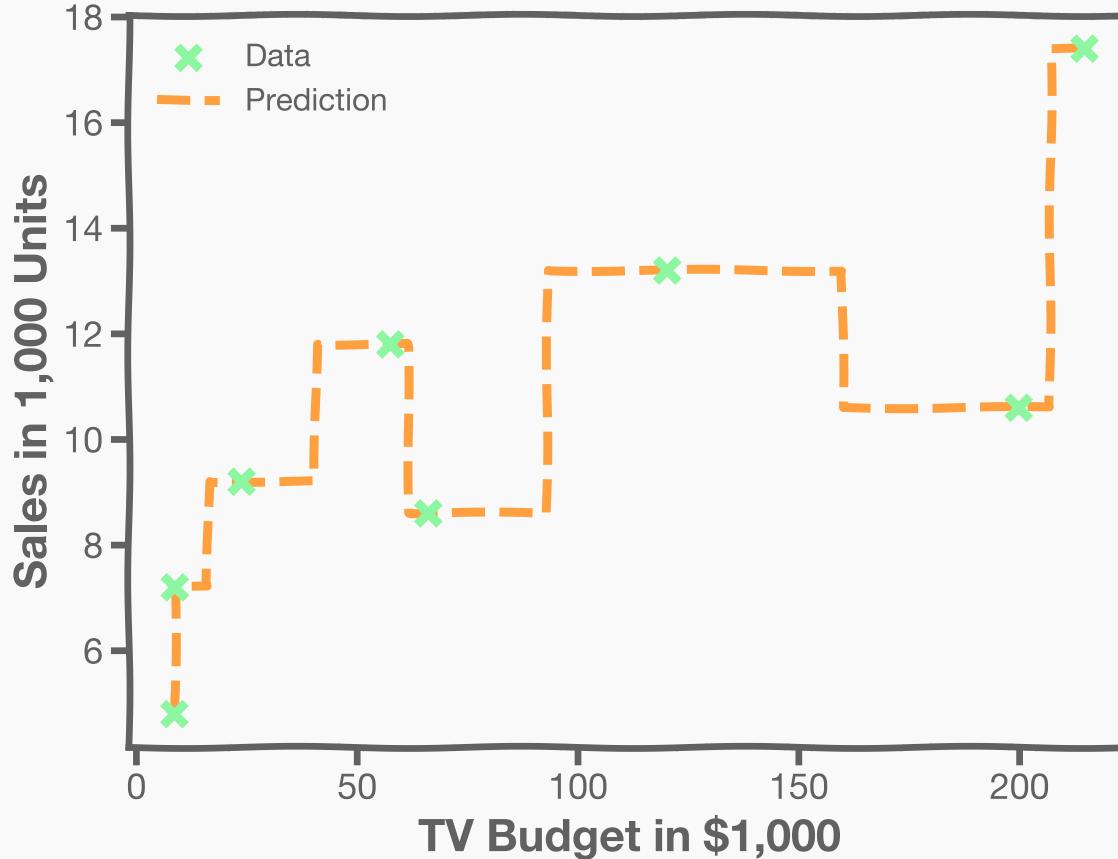
Find distances to
all other points
 $D(x_q, x_i)$

Find the nearest
neighbor, (x_p, y_p)

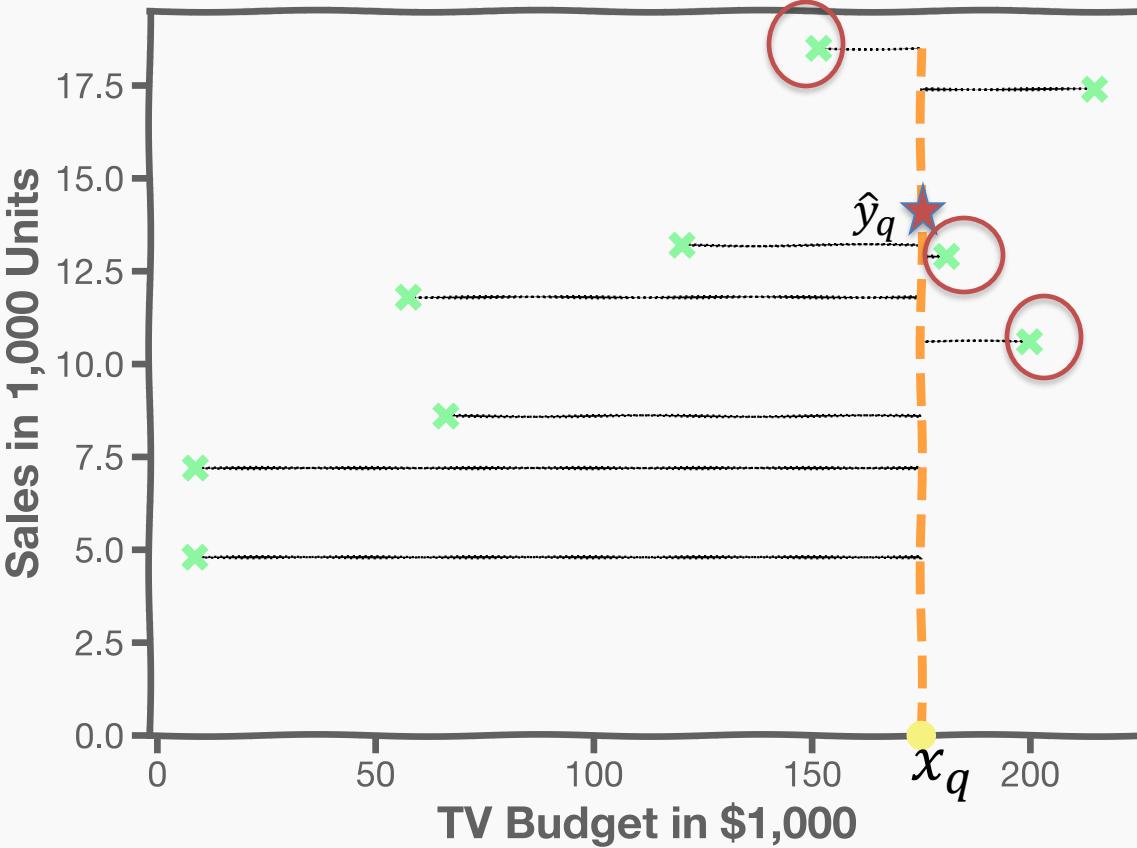
Predict $\hat{y}_q = y_p$

k-Nearest Neighbors - kNN

Do the same for “all” x' s



k-Nearest Neighbors – kNN



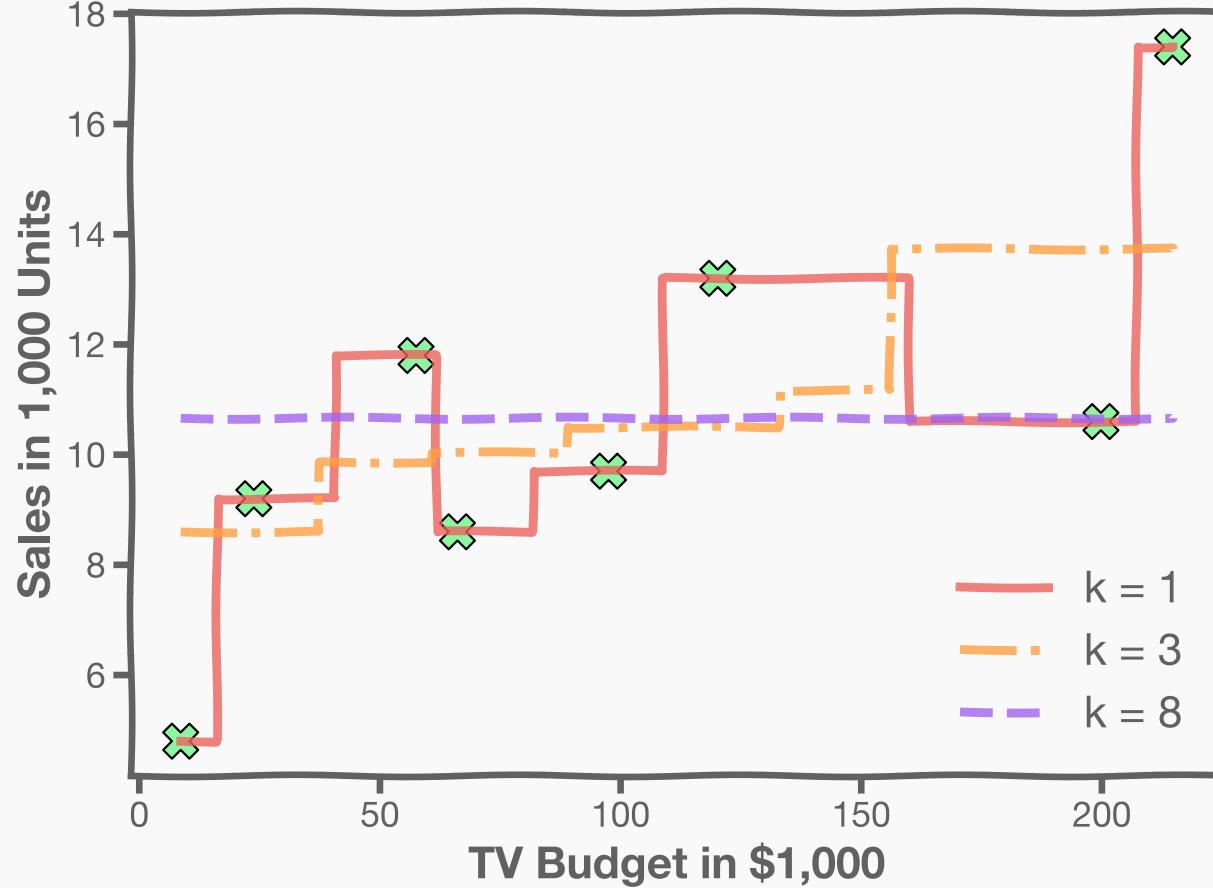
What is \hat{y}_q at some x_q ?

Find distances to
all other points
 $D(x_q, x_i)$

Find the k-nearest
neighbors, x_{q_1}, \dots, x_{q_k}

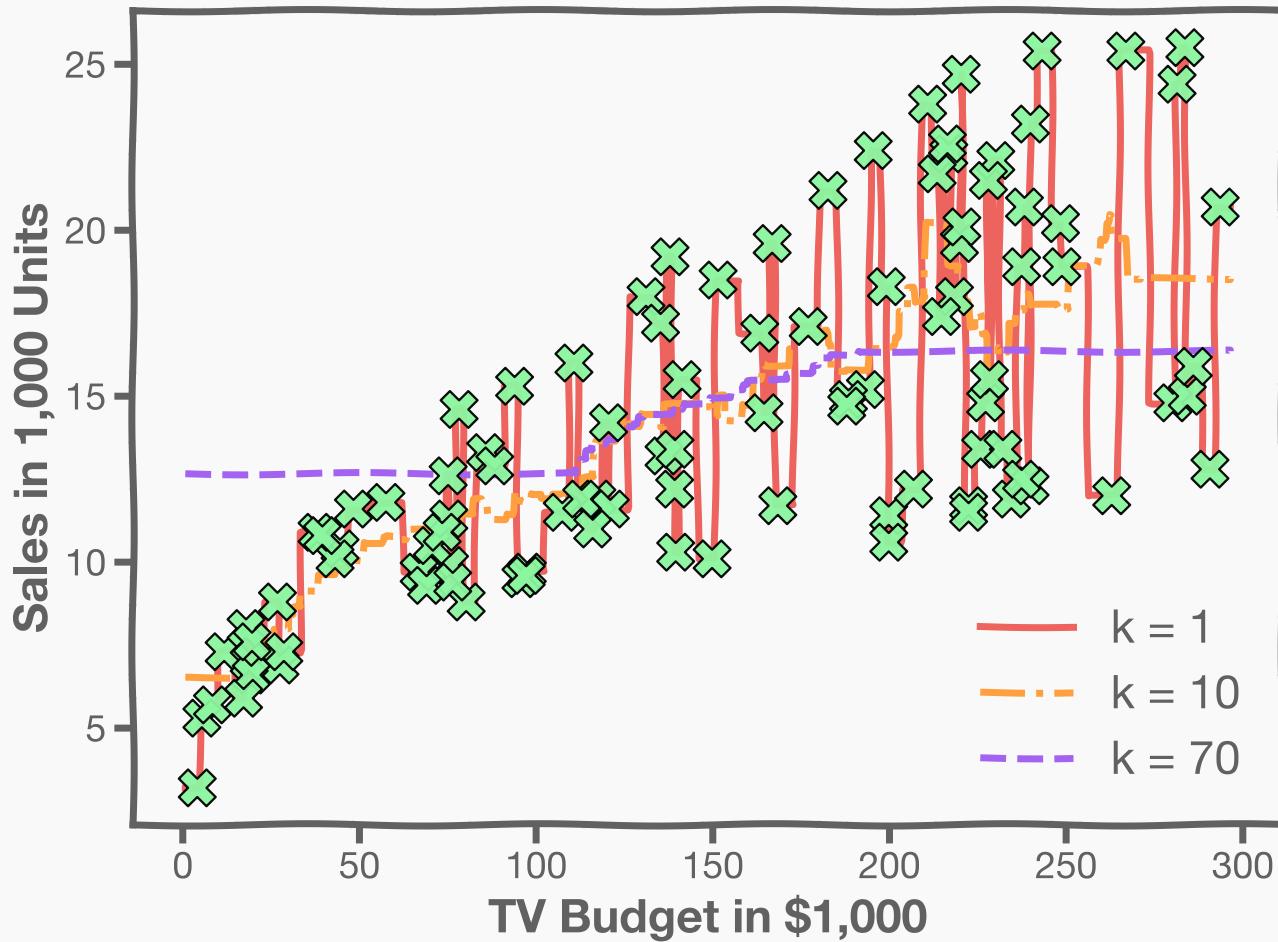
Predict $\hat{y}_q = \frac{1}{k} \sum_i^k y_{q_i}$

k-Nearest Neighbors – kNN



k-Nearest Neighbors – kNN

We can try different k-models on more data



k-Nearest Neighbors – kNN

The **very human way** of decision making by similar examples. kNN is a **non-parametric** learning algorithm.

The k-Nearest Neighbor Algorithm:

Given a dataset $D = \{(x^{(1)}, y^{(1)}), \dots, (x^{(N)}, y^{(N)})\}$, for every new X :

1. Find the k-number of observations in D most similar to X :

$$\{(x^{(n_1)}, y^{(n_1)}), \dots, (x^{(n_k)}, y^{(n_k)})\}$$

These are called the **k-nearest neighbors** of x

2. Average the output of the k-nearest neighbors of x

$$\hat{y} = \frac{1}{K} \sum_{k=1}^K y^{(n_k)}$$

Exercises + Quizzes Review

- **Pre-class quiz:** Not graded (see  emoji).
- **Post-class Q&A:** Not graded (see  emoji).
- **Attendance:** For every **8 sessions attended**, you earn an additional late day.

Note: This does not apply to Extension School students.

- **Quizzes:** Must be completed before the next lecture. The lowest 1/3 of quiz grades (including missed ones) will be dropped.

Story Board

Emoji key:

-  Instructor-led demonstration
-  Student's exercise in class
-  Reference code - No grading for this activity (Optional)
-  Student's exercise after class
-  No grading for this activity (Optional)
-  (One attempt only)

Before class:

-  Pre-Class Reading
-  Pre-Class Quiz

During Class:

-  Attendance
-  Introduction to Regression, kNN Regression
-  Simple Data Plotting
-  Simple kNN Regression
-  Error evaluation and Model Comparison

After class:

-  Post-Class Quiz
-  Post class Q&A

Exercises + Quizzes Review

- **Exercises:** Sometimes led by the instructor , sometimes done in class  or at home .
- Regardless, they are **due by the beginning of the next lecture.**

Note: Exercises are only counted in the final grade if they improve your grade. Otherwise, their weight is shifted to quizzes (from 8% to 10%).

Story Board

Emoji key:

-  Instructor-led demonstration
-  Student's exercise in class
-  Reference code - No grading for this activity (Optional)
-  Student's exercise after class
-  No grading for this activity (Optional)
-  (One attempt only)

Before class:

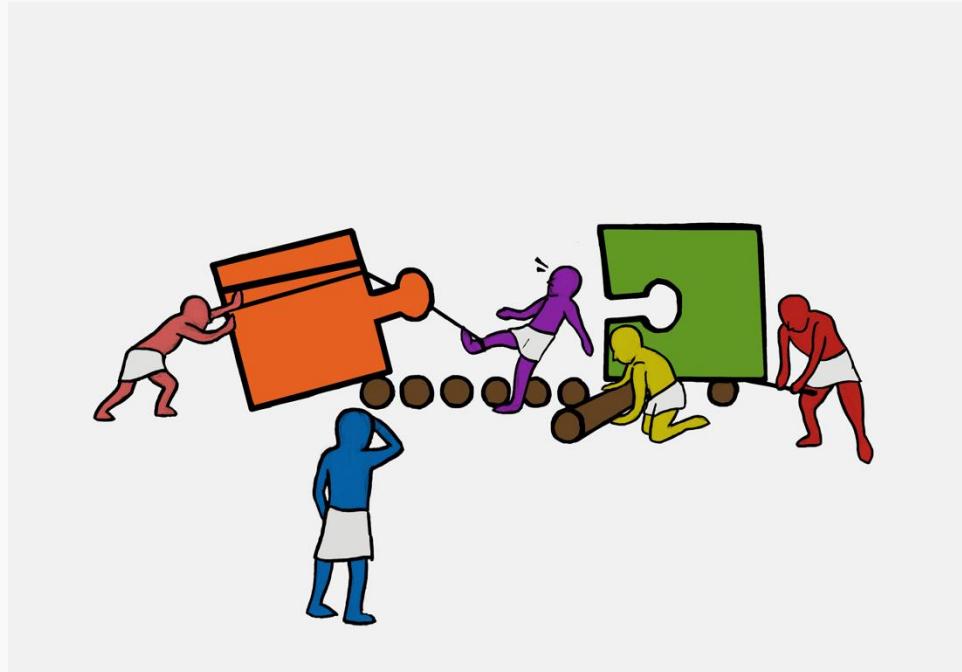
-  Pre-Class Reading
-  Pre-Class Quiz

During Class:

-  Attendance
-  Introduction to Regression, kNN Regression
-  Simple Data Plotting
-  Simple kNN Regression
-  Error evaluation and Model Comparison

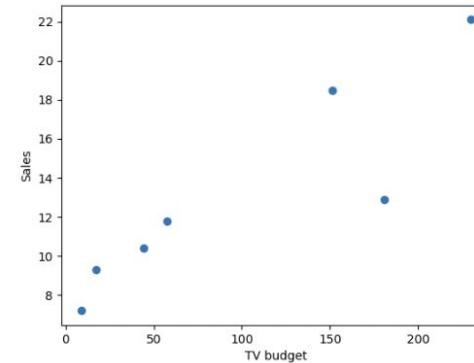
After class:

-  Post-Class Quiz
-  Post class Q&A



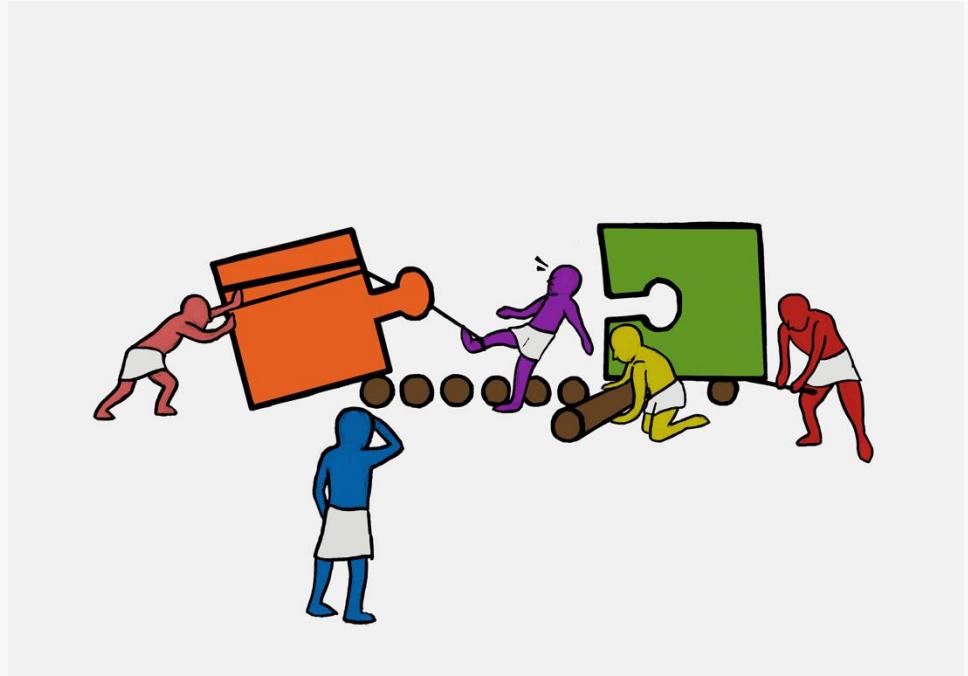
👨‍💻 Exercise: A.1 - Simple Data Plotting

The aim of this exercise is to **plot** TV Ads vs Sales based on the Advertisement dataset which should look similar to the graph given below.



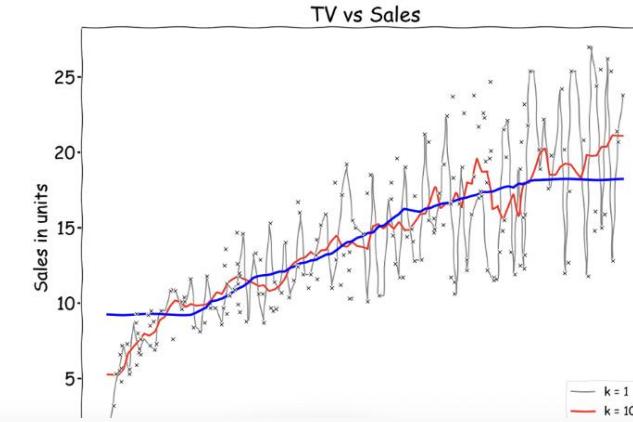
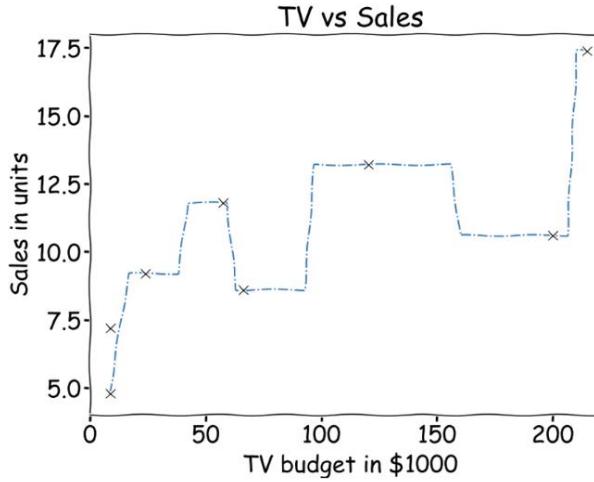
Instructions:

- Read the Advertisement data and view the top rows of the dataframe to get an understanding of the data and the columns.
- Select the first 7 observations and the columns `TV` and `sales` to make a new data frame.



🏋️ Exercise: A.2 - Simple kNN Regression

The goal of this exercise is to **re-create the plots** given below. You would have come across these graphs in the lecture as well.



Lecture #2: Data and Viz

CS109A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai and Chris Gumb



Lecture Outline: Data, Summaries, and Visuals

- What (is)are Data?
- Exploratory Data Analysis (EDA)
 - Descriptive Statistics
 - Basic Visualizations
- Historical Interlude
- Effective Visualizations
- Thanks to Kevin Rader for some of the material.

Reading: Ch. 1 in *An Introduction to Statistical Learning* (ISL)

The Data Science Process

Recall the data science process:

Today we will begin introducing
the data collection and data
exploration steps.

Ask an interesting question

Get the Data

Explore the Data

Model the Data

Communicate/Visualize the Results

What are data?

“A datum is a single measurement of something on a scale that is understandable to both the recorder and the reader. Data are multiple such measurements.”

Claim: everything is (can be) data!



Where do data come from?

- **Internal sources:** already collected by or is part of the overall data collection of your organization.
For example: business-centric data that is available in the organization data base to record day to day operations; scientific or experimental data.
- **Existing External Sources:** available in ready to read format from an outside source for free or for a fee.
For example: public government databases, stock market data, Yelp reviews, [your favorite sport]-reference, Kaggle.
- **External Sources Requiring Collection Efforts:** available from external source but acquisition requires special processing.
For example: data appearing only in print form, or data on websites.

Ways to gather online data

How to get data generated, published or hosted online:

- **API (Application Programming Interface):** using a prebuilt set of functions developed by a company to access their services. Often pay to use. For example: Google Map API, Facebook API, Twitter API
- **RSS (Rich Site Summary):** summarizes frequently updated online content in standard format. Free to read if the site has one. For example: news-related sites, blogs
- **Web scraping:** using software, scripts or by-hand extracting data from what is displayed on a page or what is contained in the HTML file (often in tables).

Web scraping

- **Why do it?** Older government or smaller news sites might not have APIs for accessing data, or publish RSS feeds or have databases for download. Or, you don't want to pay to use the API or the database.
- **How do you do it?** See HW1 (beautifulsoup)
- **Should you do it?**
 - You just want to explore: Are you violating their terms of service? Privacy concerns for website and their clients?
 - You want to publish your analysis or product: Do they have an API or fee that you are bypassing? Are they willing to share this data? Are you violating their terms of service? Are there privacy concerns?

Types of data

What kind of values are in your data (data types)?

Simple or atomic:

- **Numeric:** integers, floats
- **Boolean:** binary or true false values
- **Strings:** sequence of symbols

Data types

What kind of values are in your data (data types)? Compound, composed of a bunch of atomic types:

- **Date and time:** compound value with a specific structure
- **Lists:** a list is a sequence of values
- **Dictionaries:** A dictionary is a collection of key-value pairs, a pair of values $x: y$ where x is usually a string called the key representing the “name” of the entry, and y is a value of any type.

Example: Student record: what are x and y ?

- First: Natesh
- Last: Pillai
- Classes: [CS-1009A]

Data storage

How is your data represented and stored (data format)?

- **Tabular Data:** a dataset that is a two-dimensional table, where each row typically represents a single data record, and each column represents one type of measurement (csv, dat, xlsx, etc.).
- **Structured Data:** each data record is presented in a form of a [possibly complex and multi-tiered] dictionary (json, xml, etc.)
- **Semistructured Data:** not all records are represented by the same set of keys or some data records are not represented using the key-value pair structure.

Tabular Data

In tabular data, we expect each record or observation to represent a set of measurements of a single object or event. Here's an example:

In [4]:

```
imdb = pd.read_csv('imdb_top_1000.csv')
imdb.head()
```

Out[4]:

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating	Overview	Meta_score	Director	Star1
0	https://m.media-amazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	A	142 min	Drama	9.3	Two imprisoned men bond over a number of years...	80.0	Frank Darabont	Tim Robbins
1	https://m.media-amazon.com/images/M/MV5BM2MyNj...	The Godfather	1972	A	175 min	Crime, Drama	9.2	An organized crime dynasty's aging patriarch ...	100.0	Francis Ford Coppola	Marlon Brando
2	https://m.media-amazon.com/images/M/MV5BMTMxNT...	The Dark Knight	2008	UA	152 min	Action, Crime, Drama	9.0	When the menace known as the Joker wreaks havoc...	84.0	Christopher Nolan	Christian Bale
3	https://m.media-amazon.com/images/M/MV5BMWMwMG...	The Godfather: Part II	1974	A	202 min	Crime, Drama	9.0	The early life and career of Vito Corleone	90.0	Francis Ford Coppola	Al Pacino

Tabular Data

In [4]:

```
imdb = pd.read_csv('imdb_top_1000.csv')
imdb.head()
```

Out[4]:

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating	Overview	Meta_score	Director	Star1
0	https://m.media-amazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	A	142 min	Drama	9.3	Two imprisoned men bond over a number of years...	80.0	Frank Darabont	Tim Robbins
1	https://m.media-amazon.com/images/M/MV5BM2MyNj...	The Godfather	1972	A	175 min	Crime, Drama	9.2	An organized crime dynasty's aging patriarch t...	100.0	Francis Ford Coppola	Marlon Brando
2	https://m.media-amazon.com/images/M/MV5BMTMxNT...	The Dark Knight	2008	UA	152 min	Action, Crime, Drama	9.0	When the menace known as the Joker wreaks havo...	84.0	Christopher Nolan	Christian Bale
3	https://m.media-amazon.com/images/M/MV5BMWMwMG...	The Godfather: Part II	1974	A	202 min	Crime, Drama	9.0	The early life and career of Vito Corleone	90.0	Francis Ford Coppola	Al Pacino

Each type of measurement is called a **variable** or an **attribute** of the data (e.g. seq_id, status and duration are variables or attributes). The number of attributes is called the **dimension**. These are often called **features**.

We expect each table to contain a set of **records** or **observations** of the same kind of object or event.

Types of Data

We'll see later that it's important to distinguish between classes of variables or attributes based on the type of values they can take on.

- **Quantitative variable:** is numerical and can be either:
 - **discrete** - a finite number of values are possible in any bounded interval. For example: “Number of siblings” is a discrete variable
 - **continuous** - an infinite number of values are possible in any bounded interval. For example: “Height” is a continuous variable
- **Categorical variable:** no inherent order among the values For example: “What kind of pet do you have?” is a categorical variable

Common Issues

Common issues with data:

- Missing values: how do we fill in?
- Wrong values: how can we detect and correct?
- Messy format
- Not usable: the data cannot answer the question posed

Messy Data

The following is a table accounting for the number of produce deliveries over a weekend.

What are the variables in this dataset? What object or event are we measuring?

	Friday	Saturday	Sunday	ID	Time	Day	Number
Morning	15	158	10	1	Morning	Friday	15
Afternoon	2	90	20	2	Morning	Saturday	158
Evening	55	12	45	3	Morning	Sunday	10
What's the issue? How do we fix it?				4	Afternoon	Friday	2
				5	Afternoon	Saturday	9
				6	Afternoon	Sunday	20
				7	Evening	Friday	55
				8	Evening	Saturday	12
				9	Evening	Sunday	45

Tabular = Happy Pavlos



Common causes of messiness are:

- Column headers are values, not variable names
- Variables are stored in both rows and columns
- Multiple variables are stored in one column/entry
- Multiple types of experimental units stored in same table

In general, we want each file to correspond to a dataset, each column to represent a single variable and each row to represent a single observation.

We want to **tabularize** the data. This makes Python happy.

Lecture Outline: Data, Summaries, and Visuals

- What are Data?
- Exploratory Data Analysis (EDA)
 - Descriptive Statistics
 - Basic Visualizations
- Historical Interlude
- Effective Visualizations

Reading: Ch. 1 in *An Introduction to Statistical Learning* (ISL)

Basics of Sampling

Population versus sample:

- A **population** is the entire set of objects or events under study. Population can be hypothetical “all students” or all students in this class.
- A **sample** is a “representative” subset of the objects or events under study. Needed because it’s impossible or intractable to obtain or compute with population data.

Biases in samples:

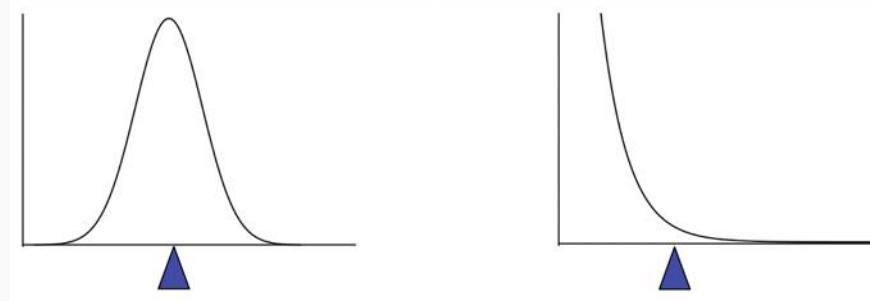
- **Selection bias:** some subjects or records are more likely to be selected
- **Volunteer/nonresponse bias:** subjects or records who are not easily available are not represented

PILLA Examples?

Sample mean

The **mean** of a set of n observations of a variable is denoted \bar{x} and is defined as:

$$\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{n} = \frac{1}{n} \sum_{i=1}^n x_i$$



The mean describes what a “typical” sample value looks like, or where is the “center” of the distribution of the data.

Key theme: there is always uncertainty involved when calculating a sample mean to estimate a population mean.

Sample median

The **median** of a set of n number of observations in a sample, ordered by value, of a variable is defined by

$$\text{Median} = \begin{cases} x_{(n+1)/2} & \text{if } n \text{ is odd} \\ \frac{x_{n/2} + x_{(n+1)/2}}{2} & \text{if } n \text{ is even} \end{cases}$$

Example (already in order):

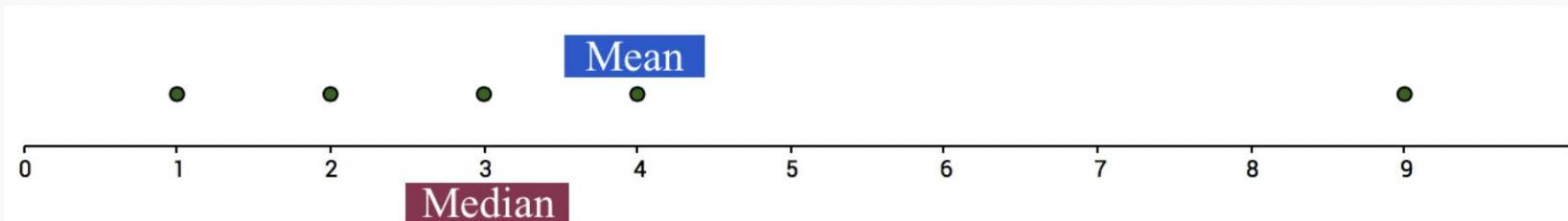
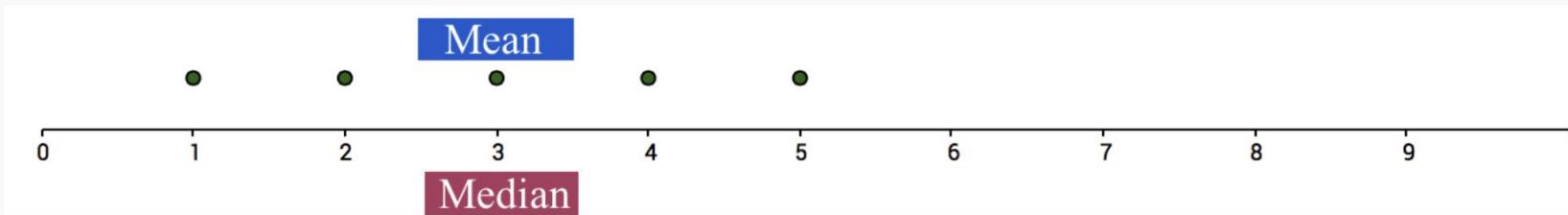
Ages: 17, 19, 21, 22, 23, 23, 23, 38

$$\text{Median} = (22+23)/2 = 22.5$$

The median also describes what a typical observation looks like, or where is the center of the distribution of the sample of observations.

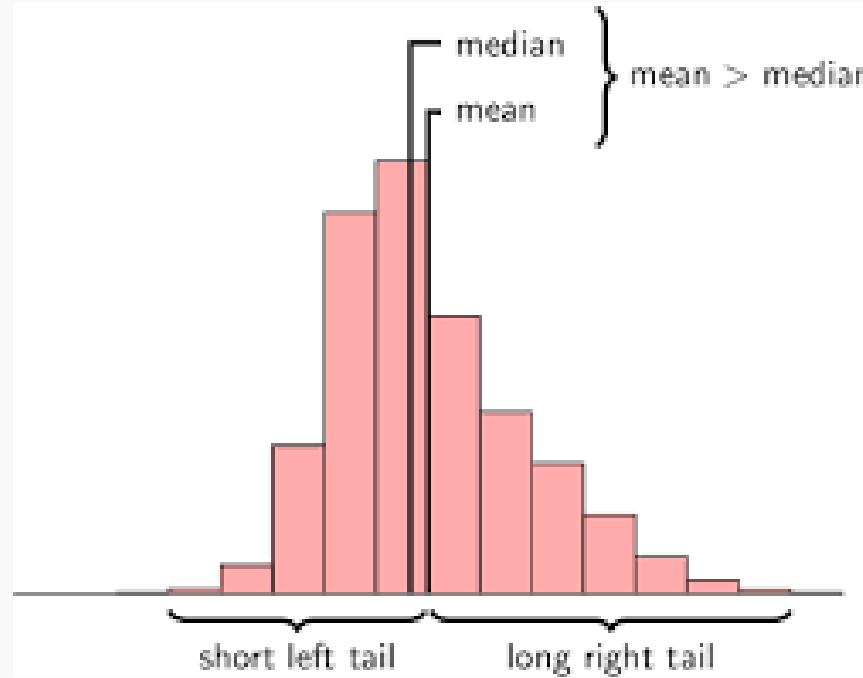
Mean vs. Median

The mean is sensitive to extreme values (**outliers**)



Mean, median, and skewness

The mean is sensitive to outliers:



The above distribution is called **right-skewed** since the mean is greater than the median. Note: **skewness** often “follows the longer tail”.

Computational time

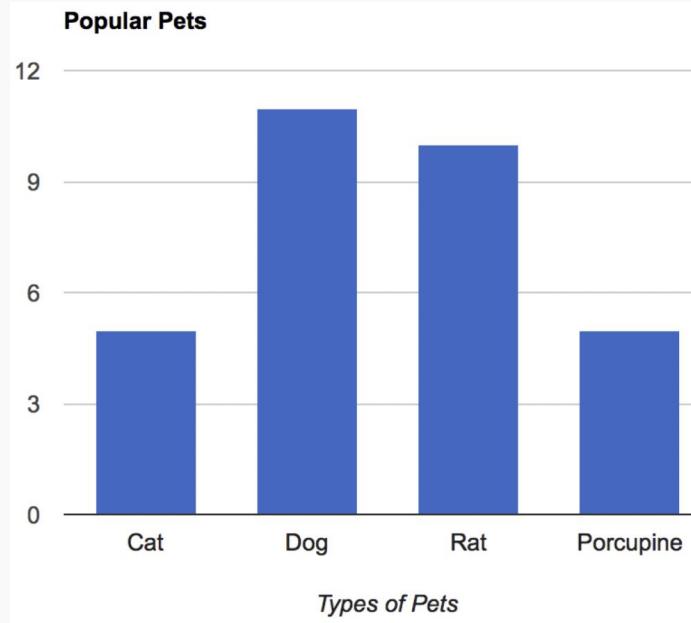
How hard (in terms of algorithmic complexity) is it to calculate

- the mean?
at most $O(n)$
- the median?
at most $O(n \log n)$ or possibly $O(n)$

Note: Practicality of implementation should be considered!

Regarding Categorical Variables...

For categorical variables, neither mean or median make sense.
Why?



The mode might be a better way to find the most
“representative” value.

Measures of Spread: Range

The spread of a sample of observations measures how well the mean or median describes the sample.

One way to measure spread of a sample of observations is via the **range**.

$$\text{Range} = \text{Maximum Value} - \text{Minimum Value}$$

Measures of Spread: Variance

The (sample) **variance**, denoted s^2 , measures how much on average the sample values deviate from the mean:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n |x_i - \bar{x}|^2$$

Note: the term $|x_i - \bar{x}|$ measures the amount by which each x_i deviates from the mean \bar{x} . Squaring these deviations means that s^2 is sensitive to extreme values (outliers).

Note: s^2 doesn't have the same units as the x_i :(
What does a variance of 1,008 mean? Or 0.0001?

Measures of Spread: Standard Deviation

The (sample) **standard deviation**, denoted s , is the square root of the variance

$$s = \sqrt{s^2} = \sqrt{\frac{1}{n-1} \sum_{i=1}^n |x_i - \bar{x}|^2}$$

Note: s does have the same units as the x_i . Phew!

Lecture Outline: Data, Summaries, and Visuals

- What are Data?
- **Exploratory Data Analysis (EDA)**
 - Descriptive Statistics
 - **Basic Visualizations**
- Historical Interlude
- Effective Visualizations

Reading: Ch. 1 in *An Introduction to Statistical Learning* (ISL)

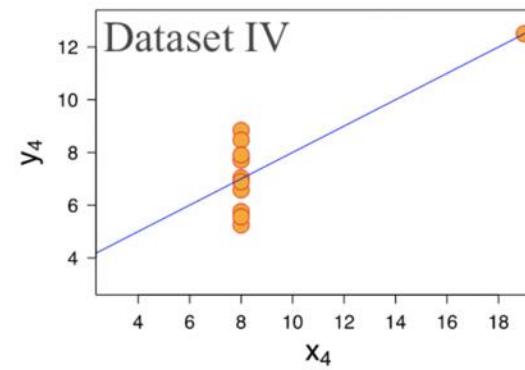
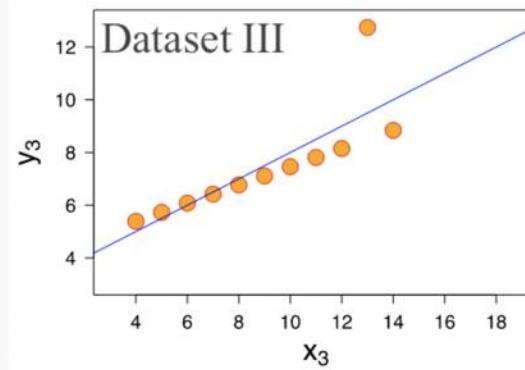
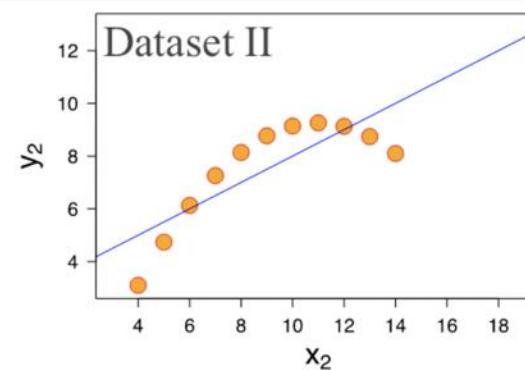
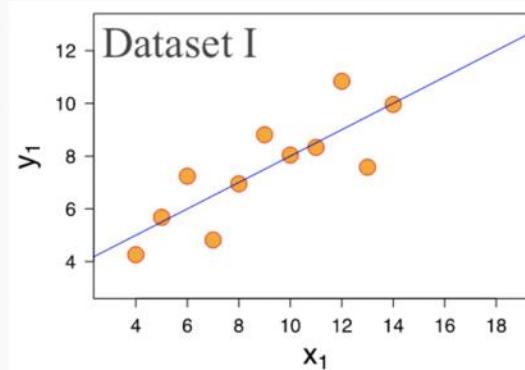
Anscombe's Data

The following four data sets comprise the Anscombe's Quartet; all four sets of data have identical simple summary statistics.

Dataset I		Dataset II		Dataset III		Dataset IV		
x	y	x	y	x	y	x	y	
10	8.04	10	9.14	10	7.46	8	6.58	
8	6.95	8	8.14	8	6.77	8	5.76	
13	7.58	13	8.74	13	12.74	8	7.71	
9	8.81	9	8.77	9	7.11	8	8.84	
11	8.33	11	9.26	11	7.81	8	8.47	
14	9.96	14	8.1	14	8.84	8	7.04	
6	7.24	6	6.13	6	6.08	8	5.25	
4	4.26	4	3.1	4	5.39	19	12.5	
12	10.84	12	9.13	12	8.15	8	5.56	
7	4.82	7	7.26	7	6.42	8	7.91	
5	5.68	5	4.74	5	5.73	8	6.89	
Sum:	99.00	82.51	99.00	82.51	99.00	82.51	99.00	82.51
Avg:	9.00	7.50	9.00	7.50	9.00	7.50	9.00	7.50
Std:	3.32	2.03	3.32	2.03	3.32	2.03	3.32	2.03

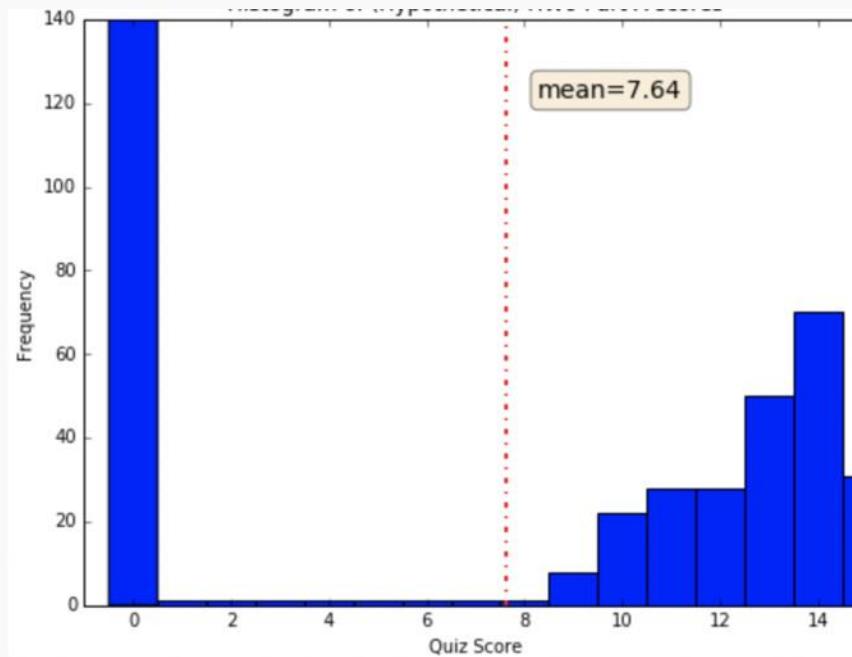
Anscombe's Data (cont.)

Summary statistics clearly don't tell the story of how they differ. But a picture can be worth a thousand words:



More Visualization Motivation

If I tell you that the average score for Homework 0 in my other class was: $7.64/15 = \underline{50.9\%}$, what does that suggest?



And what does the graph suggest?

More Visualization Motivation

Visualizations help us to analyze and explore the data. They help to:

- Identify hidden patterns and trends
- Formulate/test hypotheses
- Communicate any modeling results
 - Present information and ideas succinctly
 - Provide evidence and support
 - Influence and persuade
- Determine the next step in analysis/modeling

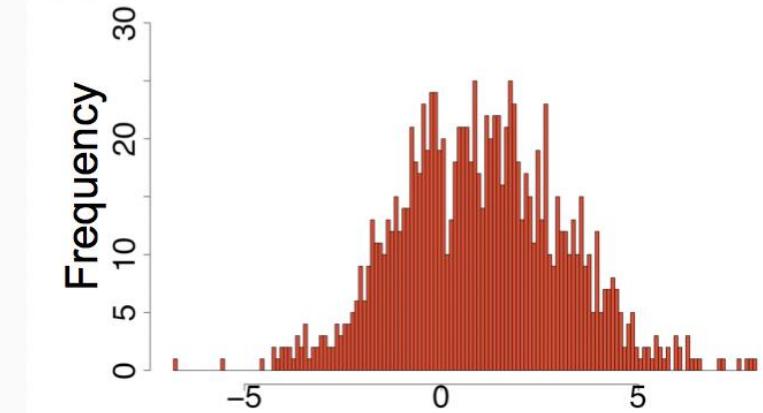
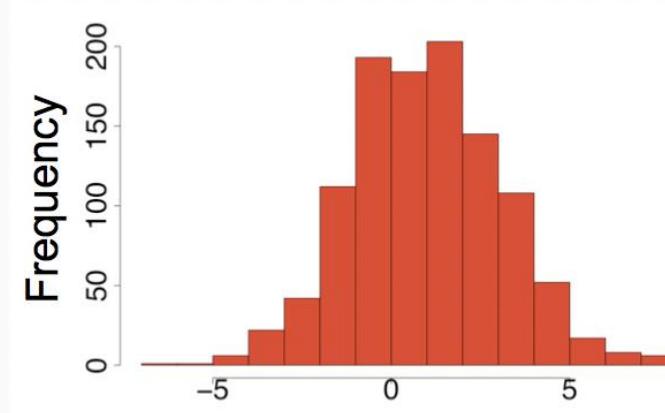
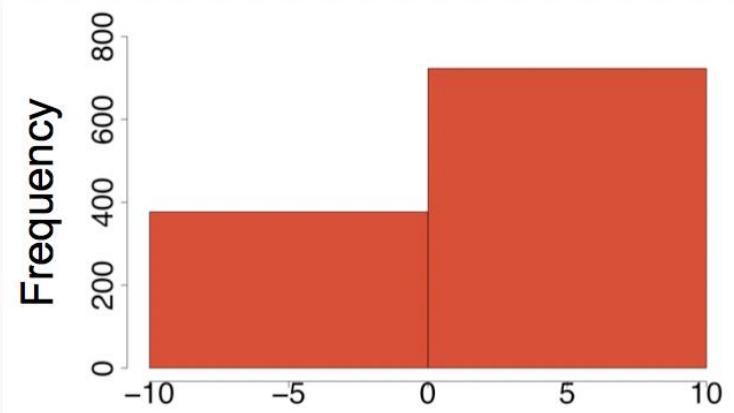
Types of Visualizations

What do you want your visualization to show about your data?

- **Distribution:** how a variable or variables in the dataset distribute over a range of possible values.
- **Relationship:** how the values of multiple variables in the dataset relate
- **Composition:** how the dataset breaks down into subgroups
- **Comparison:** how trends in multiple variable or datasets compare

Histograms to visualize distribution

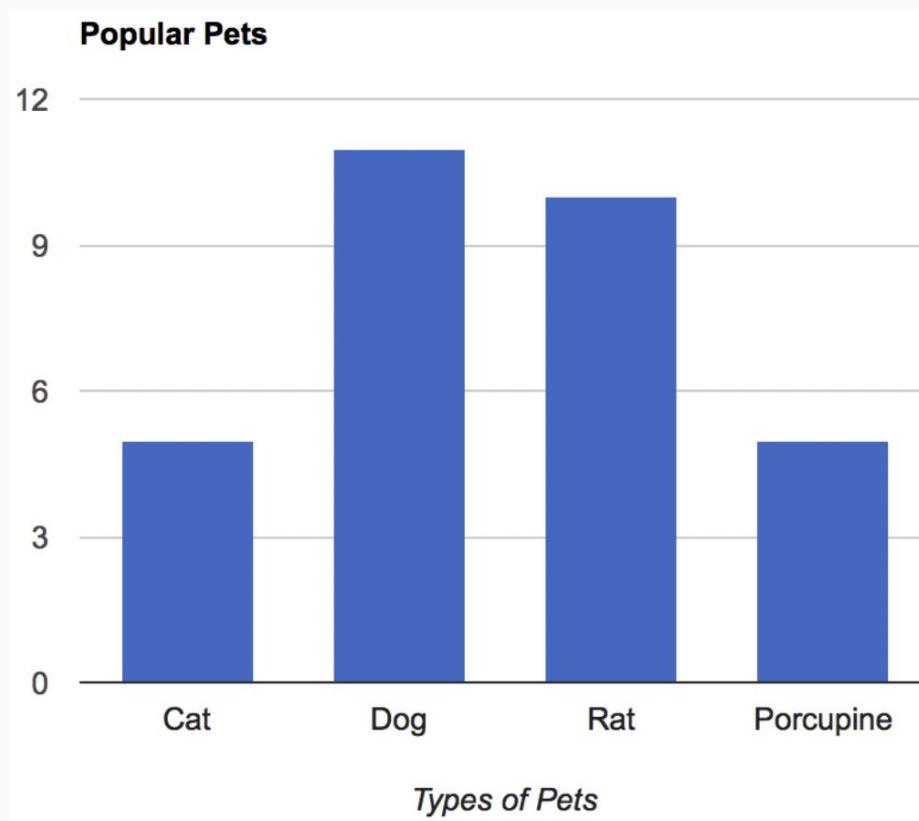
A **histogram** is a way to visualize how 1-dimensional data is distributed across certain values.



Note: Trends in histograms are sensitive to number of bins.

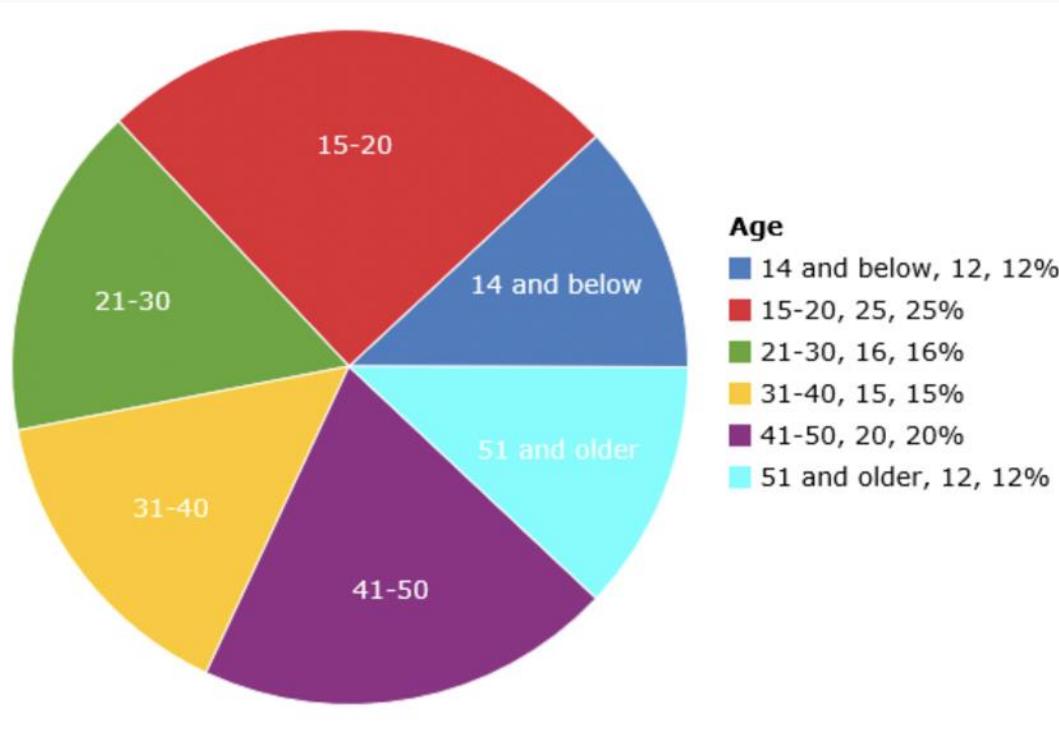
Bar plot for a categorical variable

A **bar plot** is a way to visualize the composition (aka, distribution) of a categorical variable.



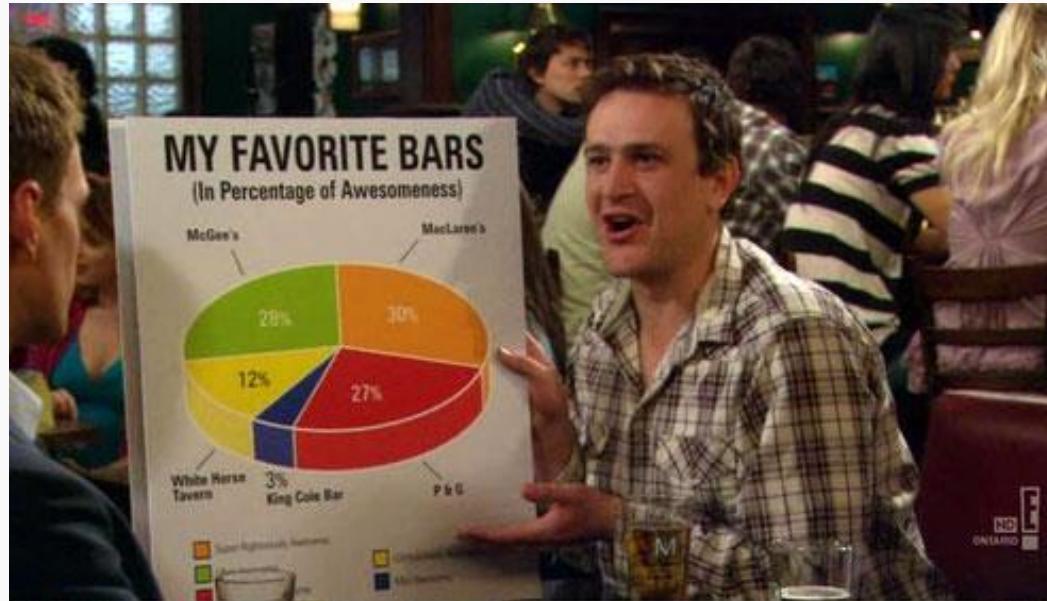
Pie chart for a categorical variable?

A **pie chart** is a way to visualize the static composition (aka, distribution) of a variable (or single group).



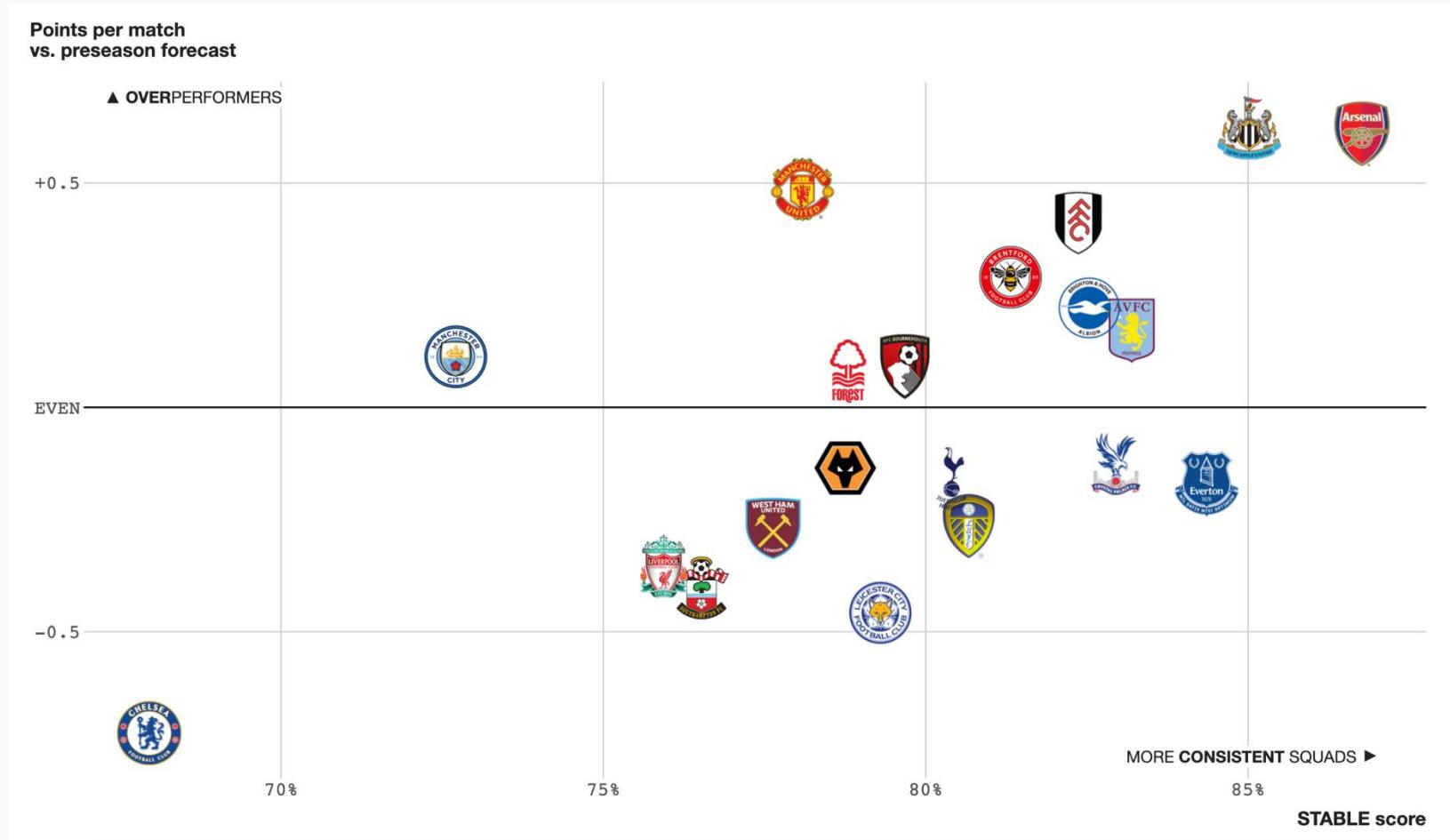
Pie charts are often frowned upon (and bar charts are used instead). Why?

Pies vs. Bars



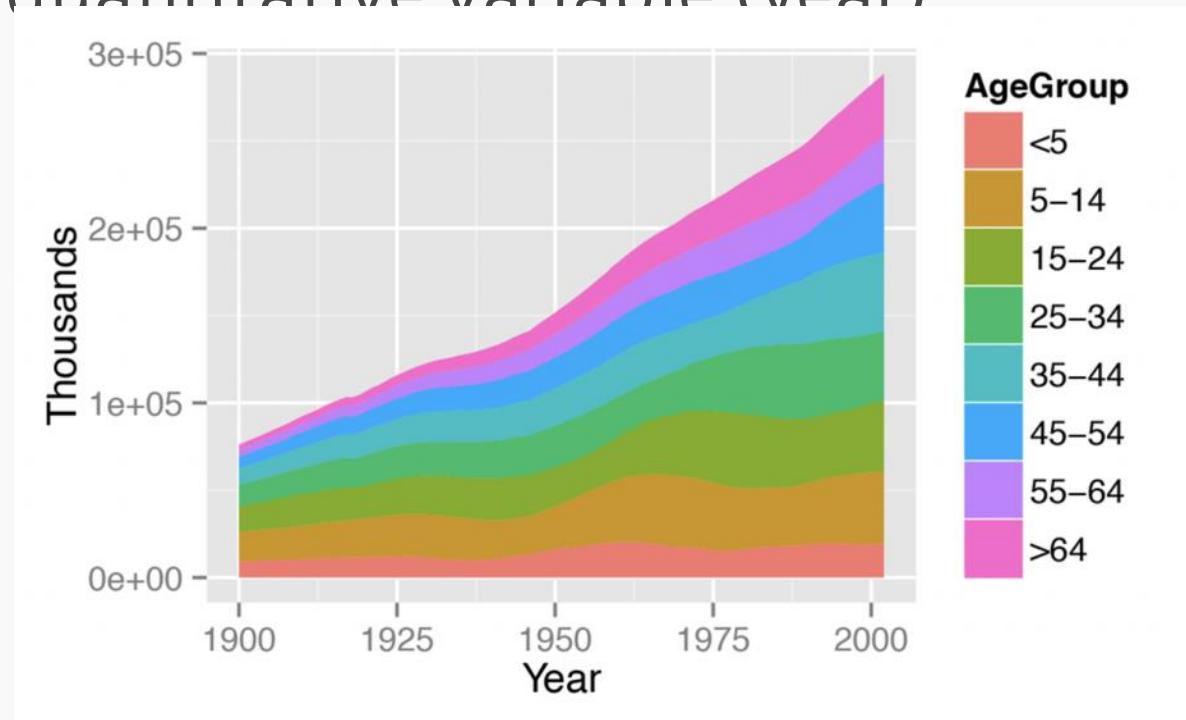
Scatter plots to visualize relationships

A **scatter plot** is a way to visualize the relationship between two different attributes of multi-dimensional data.



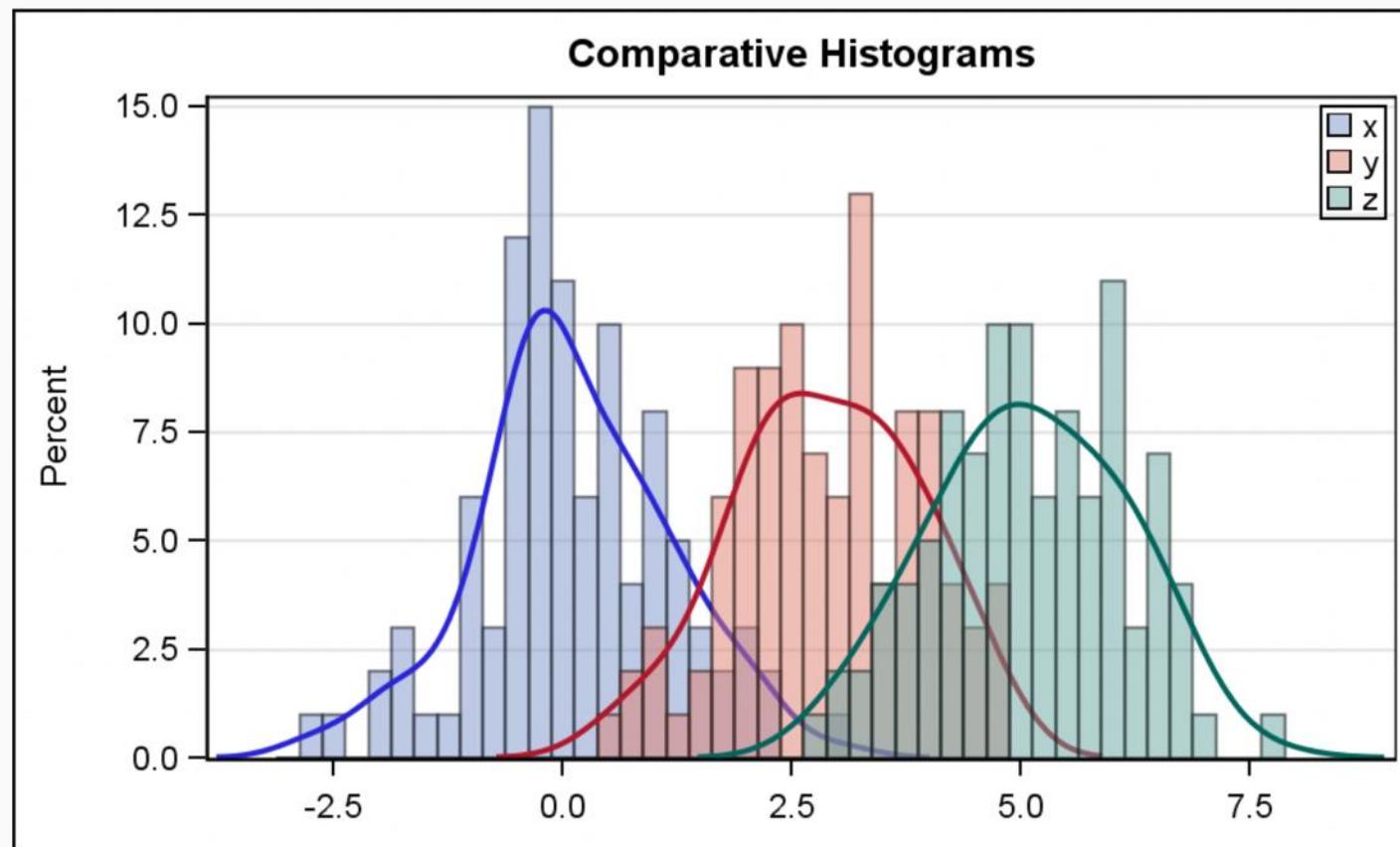
Stacked area graph to show trend over time

A **stacked area graph** is a way to visualize the composition of a group as it changes over time (or some other quantitative variable). This shows the relationship of a categorical variable (AgeGroup) to a quantitative variable (Year)



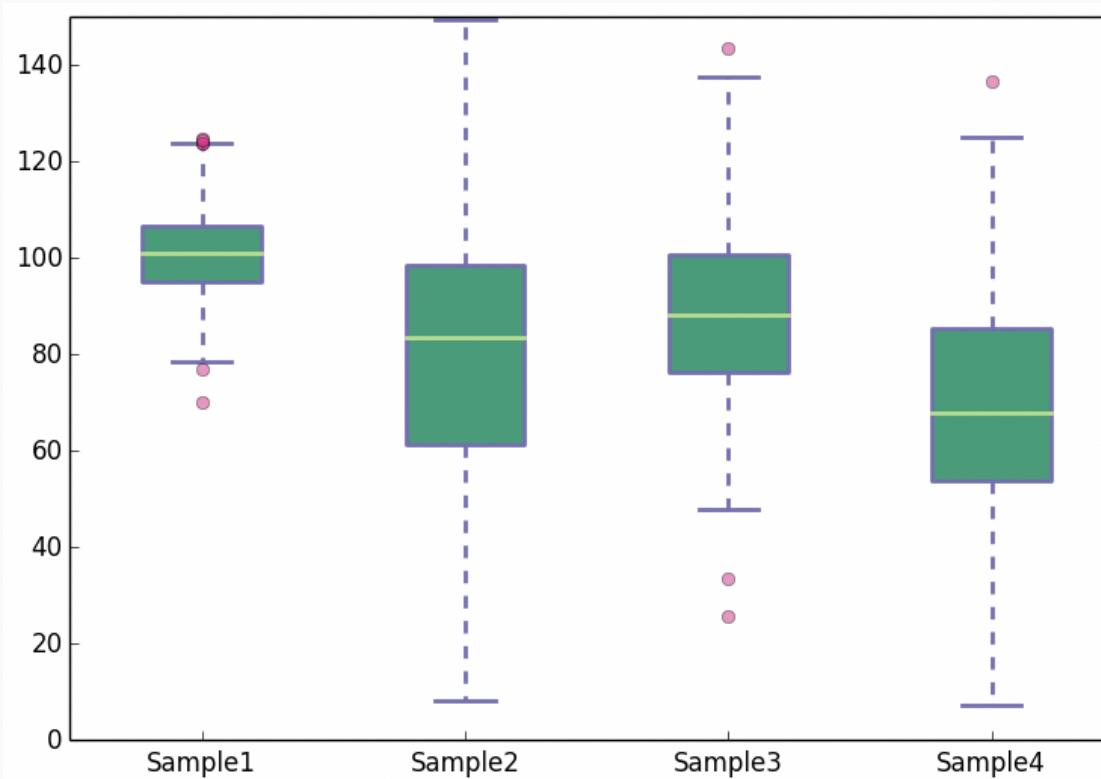
Multiple histograms

Plotting **multiple histograms** (and **kernel density estimates** of the distribution, here) on the same axes is a way to visualize how different variables compare (or how a variable differs over specific groups).



Boxplots

A **boxplot** is a simplified visualization to compare a quantitative variable across groups. It highlights the range, quartiles, median and any outliers present in a data set.



[Not] Anything is possible!

Often your dataset seem too complex to visualize:

- Data is too high dimensional (how do you plot 100 variables on the same set of axes?)
- Some variables are categorical (how do you plot values like Cat or No?)

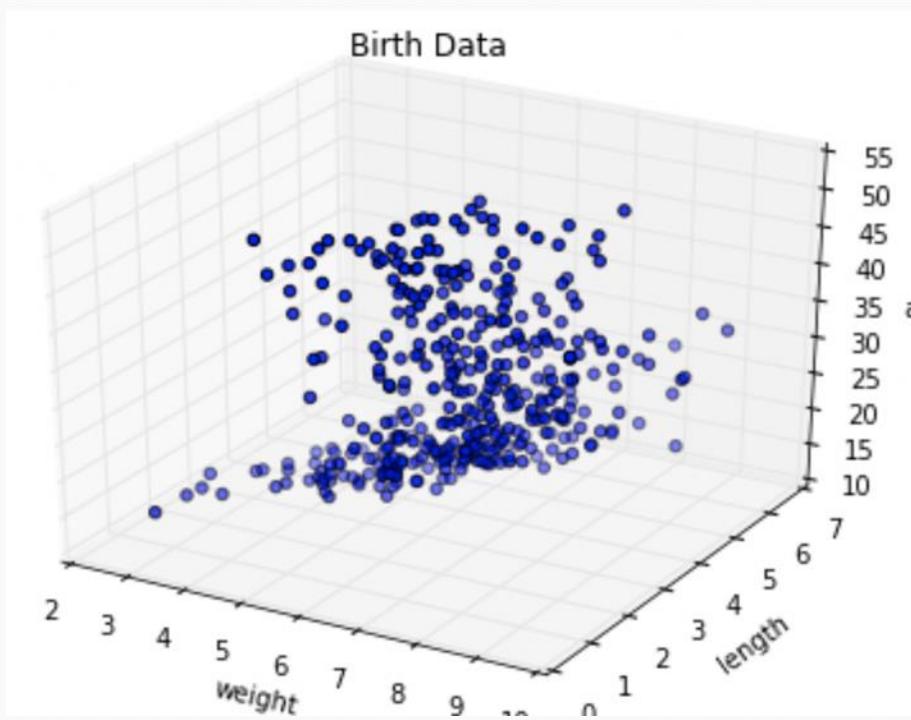


3+ variables

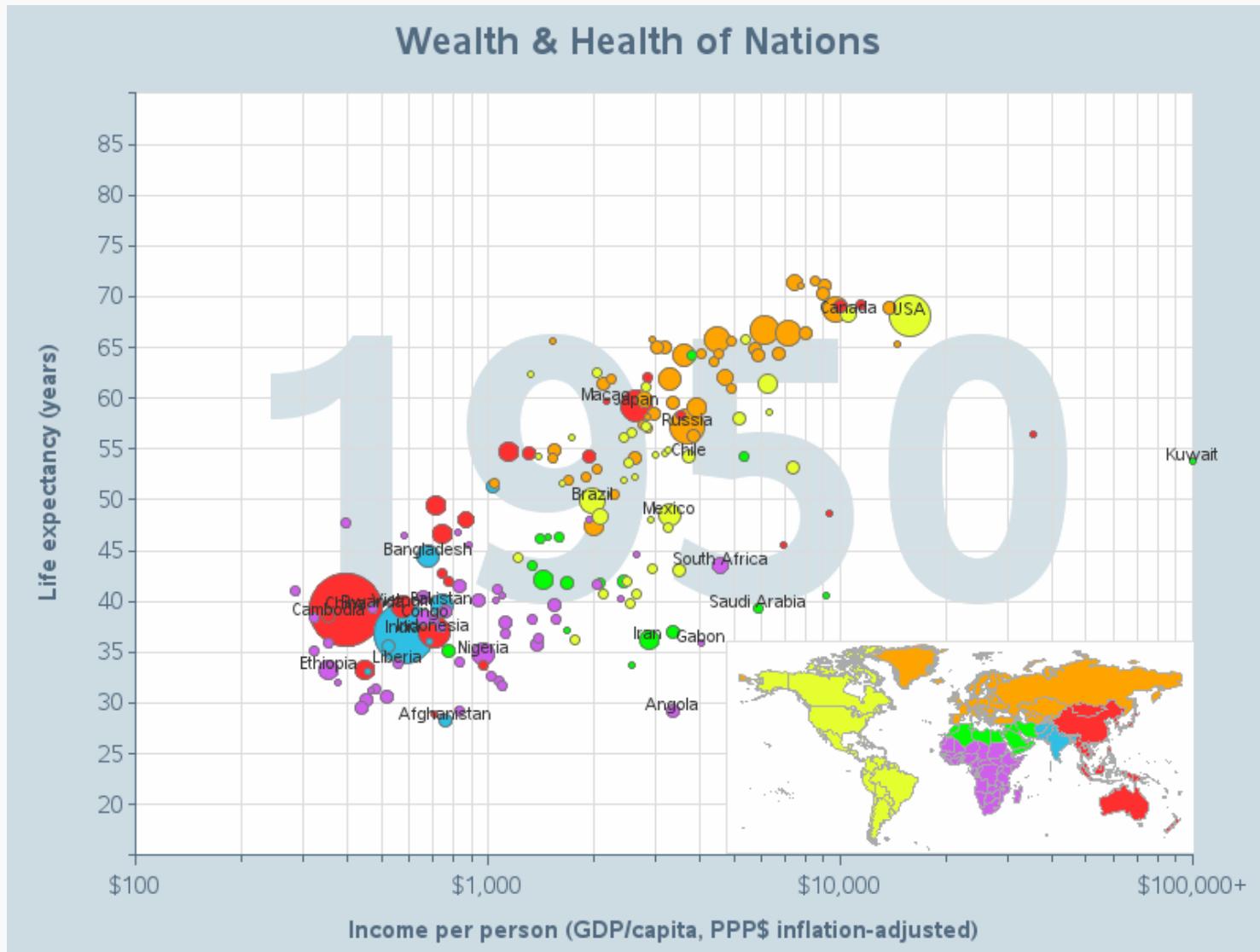
- Visualizing the distribution of 1 variable or the relationship between two variables is often straightforward.
- But how can we visualize how 3, 4, or more variables relate?
- It depends on the type of variables (categorical or numeric) that are involved.
- What is the best way to visualize relationships when...
 - all 3 are numeric?
 - 1 or more are categorical?

More dimensions not always better

When the data is high dimensional, a scatter plot of all data attributes can be impossible or unhelpful



Visualizing 3+ variables

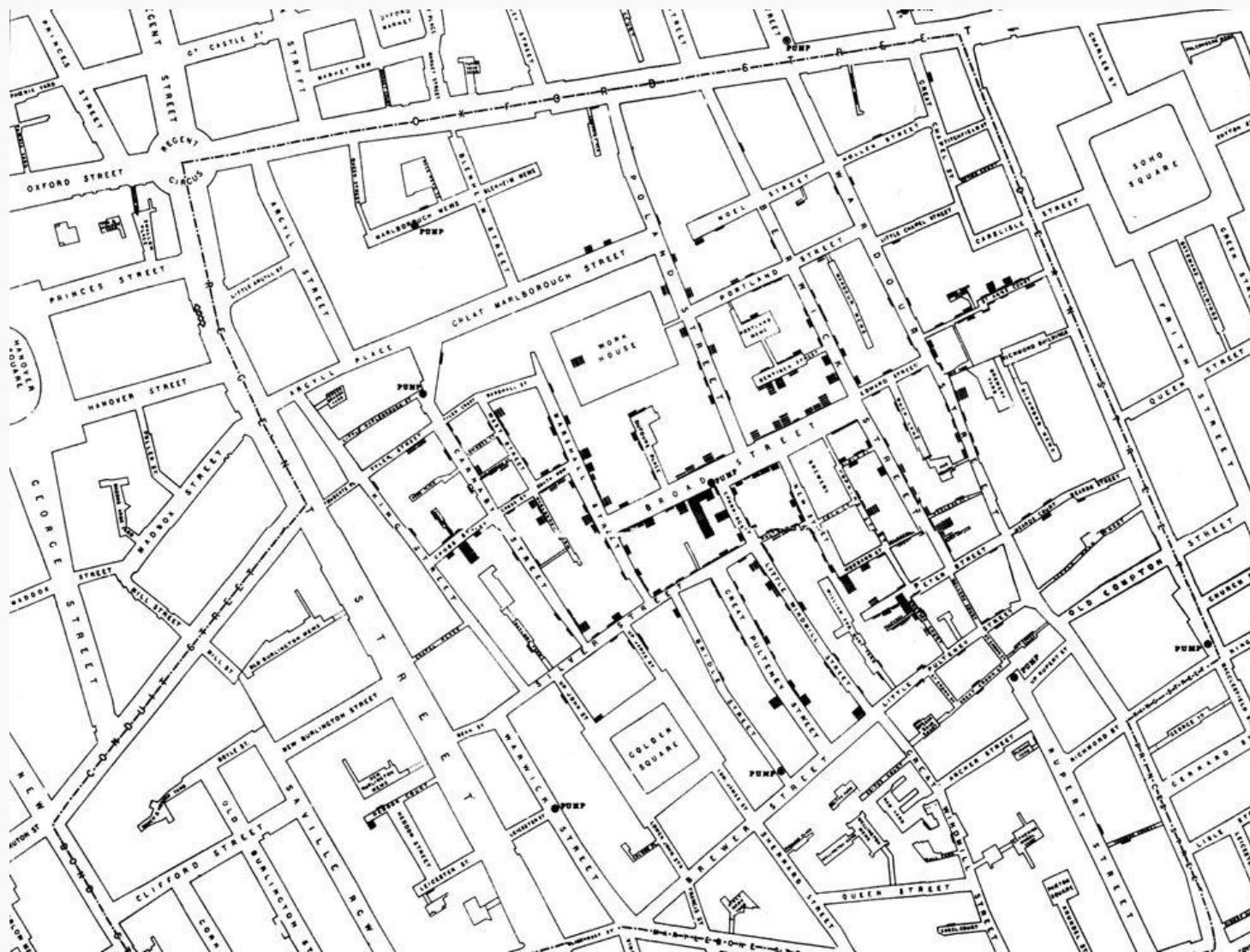


Lecture Outline: Data, Summaries, and Visuals

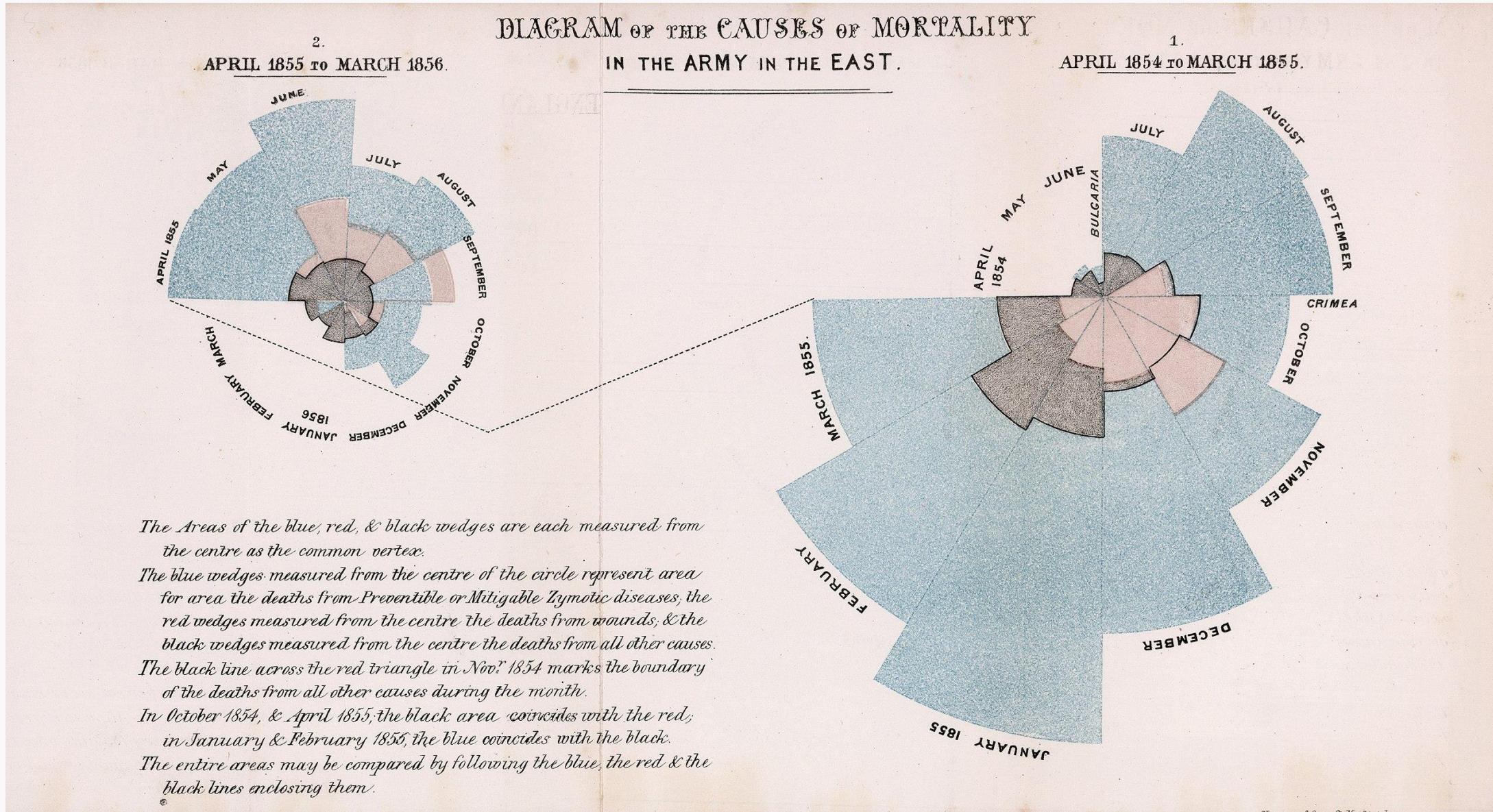
- What are Data?
- Exploratory Data Analysis (EDA)
 - Descriptive Statistics
 - Basic Visualizations
- Historical Interlude
- Effective Visualizations

Reading: Ch. 1 in *An Introduction to Statistical Learning* (ISL)

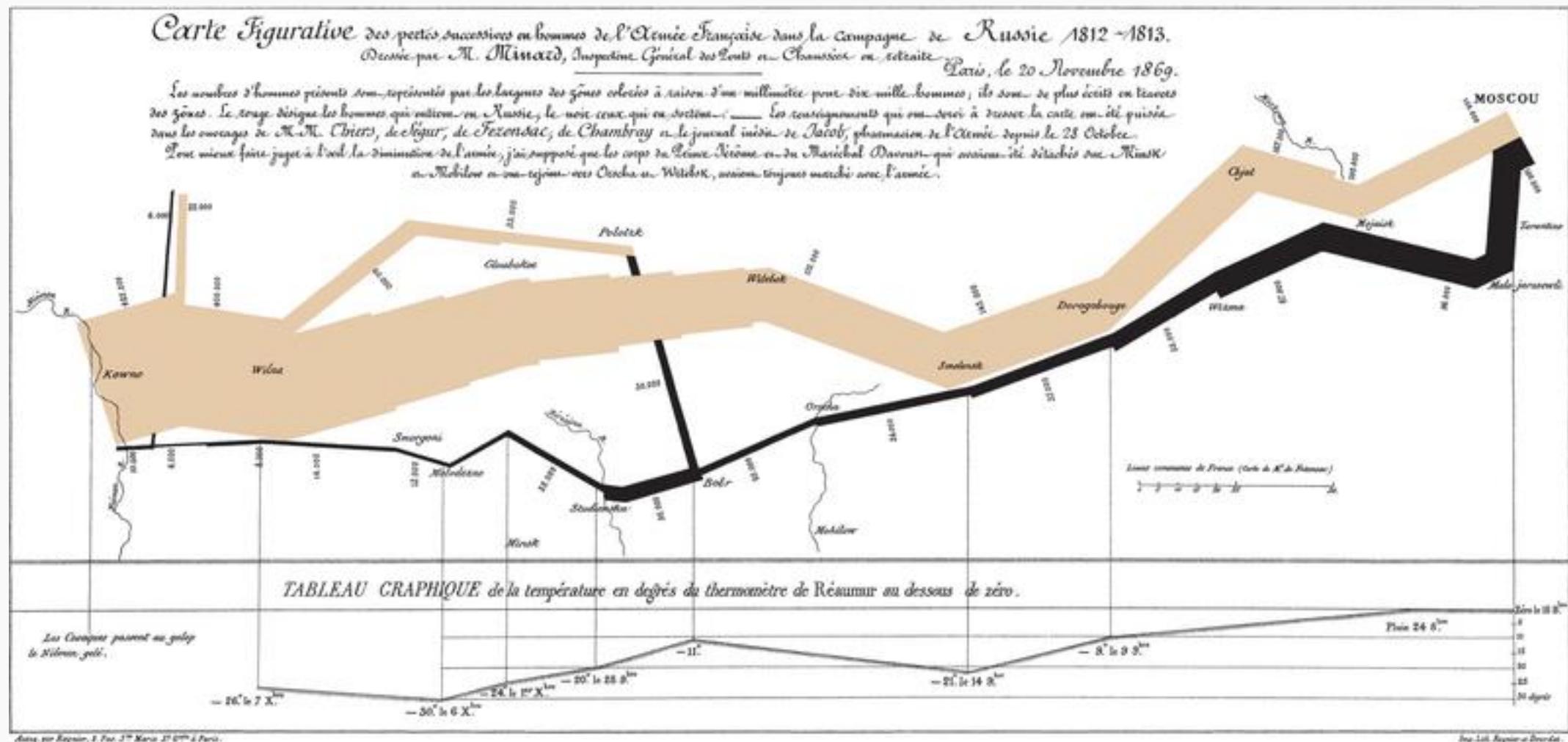
John Snow's Cholera Outbreak (1854)



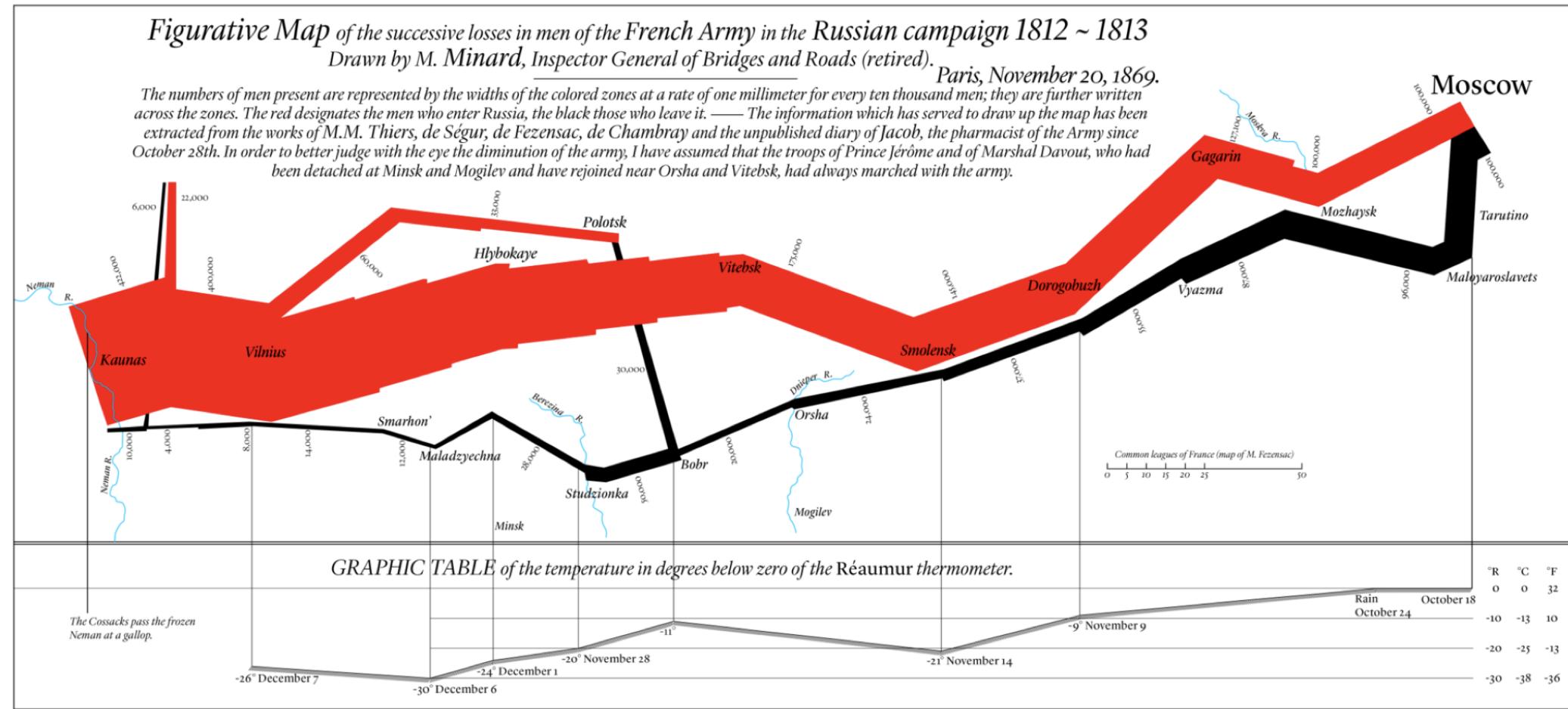
Florence Nightingale's Rose Chart of the Crimean War (1858)



Minard's Visual of Napoleon's March through Russia (1869)



Minard's Visual of Napoleon's March through Russia (1869)



Lecture Outline: Data, Summaries, and Visuals

- What are Data?
- Exploratory Data Analysis (EDA)
 - Descriptive Statistics
 - Basic Visualizations
- Historical Interlude
- **Effective Visualizations**

Reading: Ch. 1 in *An Introduction to Statistical Learning* (ISL)

Pointers for Effective Visualizations

- Have graphical integrity
- Keep it simple
- Use the right display
- Use color strategically
- Know your audience

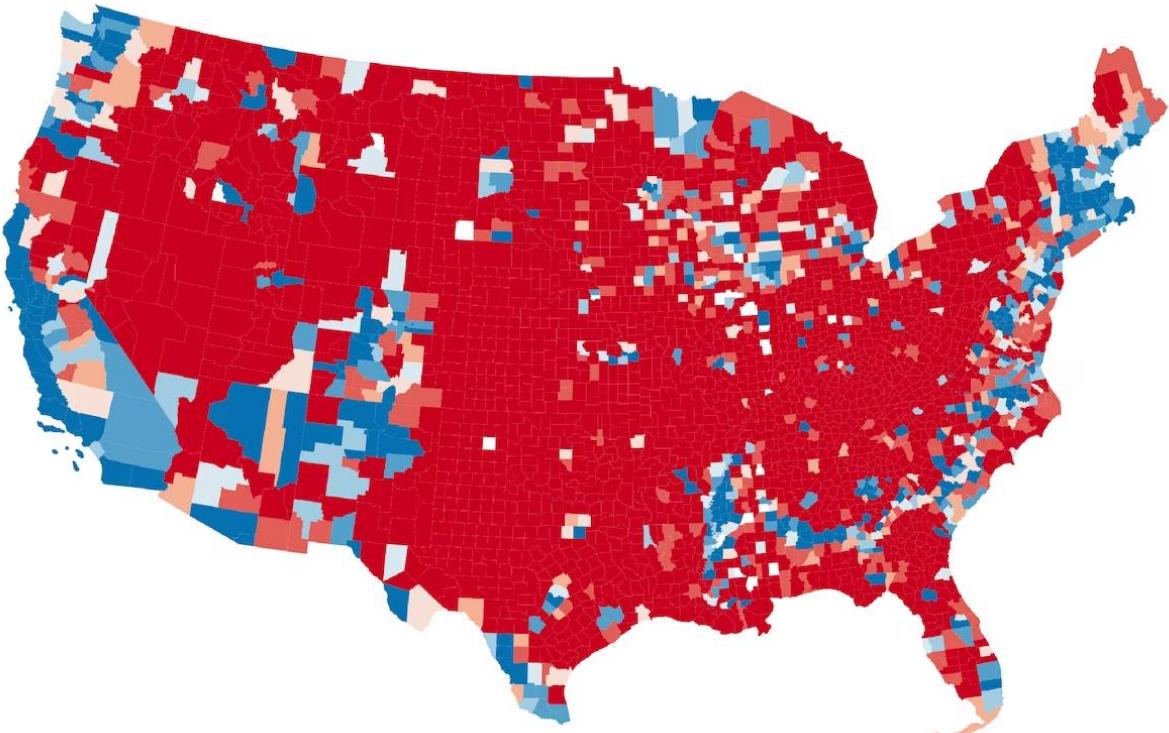


annnnnd
**WHY SHOULD
I CARE?**

Graphical Integrity

What does this map suggest about the 2020 election?

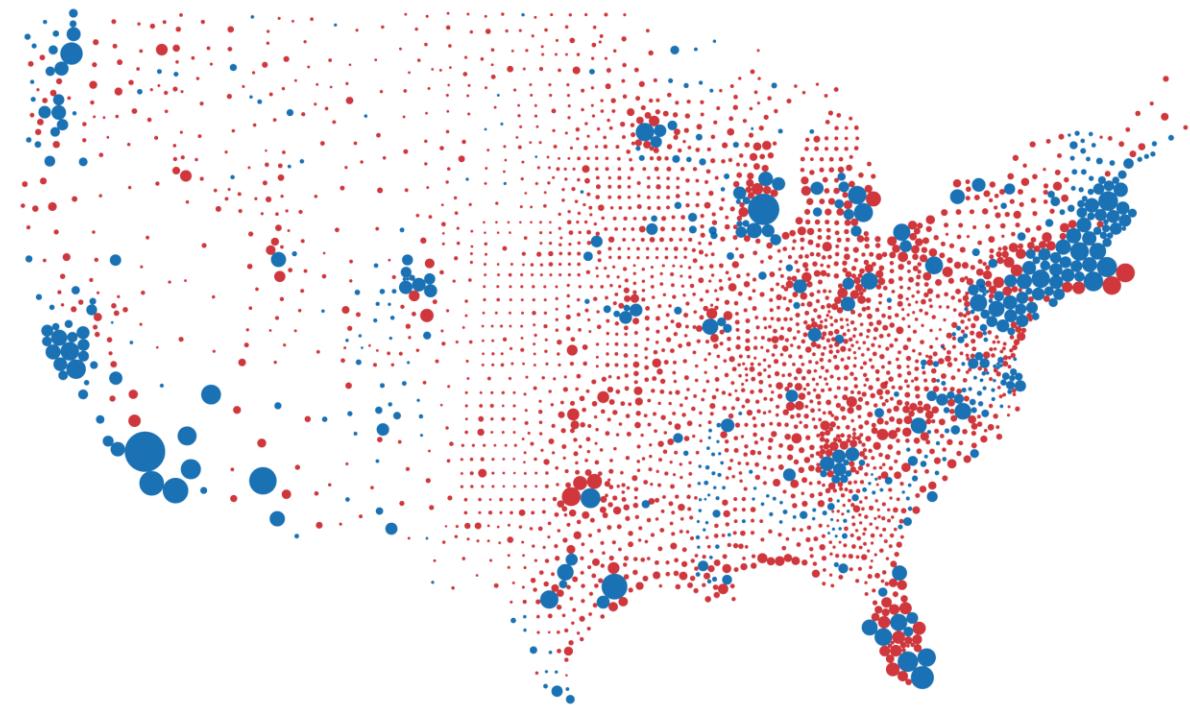
Preliminary 2020 presidential results by county



Source: Preliminary Edison Research results

THE WASHINGTON POST

Preliminary 2020 presidential results by county

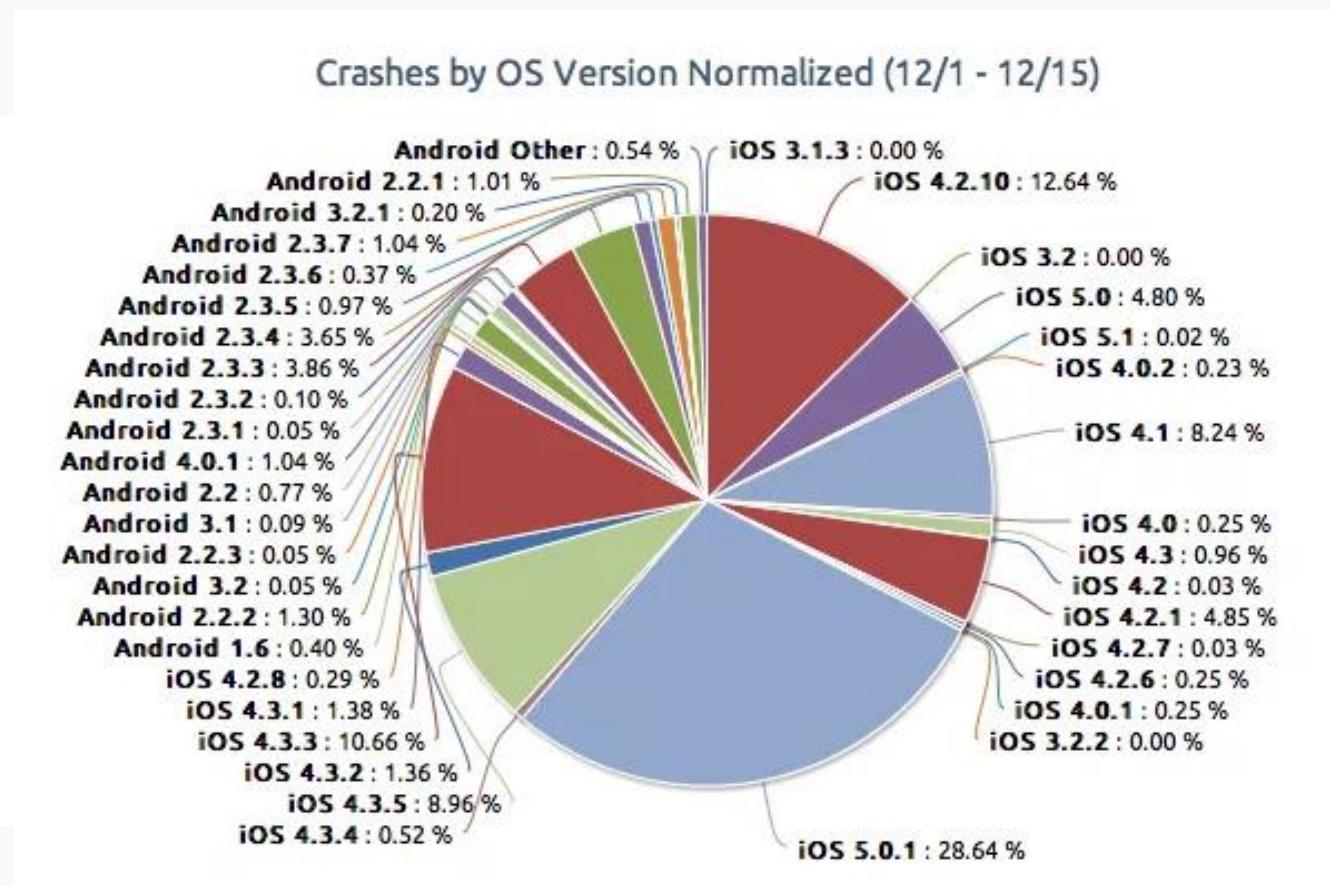
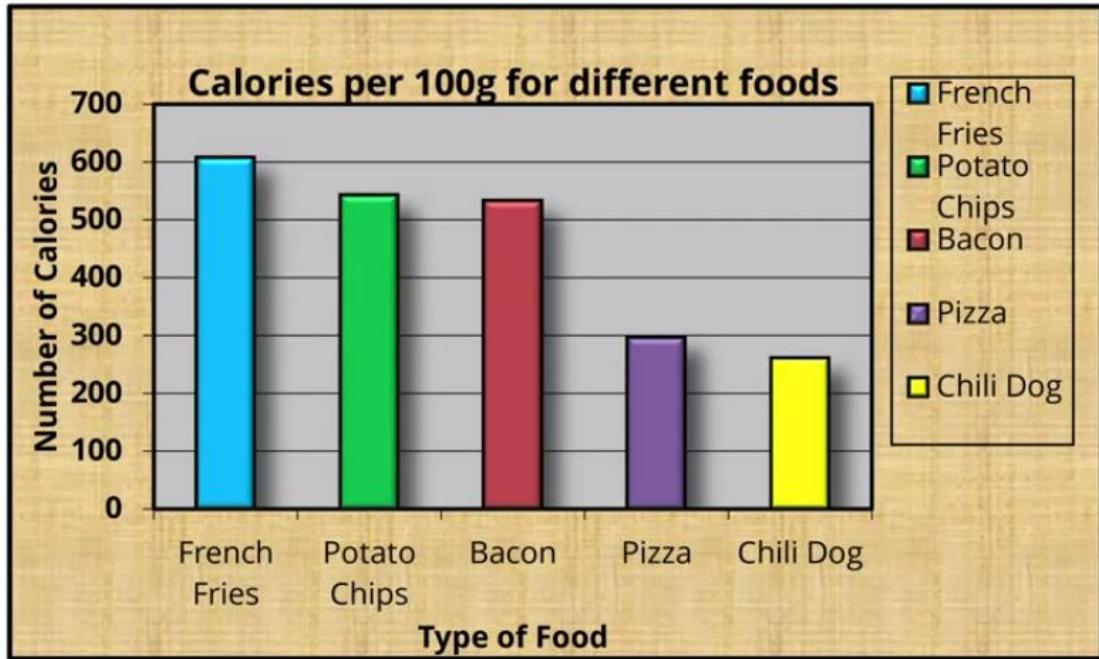


Source: Preliminary Edison Research results, turnout estimates

THE WASHINGTON POST

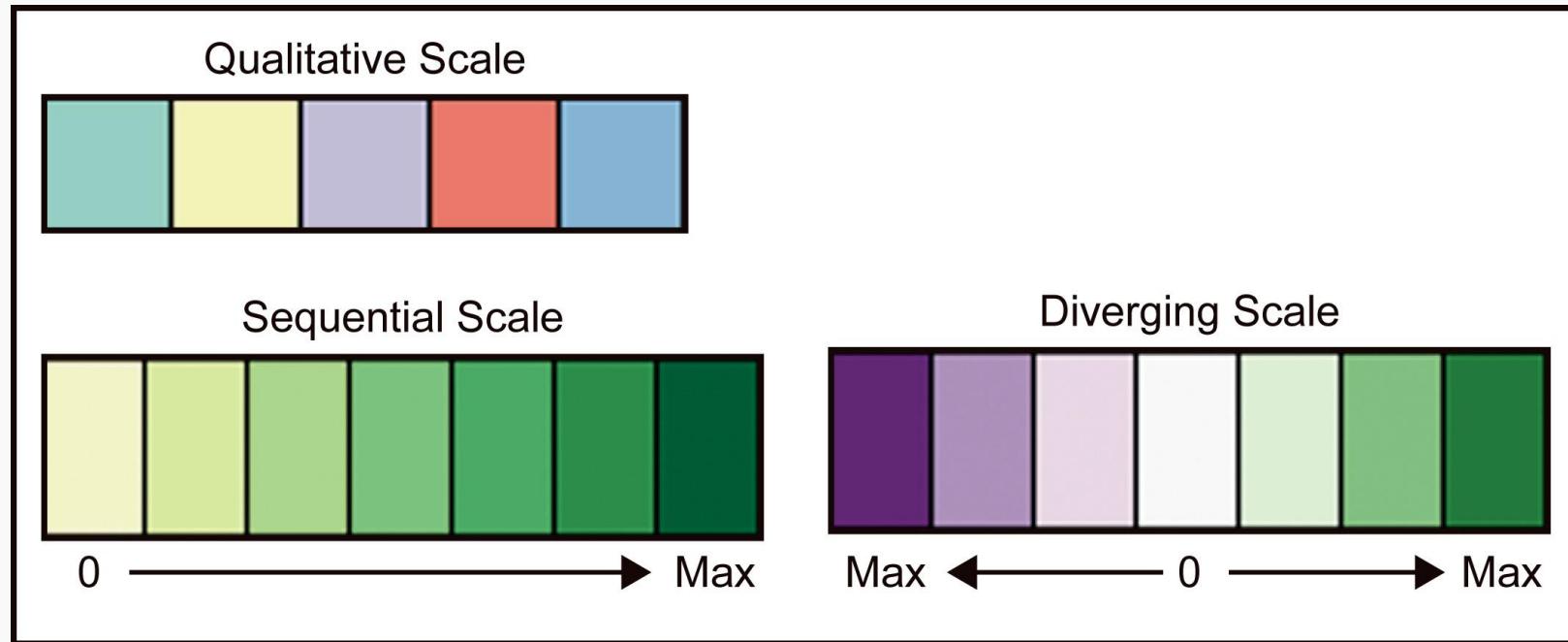
Keep it simple

Avoid Chart Junk



Use color strategically

Type of scale matters:



What are example variables for each?

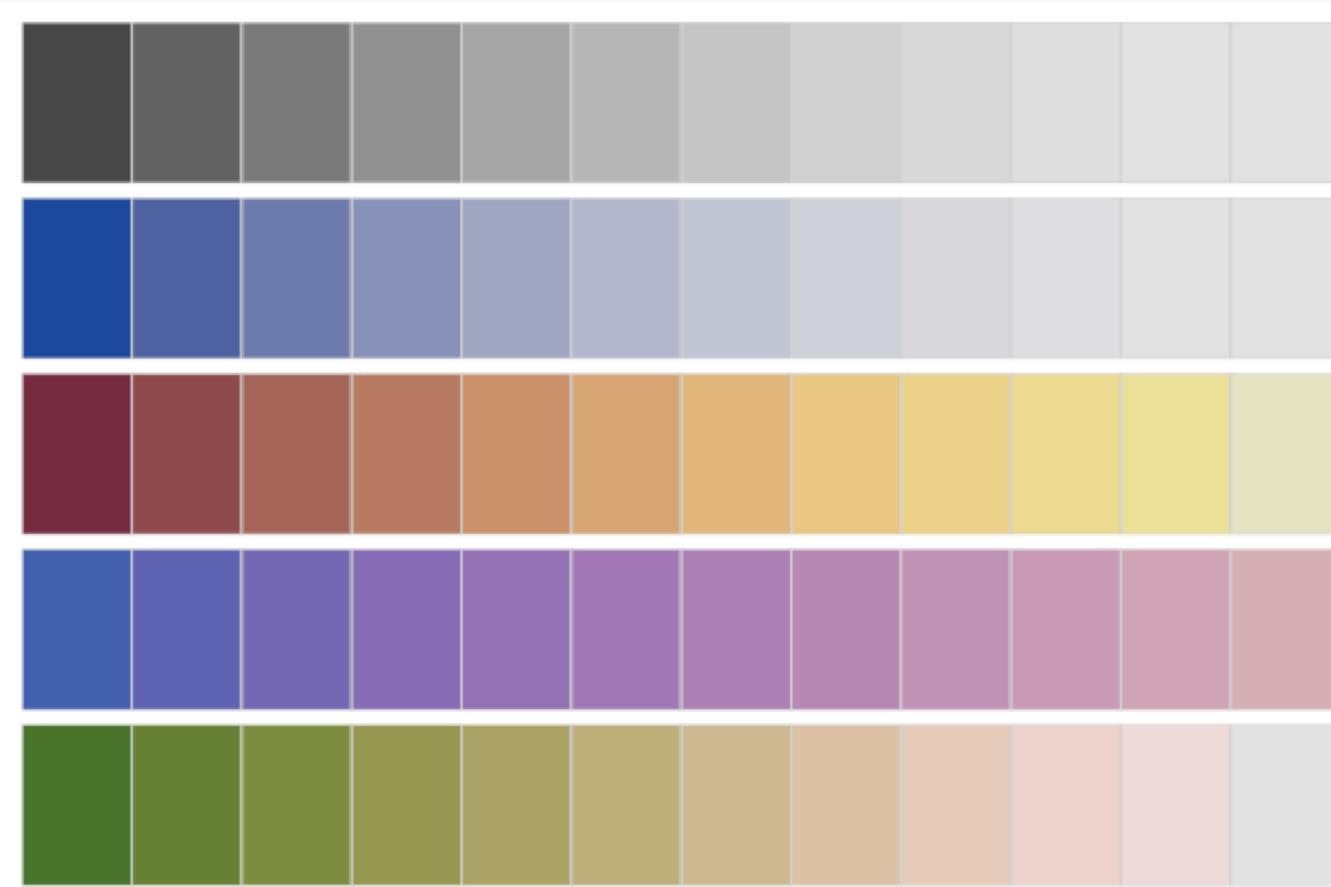
Use color strategically

Colors for categories (Do not use more than 5-8 colors at once)



Use color strategically

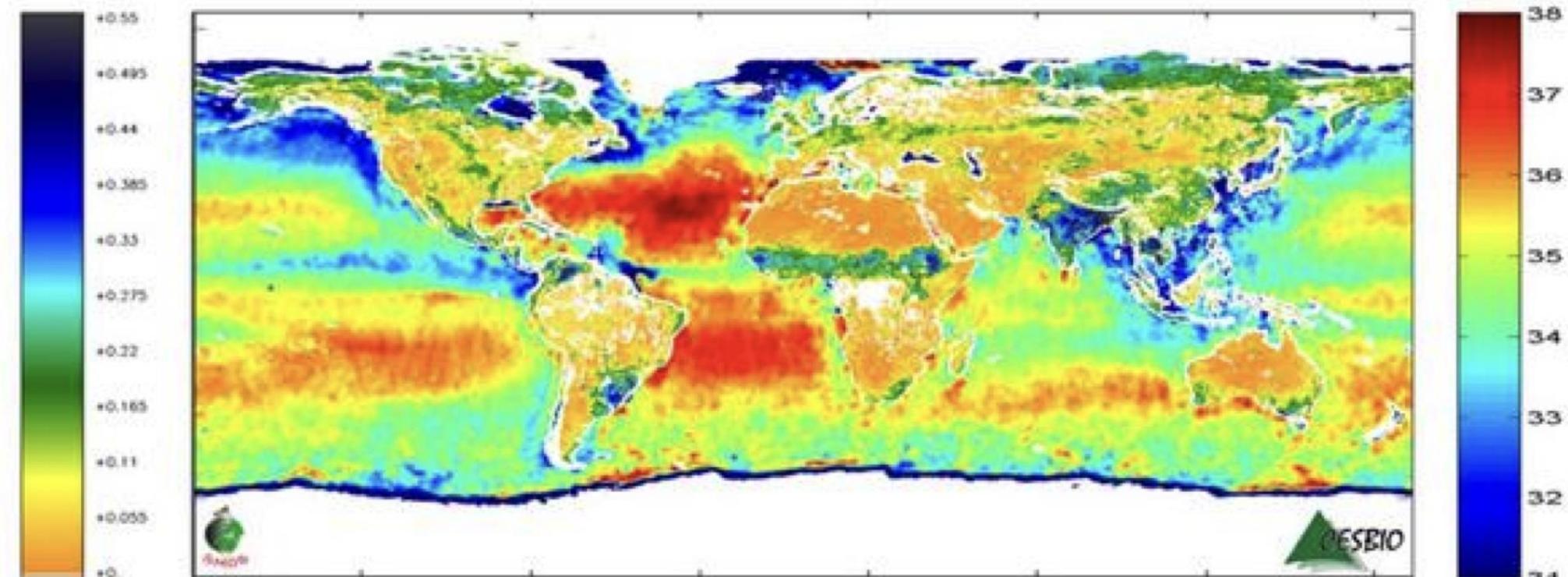
Colors for Ordinal Data -- Vary luminance and saturation



Zelis et al, 2009, "Escaping RGBland: Selecting Colors for Statistical Graphics"

Use color strategically

Beware of the Rainbow Colormap (Keep it simple)



Use color strategically

Be cognizant of color blindness



Protanope

Deuteranope

Tritanope

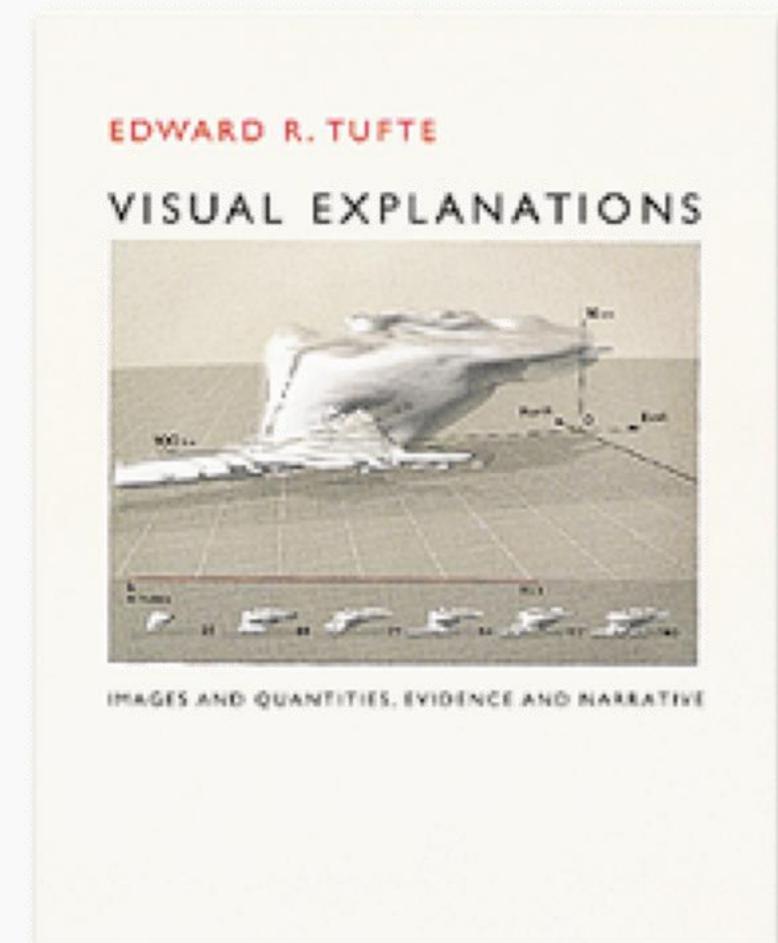
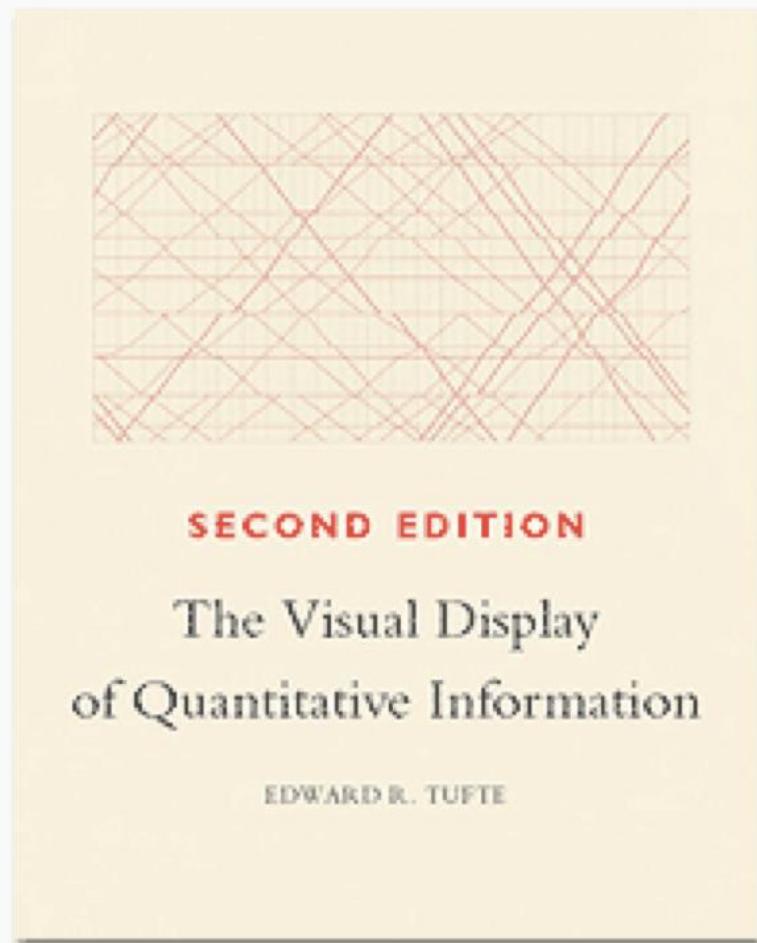
Red / green
deficiencies

Blue / Yellow
deficiency

Know your audience

- What do they know?
- What motivates them? What do they desire?
- What experiences do you share? What are common goals?
- What insights can you provide? What tools and “magical gifts”?

Edward Tufte



Edward Tufte's Principles of Graphical Excellence

Graphical excellence ...

- is the well-designed presentation of interesting data—a matter of substance, of statistics, and of design.
- consists of complex ideas communicated with clarity, precision and efficiency.
- is that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space.
- is nearly always multivariate.
- requires telling the truth.

Lecture #1: Introduction to CS1090A

aka STAT109A, AC209A, CSCIE-109A

CS109A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai and Chris Gumb



Lecture Outline

- What is data science?
- Why data science?
- How to learn and why take CS109A?
- What is this class: who, how, what?
- Demo

Lecture Outline

- **What is data science?**
- Why data science?
- How to learn and why take CS109A?
- What is this class: who, how, what?
- Demo

A little bit of history

History: The Evolution of Data Science: Early Methods

In ancient times, scientific knowledge was largely based on empirical observations. People would gather data through direct experience, such as counting stars in the sky or measuring crop yields.



History: The Evolution of Data Science: Early Methods

In ancient times, scientific knowledge was largely based on empirical observations. People would gather data through direct experience, such as counting stars in the sky or measuring crop yields.



The Evolution of Data Science: From Observation to Innovation

Thousands of years ago, science was primarily empirical in nature. Individuals would observe and count entities like stars and crops. This collected data was then used to construct devices that helped explain these phenomena.



The Evolution of Data Science: The Age of Equations

A few centuries ago, the approach to science shifted significantly. Researchers began using mathematical equations, often in the form of differential equations, to describe relationships and phenomena.

$$F = G \frac{m_1 m_2}{d^2}$$

$$i\hbar \frac{\partial}{\partial t} - \Psi = \hat{H}\Psi$$

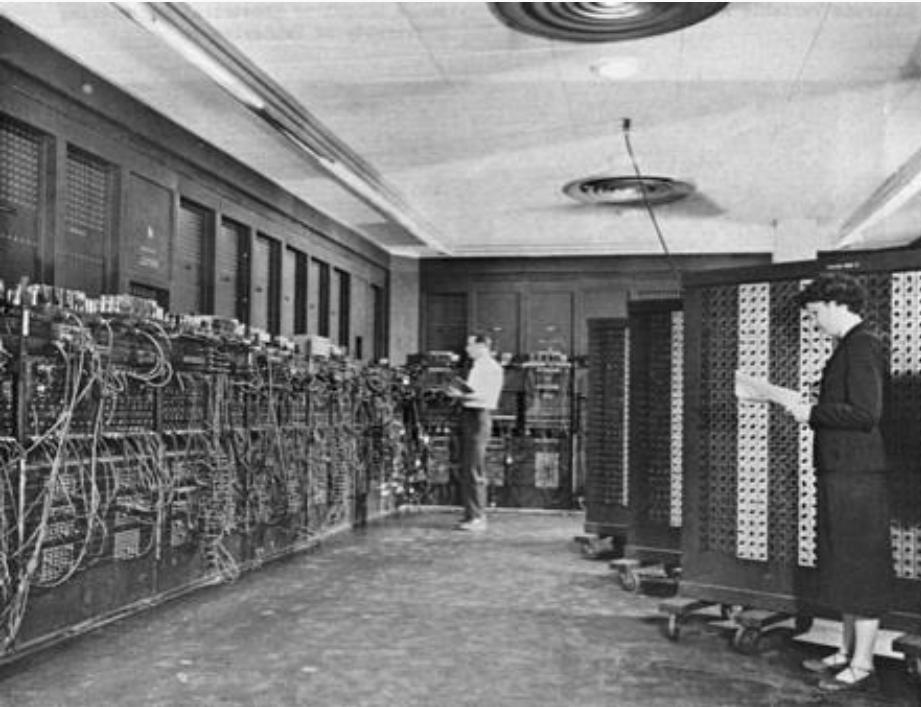
$$\begin{aligned}\nabla \cdot E &= 0 & \nabla \times E &= -\frac{1}{c} \frac{\partial H}{\partial t} \\ \nabla \cdot H &= 0 & \nabla \times H &= \frac{1}{c} \frac{\partial E}{\partial t}\end{aligned}$$

$$E = mc^2$$

$$\rho \left(\frac{\partial v}{\partial t} + v \cdot \nabla v \right) = -\nabla p + \nabla \cdot T + f$$

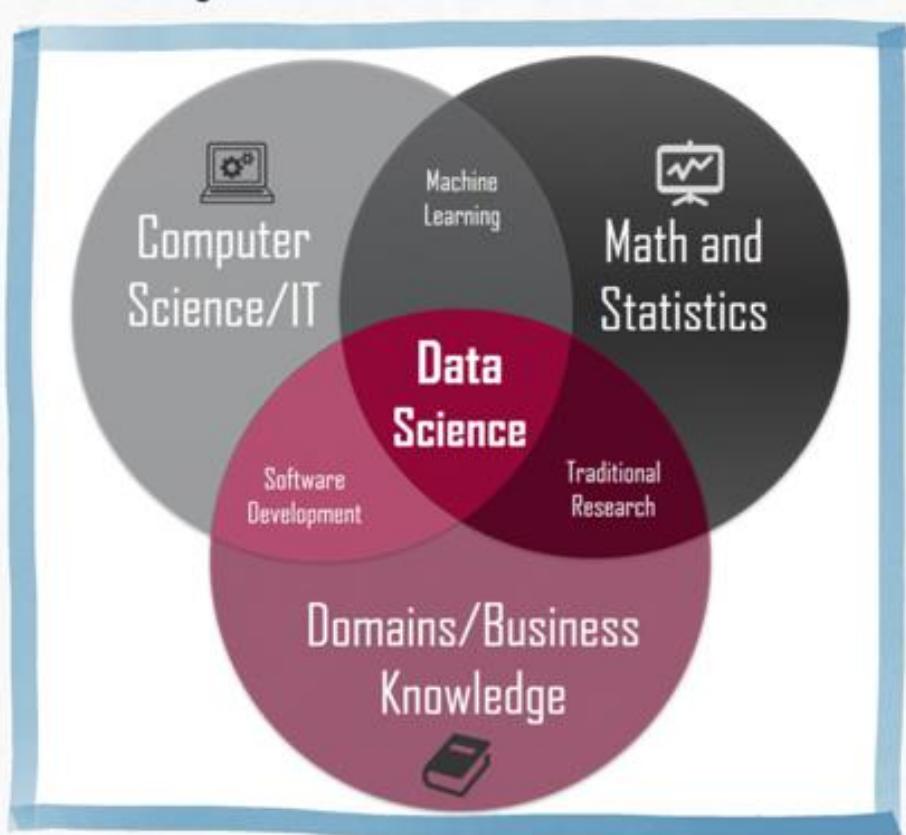
The Evolution of Data Science: The Computational Era

Approximately a century ago, another paradigm shift occurred in science with the emergence of computational approaches. This allowed for complex simulations and analyses that were previously unimaginable.



The Rise of Data Science and Machine Learning

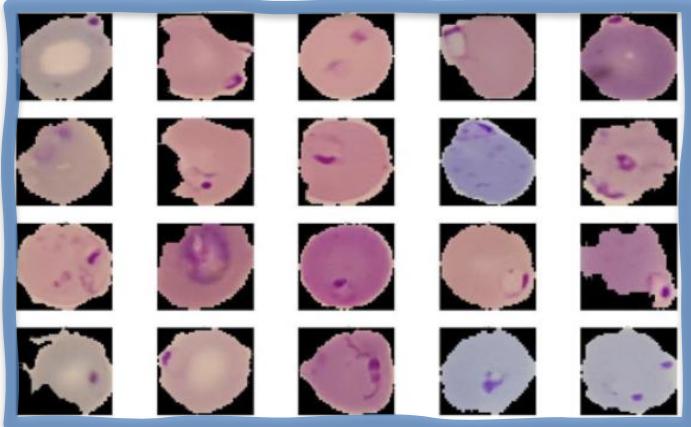
In more recent times, the focus has shifted yet again to data science and machine learning. These disciplines specialize in extracting patterns and insights from large sets of data, revolutionizing how we understand and interact with the world.



- Interdisciplinary
- Data and task focused
- Resource aware
- Adaptable to changes in the environment and needs

The Potential of Data Science

Disease Diagnosis



Detecting malaria from blood smears

Drug Discovery



Discovering new drug combinations
using language models

Generative AI

Greek
American
male bald
professor
with
glasses
and beard



Creating images from text prompts

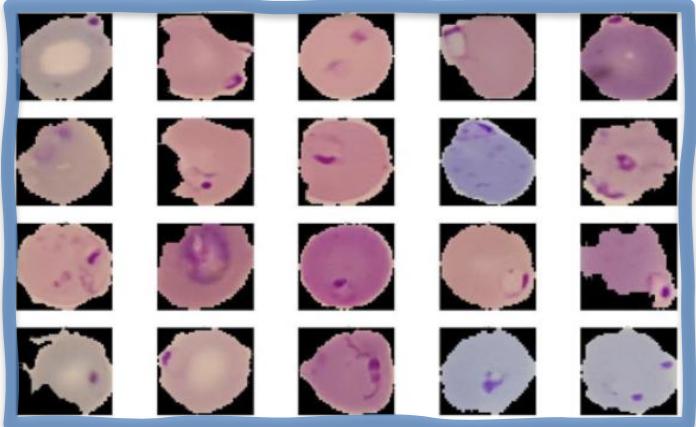
Transportation



Self driving trucks for safe night shipping

The Potential of Data Science

Disease Diagnosis



Detecting malaria from blood smears

Drug Discovery



Discovering new drug combinations
using language models

Generative AI



Creating images from text prompts

Transportation



Self driving trucks for safe night shipping

The Potential of Data Science

Gender Bias



Some DS models for evaluating job applications in some fields show bias in favor of male candidates

Racial Bias



Risk models used in US courts have shown to be biased against non-white defendants

What?

The Data Science Process

Ask an interesting question

Get the Data

Explore the Data

Model the Data

Communicate/Visualize the Results

What?

The Data Science Process

Ask an interesting question

Get the Data

Explore the Data

Model the Data

Communicate/Visualize the Results

What is the scientific goal?

What do you want to predict or estimate?

What would you do if you had **all** of the
data?

What?

The Data Science Process

Ask an interesting question

Get the Data

Explore the Data

Model the Data

Communicate/Visualize the Results

How were the data sampled?

Which data are relevant?

Are there privacy issues?

What?

The Data Science Process

Ask an interesting question

Get the Data

Explore the Data

Model the Data

Communicate/Visualize the Results

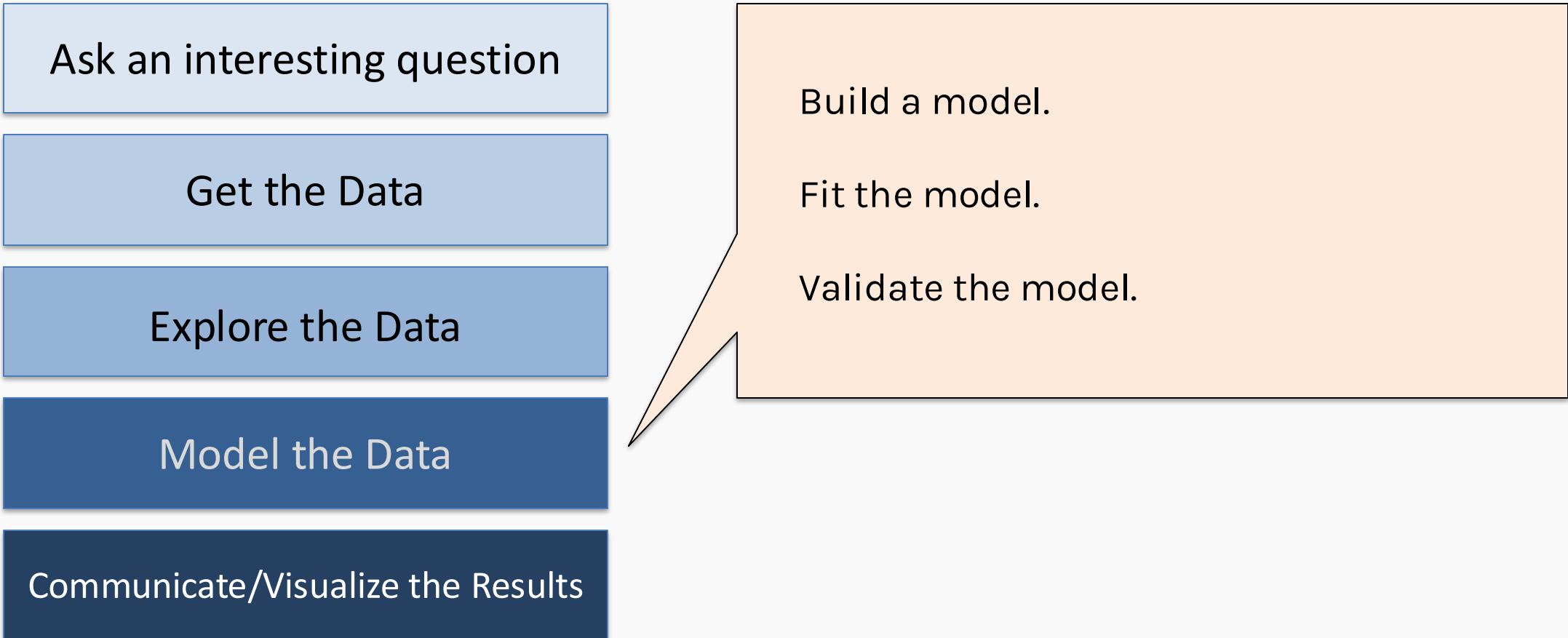
Plot the data.

Are there anomalies or egregious issues?

Are there patterns?

What?

The Data Science Process



What?

The Data Science Process

Ask an interesting question

Get the Data

Explore the Data

Model the Data

Communicate/Visualize the Results

What did we learn?

Do the results make sense?

Can we effectively tell a story?

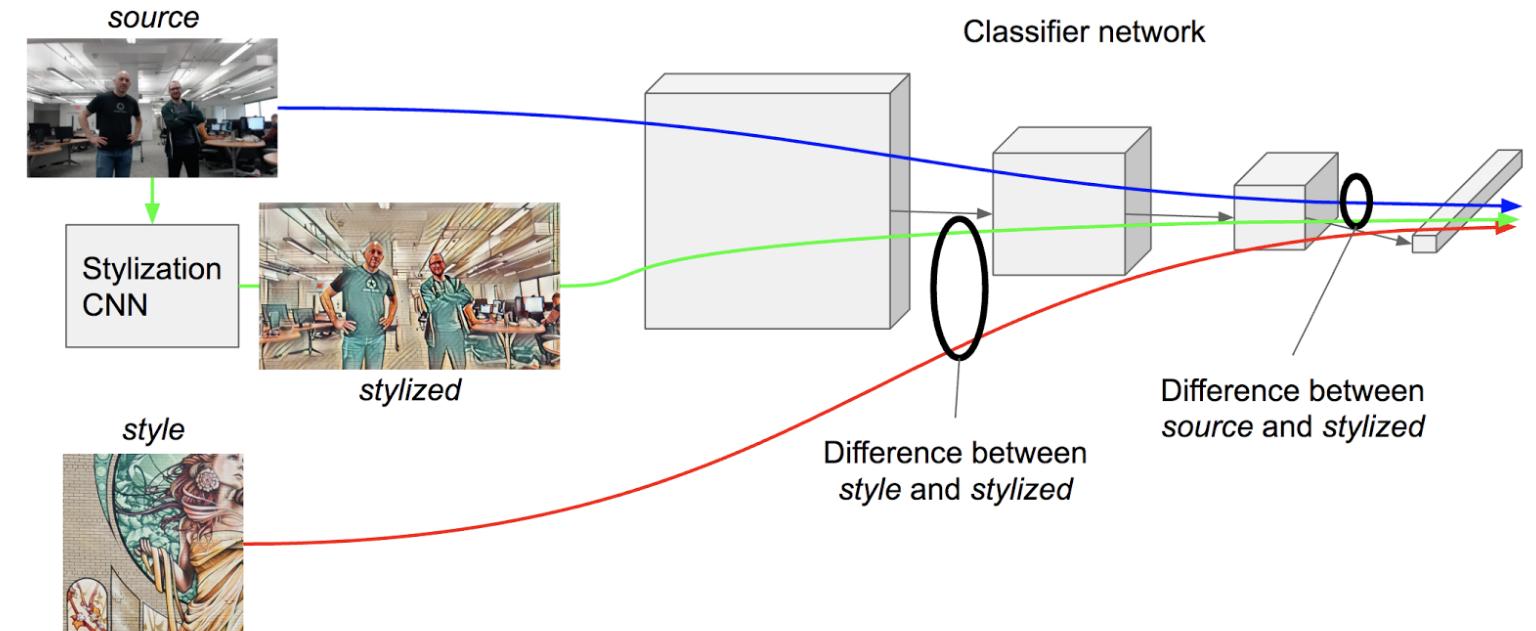
Lecture Outline

- What is data science?
- **Why data science?**
- How to learn and why CS109A?
- What is this class: who, how, what?
- Demo





Minimise Loss



But if you decide to do it...

- It's a lot of fun!
- You will be on the cutting edge of research and industry
- You'll make lots of money doing something you'll enjoy
- It's not that hard to start and do!





50 Best Jobs in America for 2022

Job Title	Median Base Salary	Job Satisfaction	Job Openings	
#1 Enterprise Architect	\$144,997	4.1/5	14,021	View Jobs
#2 Full Stack Engineer	\$101,794	4.3/5	11,252	View Jobs
#3 Data Scientist	\$120,000	4.1/5	10,071	View Jobs
#4 Devops Engineer	\$120,095	4.2/5	8,548	View Jobs
#5 Strategy Manager	\$140,000	4.2/5	6,977	View Jobs
#6 Machine Learning Engineer	\$130,489	4.3/5	6,801	View Jobs

Why?

Jobs!

50 Best Jobs in America

This report ranks jobs according to each job's Glassdoor Job Score, determined by combining three factors: number of job openings, salary, and overall employee satisfaction rating.

Employers: Want to recruit better in 2017? [Get started](#) [about how.](#)

United States | 2017

12k Shares | [f](#) [t](#) [in](#) [e](#)

1 Data Scientist



4.8 / 5
Job Score

\$110,000
Median Base Salary

4.4 / 5
Job Satisfaction

4,184
Job Openings

[View Jobs](#)

2 DevOps Engineer



A large red arrow points from the top right towards the median base salary figure for Data Scientists (\$110,000).

I want to do it because

Lecture #22: Generative Model

CS109B, STAT109B, AC209B, CSCIE-109B

CS109B Introduction to Data Science

Pavlos Protopapas, Alex Young



Lecture #22: Generative Model

CS109B, STAT109B, AC209B, CSCIE-109B

CS109B Introduction to Data Science

Pavlos Protopapas, Alex Young

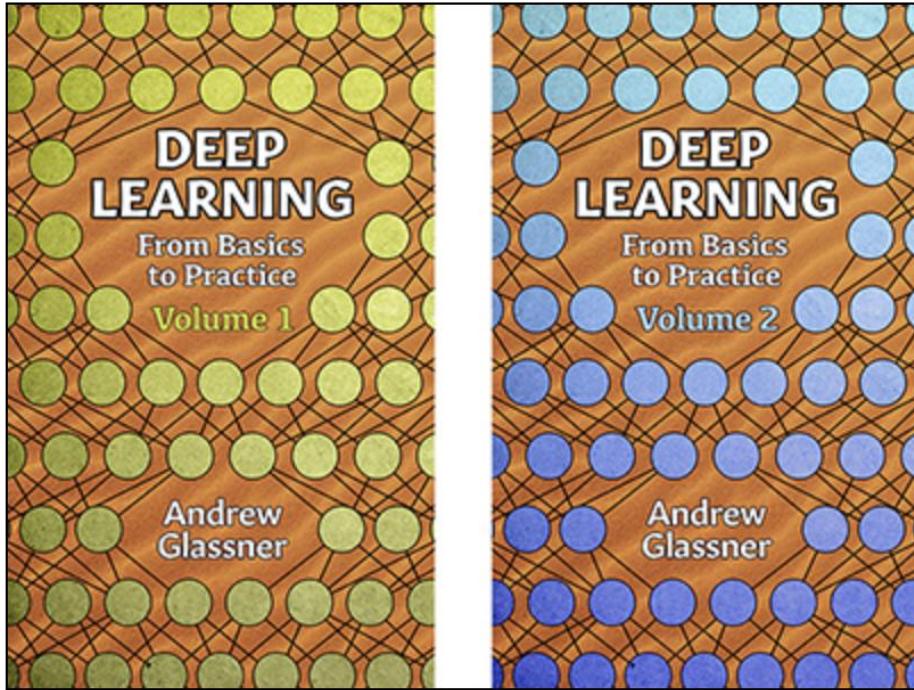


Pavlos Protopapas

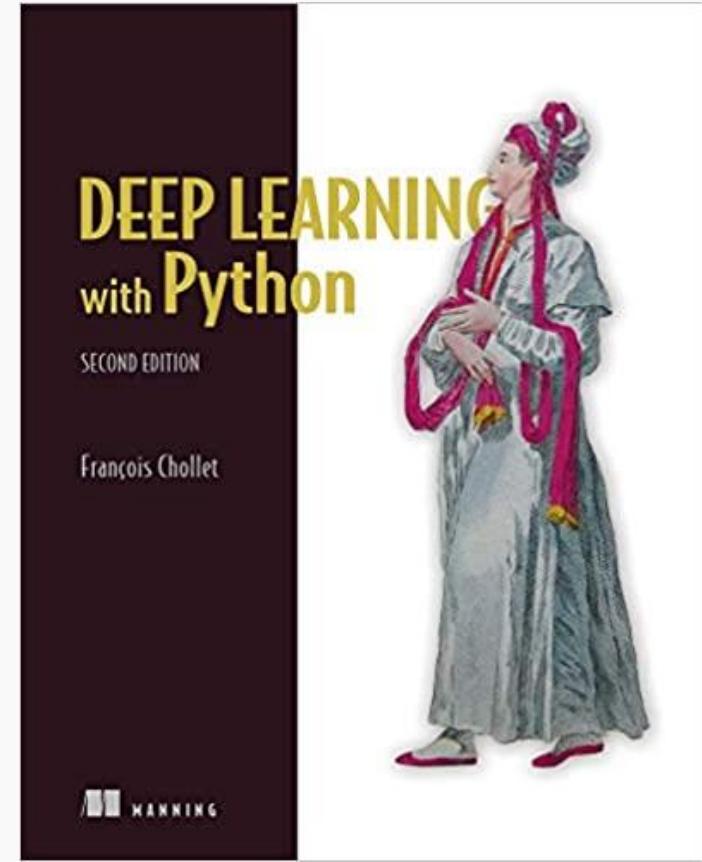
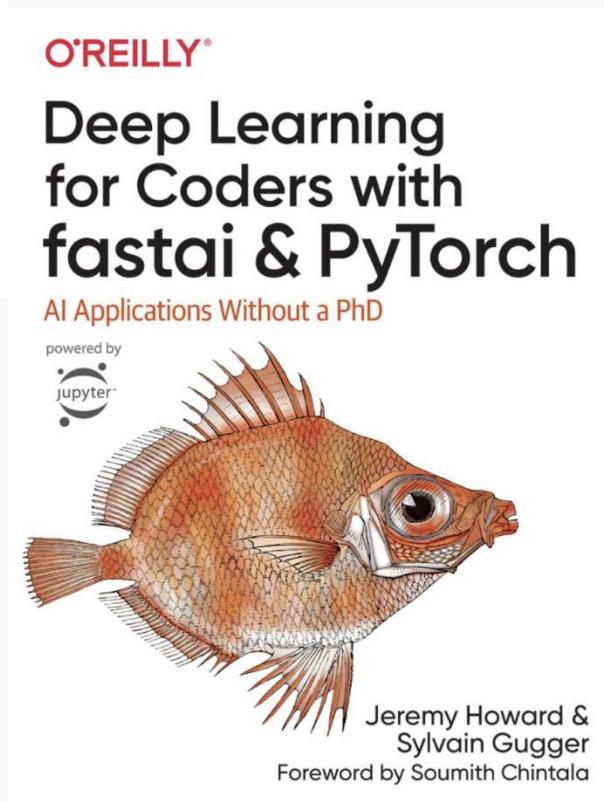


Lecture Outline

- What is data science?
- Why data science?
- **How to learn and why CS109A?**
- What is this class: who, how, what?
- Demo



Learn by Reading





Jay Alammar

Visualizing machine learning one concept at a time.
@JayAlammar on Twitter. [YouTube Channel](#)

[Blog](#) [About](#)

explained.ai

Deep explanations of machine learning
and related topics.

Website created by [Terence Parr](#).



Terence is a professor of computer science and was founding director of the [MS in data science program](#) at the University of San Francisco. While he is best known for creating the [ANTLR parser generator](#),

Terence actually started out studying neural networks in grad school (1987). After 30 years of parsing, he's back to machine learning and really enjoys trying to explain complex topics deeply and in the simplest possible way. Follow [@the_antlr_guy](#).

Lil'Log

[Archive](#) [FAQ](#) [Contact](#)

Jul 11, 2021 [generative-model](#) [math-heavy](#)

What are Diffusion Models?

Diffusion models are a new type of generative models that are flexible enough to learn any arbitrarily complex data distribution while tractable to analytically evaluate the distribution. It has been shown recently that diffusion models can generate high-quality images and the performance is competitive to SOTA GAN.

May 31, 2021 [representation-learning](#) [long-read](#) [language-model](#)

Contrastive Representation Learning

The main idea of contrastive learning is to learn representations such that similar samples stay close to each other, while dissimilar ones are far apart. Contrastive learning can be applied to both supervised and unsupervised data and has been shown to achieve good performance on a variety of vision and language tasks.

Mar 21, 2021 [nlp](#) [language-model](#) [safety](#)

Reducing Toxicity in Language Models

DEEP LEARNING

DS-GA 1008 · SPRING 2021 · NYU CENTER FOR DATA SCIENCE

INSTRUCTORS	Yann LeCun & Alfredo Canziani
LECTURES	Wednesday 9:30 – 11:30, Zoom
PRACTICA	Tuesdays 9:30 – 10:30, Zoom
FORUM	r/NYU_DeepLearning
DISCORD	NYU DL
MATERIAL	2021 repo

2021 edition disclaimer

Check the repo's [README.md](#) and learn about:

- Content new organisation
- The semester's second half intellectual dilemma
- This semester repository
- Previous releases

Lectures

Learn by Watching

Full Stack Deep Learning

Search GitHub

Home Spring 2021 Fall 2019

Spring 2021

Spring 2021 Schedule

Course Projects Showcase

Lectures

Lecture 1: DL Fundamentals

Notebook: Coding a neural net

Lecture 2A: CNNs

Lecture 2B: Computer Vision

Lecture 3: RNNs

Lecture 4: Transformers

Lecture 5: ML Projects

Lecture 6: MLOps

Infrastructure & Tooling

Lecture 7: Troubleshooting

Deep Neural Networks

Lecture 8: Data Management

Lecture 9: AI Ethics

Lecture 10: Testing & Explainability

Full Stack Deep Learning - Spring 2021

We've updated and improved our materials for our 2021 course taught at UC Berkeley and online.

Synchronous Online Course

We offered a [paid synchronous option](#) for those who wanted weekly assignments, capstone project, Slack discussion, and certificate of completion.

Enter your email below or follow us on [Twitter](#) to be the first to hear about future offerings of this option.

And check out the [course projects showcase](#).

email address

Subscribe

Week 1: Fundamentals

We do a blitz review of the fundamentals of deep learning, and introduce the codebase we will

Table of contents

- Week 1: Fundamentals
- Week 2: CNNs
- Week 3: RNNs
- Week 4: Transformers
- Week 5: ML Projects
- Week 6: Infra & Tooling
- Week 7: Troubleshooting
- Week 8: Data
- Week 9: Ethics
- Week 10: Testing
- Week 11: Deployment
- Week 12: Research
- Week 13: Teams
- Week 14-16: Projects
- Other Resources

The screenshot shows a YouTube channel page for 'Weights & Biases'. The channel has 94.3K subscribers. The main navigation bar includes HOME, VIDEOS (selected), PLAYLISTS, COMMUNITY, CHANNELS, ABOUT, and a search icon. A red 'SUBSCRIBE' button is visible on the right.

Introduction to Machine Learning

12 videos • 21,804 views • Last updated on Apr 16, 2019

PLAY ALL

Videos:

- 1 Intro to ML: Course Overview (1:51)
- 2 What is Machine Learning? (19:59)
- 3 Build and debug a neural network (21:09)
- 4 Multi-Layer Perceptrons (18:58)
- 5 Convolutional Neural Network (12:36)

Uploads:

- ALL-TERRAIN ROBOTS ARE C (23:39) [ML News] Facebook AI adapting robots | Baidu... 6.1K views • 1 day ago CC
- I'm taking a break (1:14) 9.2K views • 5 days ago
- COPYRIGHT GITHUB COPILOT (27:01) [ML News] GitHub Copilot - Copyright, GPL, Patents &... 14K views • 1 week ago CC
- FULL SELF-DRIVING VISION ONLY (23:47) Self-driving from VISION ONLY - Tesla's self-driving... 23K views • 1 week ago
- CVPR: SOCIAL MEDIA BANNED (18:27) [ML News] CVPR bans social media paper... 10K views • 2 weeks ago CC

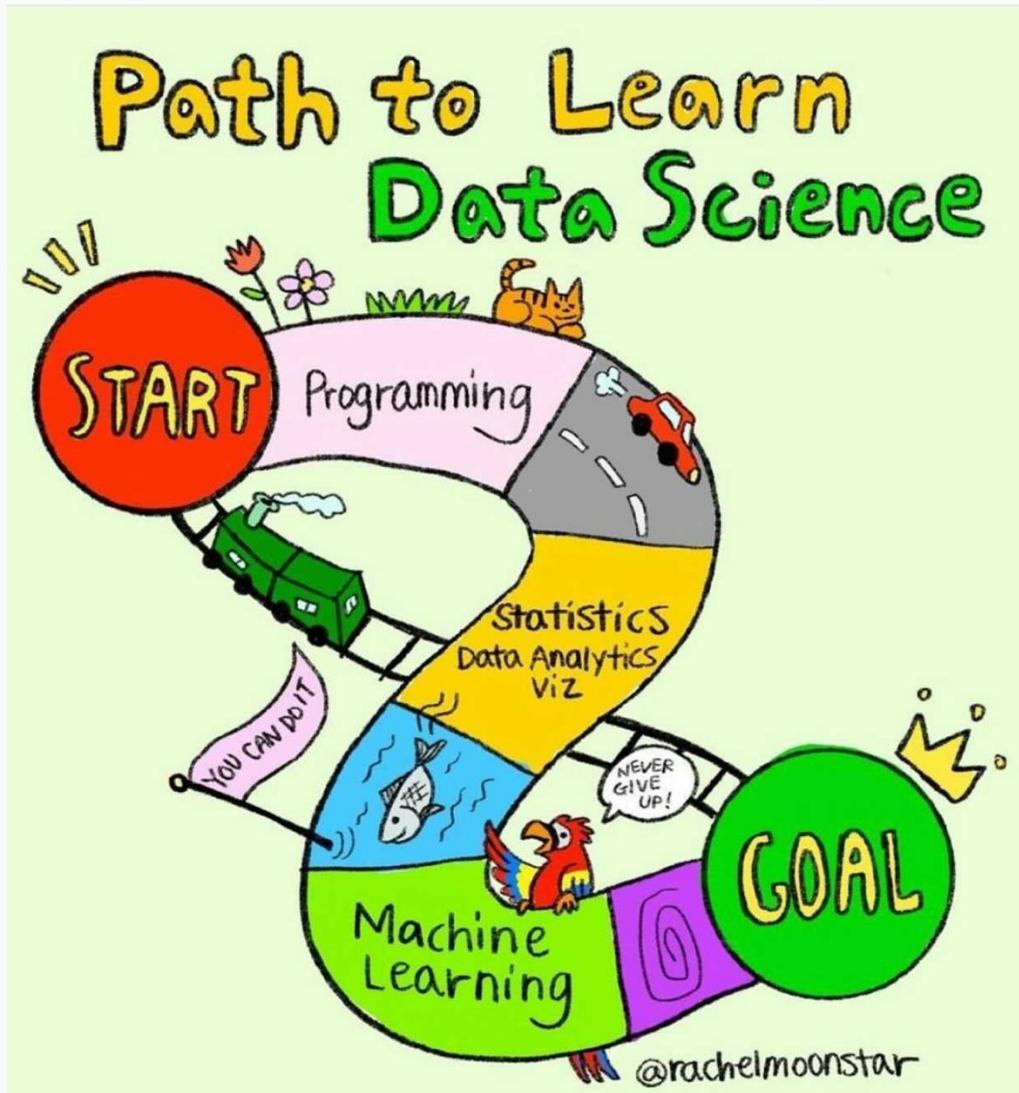
Other sections:

- The Dimpled Manifold Hypothesis Explained! Demystifying Adversarial Examples (1:14:21)
- MCDO AI DRIVE THRU (15:52) [ML News] Hugging Face course | GAN Theft Auto | ...
- XCiT: Cross-Covariance Image Transformers Explained! (35:40)
- AMP: Adversarial Motion Priors for Stylized Physics-Based Character Control (34:45)
- ML NEWS Tensorflow launches "Forum" (17:02)
- De-Biasing GPT-3 | RL cracks chip design | ...

Lecture Outline

- What is data science?
- Why data science?
- **How to learn and why CS109A?**
- What is this class: who, how, what?
- Demo

Memes!



Lecture Outline

- What is data science?
- Why data science?
- How to learn and why CS109A?
- **What is this class: who, how, what?**
- Demo

Why?

Why are you here?

What?

The material of the course will integrate the five key facets of an investigation using data:

1. **Data collection:** data wrangling, cleaning, and sampling to get a suitable data set.
2. **Data management:** accessing data quickly and reliably.
3. **Exploratory data analysis;** generating hypotheses and building intuition.
4. **Prediction or statistical learning.**
5. **Communication:** summarizing results through visualization, stories, and interpretable summaries.

Goals of the course

Theory/Intuition

1. Key Machine Learning concepts
2. Important metrics for evaluation
3. Extracting insights from analysis of the models

Practice

1. Implement ML and deep learning models using python libraries
2. Using free online tools and resources for data science
3. Handling different kinds of data

Impact

1. Solving real-life problems using DS
2. Evaluating the social impact of DS

Weeks 1-2: Data

Data Formats + Web Scraping
Pandas

Weeks 3-5: Regression

kNN Regression
Linear Regression
Multi and Poly Regression
Model Selection and Cross Validations
Inference
Bootstrap
Ridge and Lasso Regularization

Weeks 6: Data Issues

PCA
Missingness

Weeks 7: Data Issues

Midterm 1

Weeks 8: Classification

Logistic Regression

Week 9: Causal Reasoning

Causal Inference

Weeks 10-13: Decision Trees

Decision Trees
Bagging
Random Forest
Boosting Methods
Mixture of Experts

Weeks 14

Ethics



MOVIECLIPS.com

After CS109A

CS109B

A. Neural Networks:

- MLP
- CNNs
- RNNs
- Generative models
- Deep RL

B. Unsupervised Clustering

C. Bayesian Modeling

AC215 Next Fall

A. Productionize Data Science, from notebooks to the cloud

B. Big models, transfer learning and architecture learning

C. Design and Development

D. Deployment, Scaling, & Automation

Not an exclusive list

- CS171/CS271 (Visualization)
- CS181 (ML)
- CS18A (AI)
- CS 187 (NLP)
- Stat 110 (Probability)
- Stat 111 (Inference)
- Stat 139 (Linear Models)
- Stat 149 (Generalized Linear Models)
- Stat 131 (Time Series)
- Stat 171 (Stochastic Processes)
- Stat 195 (Statistical Machine Learning).
- CS208 (Privacy)
- CS282R (ML: Generative Models)
- CS282BR (Sequential Learning)
- AC295/CS287 (DL for NLP)

Who? Instructors



Pavlos Protopapas

Scientific Director
For DS and CSE
masters programs

Principle Investigator of StellarDNN, a research lab within IACS/SEAS. Research in the intersection of [astronomy](#), ML and statistics. He uses Neural Networks to solve problems in astronomy and physics and applying NLP techniques in astronomical time series analysis.

He loves classical music and opera, and he often visits the Boston Symphony Orchestra.

A certified cook from *Le Cordon Bleu* but loves [eating](#) more than cooking.

Funny fact: During a failed military service he was declared the worst soldier in NATO.

tiktok: @pavlosprotopapas



Digestion Time

Who? Instructors



Natesh Pillai
Professor of
Statistics

He graduated from Duke University in 2008 and did his post-doctoral research at Warwick University.

His interests are the interface of applied probability and statistics, with a particular research focus on climate.

Natesh is also part of the Harvard Data Science Initiative. He was awarded the young scientist award by the International Indian Statistical Association in 2018. He is currently a distinguished engineer at LinkedIn working on responsible AI. Prior to that, he was a chief scientist at Correlation One, where he developed a data science curriculum for professionals and trained a few cohorts of students across the world.

In his free time, he dabbles in chess.

Who? Preceptor



Chris Gumb
Preceptor
SEAS

Chris has been a member of the CS109A & B teaching staff for the past 7 years.

As preceptor, he teaches labs, coordinates the TF team, develops course materials, and handles logistics.

When not answering your Ed posts and emails he enjoys making music and seeing films with friends.

Frequently spotted at the local independent movie theaters, he's basically made of popcorn 

Who? ~40 Teaching Fellows!

Omar Abdel Haq

Bailey Bai

Kushagra Chitkara

Labdhi Gandhi

Leslie Gu

Panthon Imemkamon

Ziqing Luo

Megan Luu

Shiyu Ma

Tanner Marsh

Siona Prasad

Robert Roessler

Elaine Swanson

Yuan Tang

Xu (Victoria) Tang

Xinjie Yi

Jacob Yu

Haoran Zhang

Rama Edlabadkar

Aalto Lin

Li Yao

Eunice Liu

Matthew Andrews

Tina Gong

Dhati Oommen

Pranav Ramesh

Aseel Rawashdeh

Josh Rosenblum

Omar Mohammad Siddiqui

Dhrubhagat Singh

Eric Tang

Alice Wu

Matthew Andrews

... and more!

Course Components

Lectures, Labs and Office Hours

In lecture we'll **cover the material** that you will need to complete the **homework** and to survive the rest of your life in CS109A.

We will use a mix of slides and exercises via *edstem*.

1. Lecture slides and associated notebooks will be posted before lecture on *edstem*.
2. Lectures will be video taped (and live streamed for the extension school students) and are usually posted on Canvas within 24 hours.

Mon/Wed 9:00-10:15am **in person** @Science Center Hall B and @Zoom for Extension School Students (zoom link is on canvas under zoom).

Lecture format

ASYNCHRONOUS

- Quiz
- Finish exercises from previous lecture
- Reading

SYNCHRONOUS

Questions from asynchronous material and review

Live Lecture

Q&A

Hands-on exercises in breakout rooms

Discussion about the exercises

Repeat

:

Summary and conclusions

Lectures, **Labs**, and Office Hours

Labs will be a mix of review material, tutorials on how to practically solve problems with Python libraries, and some hands-on exercises.

Friday 9:00*-10:15am [in person](#) @Science Center, Hall B [@Zoom](#) for Extension School

Attendance

Attending class isn't just required; it's something I look at closely when deciding on academic and professional recommendations.

Please understand that consistent presence and engagement in the classroom are highly valued in this course.



Attendance

All lectures are videotaped, so you can watch them later if you can't attend.

BUT

You will earn 1 extra late day for every 8 lectures/labs you attend!

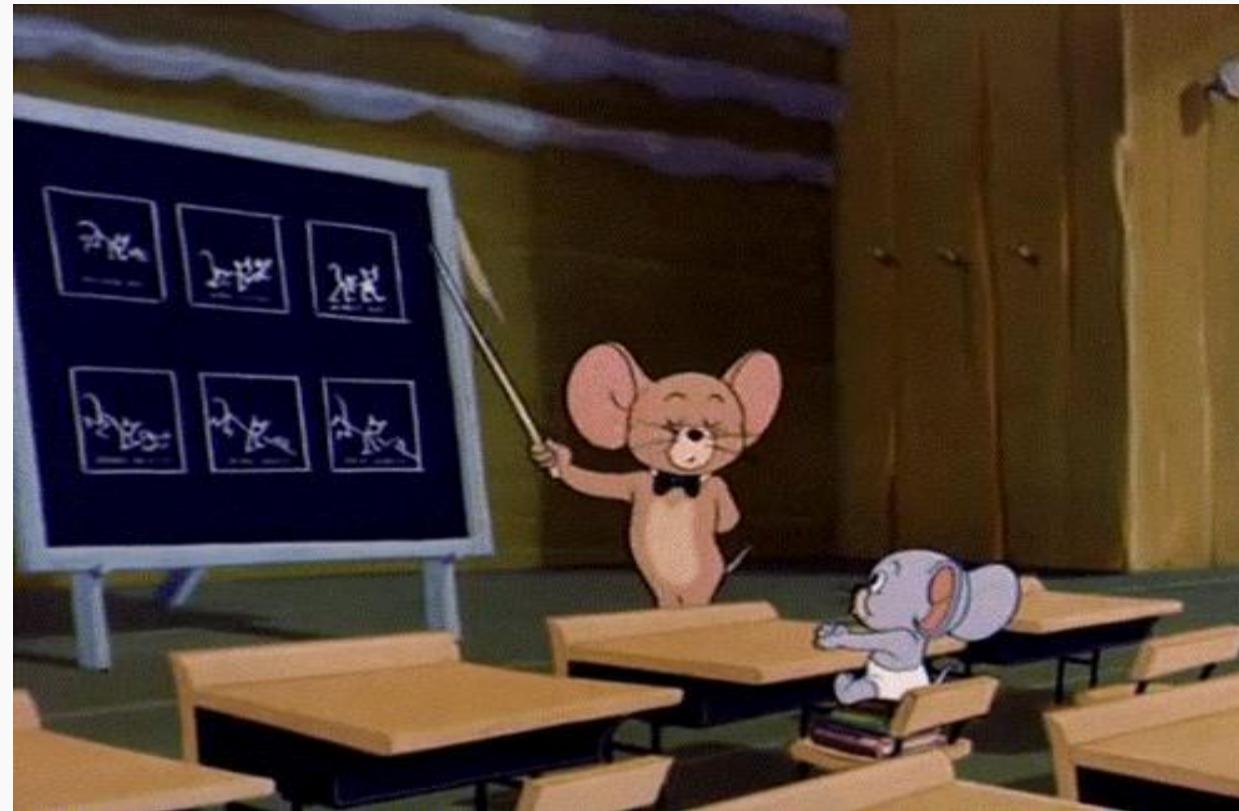


Lectures, Labs, and Office Hours

Office hours will be posted before next week.

There will be a Google calendar made available through Canvas with all course components and OHs.

Assignments



Five Graded Components

Homework: 35%

Homework 0: 1%
Homeworks 1-6: 34%

Students are encouraged to work in pairs on HW assignments.

Exercises: 2%

During lecture.
All test cases are weighted equally.
Due at the beginning of the next morning lecture or lab.

We will only count the exercises category if it helps your overall grade.

Quizzes: 8%

End of each lecture.

1/3 of the quizzes will be dropped from your grade.

All questions are weighted equally.

Due at the beginning of the next lecture or lab.

Midterms: 30%

2 Midterms, each a mix of multiple choice and coding questions.
Multiple choice will be in-person, coding questions will be take-home exam.

Projects: 25%

Milestone dates and details to be announced soon.

Homework(s)

There will be 6 homeworks (not including Homework 0):

- Homework 0 (due Sept 13th; all honest attempts get full credit)
- Homework 1: Web scraping, BeautifulSoup, Basic Pandas, and Plotting
- Homework 2: Regression kNN and LinReg
- Homework 3: Multi- & polynomial Regression, Regularization, Inference
- Homework 4: High Dimensional Data and PCA
- Homework 5: Logistic Regression
- Homework 6: Trees, Bagging, Random Forest, and Boosting

Homework(s)

You are encouraged but not required to submit **in pairs** on HWs 1-6

We will be using the Groups function on Canvas to do this, details to be announced later.

HWs 1-6 are **due 10 pm Wednesdays**, and homework will be released on Wednesdays.

Late submission policy: Each student is allowed up to 4 late days over the semester with at most 1 day applied to any single homework. Outside of these allotted late days, late homework will **not be accepted**.



Digestion Time

Final Project

There will be **a final group project** (3-5 students) due during exams period.

- You can propose to use a (public) data set of your choice and your own project definition (to be approved by the instructors).
- Project proposal process starts September 27th.

Help

The process to get help is:

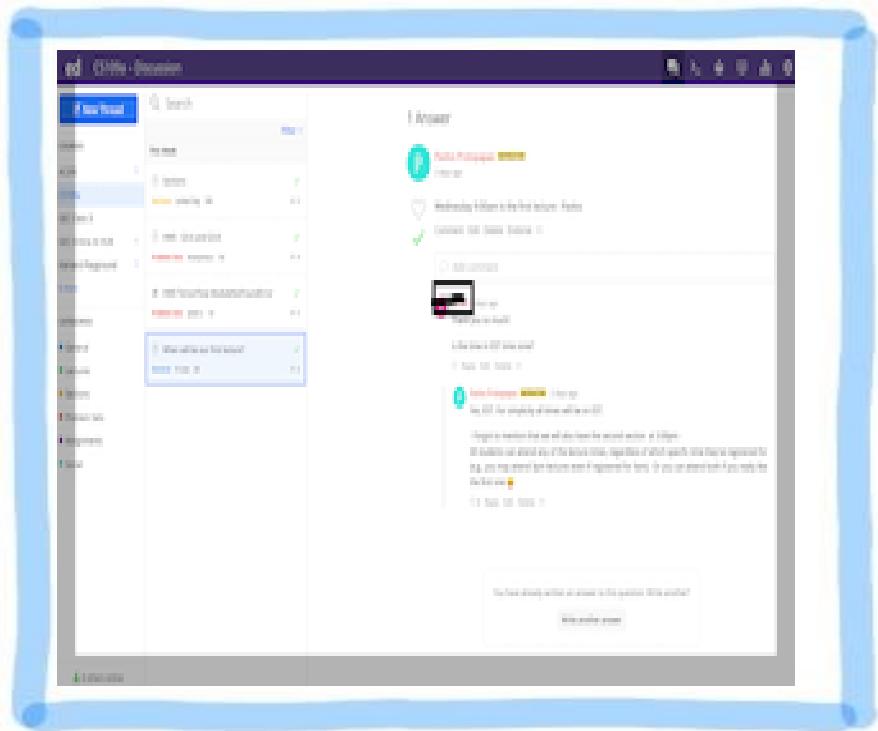
1. Post the question on *Edstem*, and hopefully, your peers will answer. The teaching staff also monitor and respond to posts.
2. Attend the [Office Hours](#); this is the best way to get help.
3. For private matters, send an email to the Helpline:
cs1090a2024@gmail.com.
4. For personal matters, send an email to Pavlos or Natesh.

Prompt for LLMs:
Write an email to a cranky professor.
Keep it concise and under 30 words.

[Weekends will be slow days, so please be patient!](#)

Tools for the course

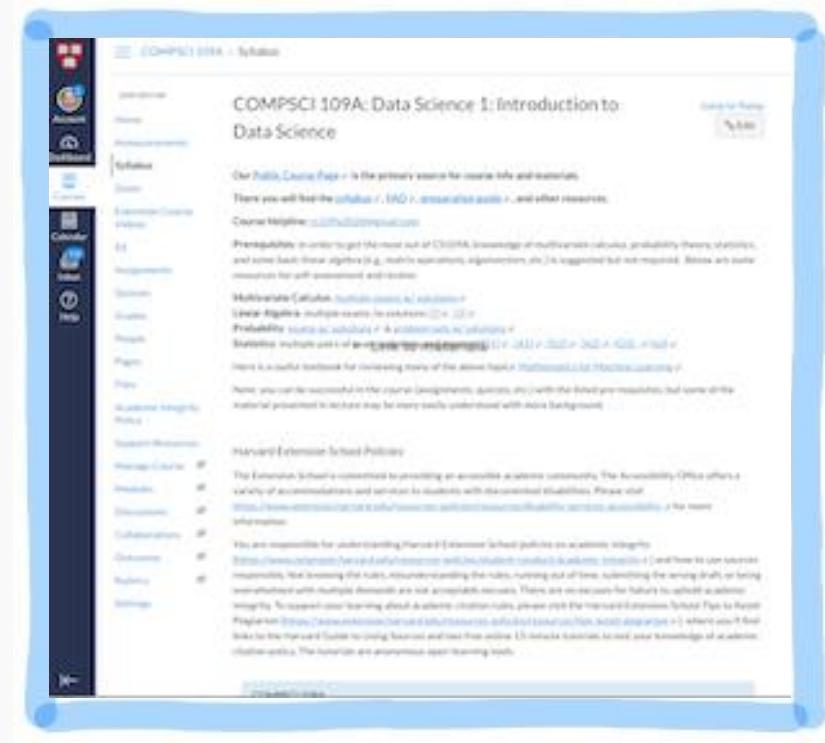
edstem



- Forum
- Quizzes
- Reading assignments
- Hands on exercises
- Lecture slides



Canvas



- Syllabus
- Schedule
- Homework Assignments
- Video Recordings
- Grades

Can I audit this class?

Yes, CS109A does accept auditors, but all auditors must agree to abide by the rules described in the syllabus

Can I take this class asynchronously?

College students: This is not allowed.

Graduate students: This is not ideal. Attending classes is very important and part of being a student here. The decision is yours and your program academic coordinator. We feel you should attend at least 50% of the classes.

Am I prepared for this class?

Proficiency in Python, basic math (calculus), basic stats are expected.

We offered a class called *Bedrock Data Science* this summer which helps with some of these topics.

We are making these material available under resources on ED for you to brush up on your Python, linear algebra, and statistics.

If I miss a class, will it affect my grade?



FAQ

I have a trip planned during the midterm. Can I take the midterm earlier or later?

Midterm 1 is on 10/18 at 9:00 am in person

Extension school have 3 Zoom time slots on 10/18 & **10/19**.

Midterm 2 is on 12/11 at 9:00am in person

Extension students have 3 Zoom time slots across 12/11 & **12/12**

Make sure these are on your calendar!

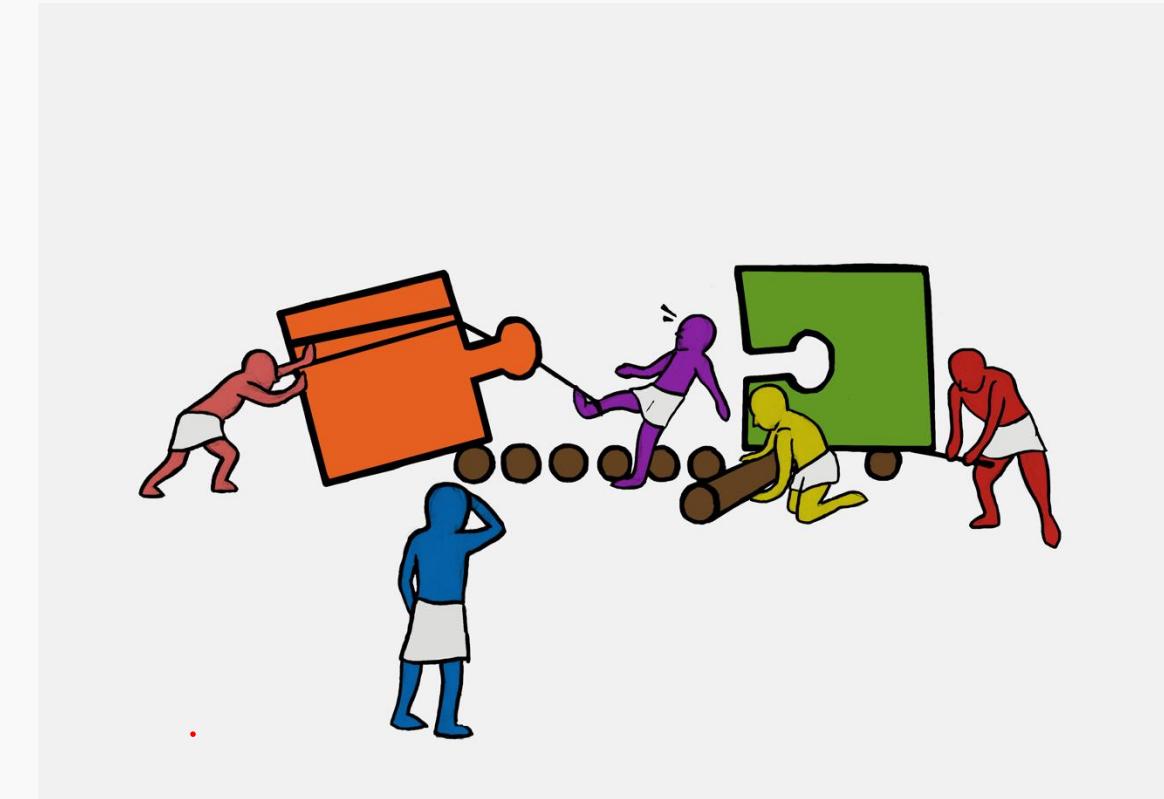
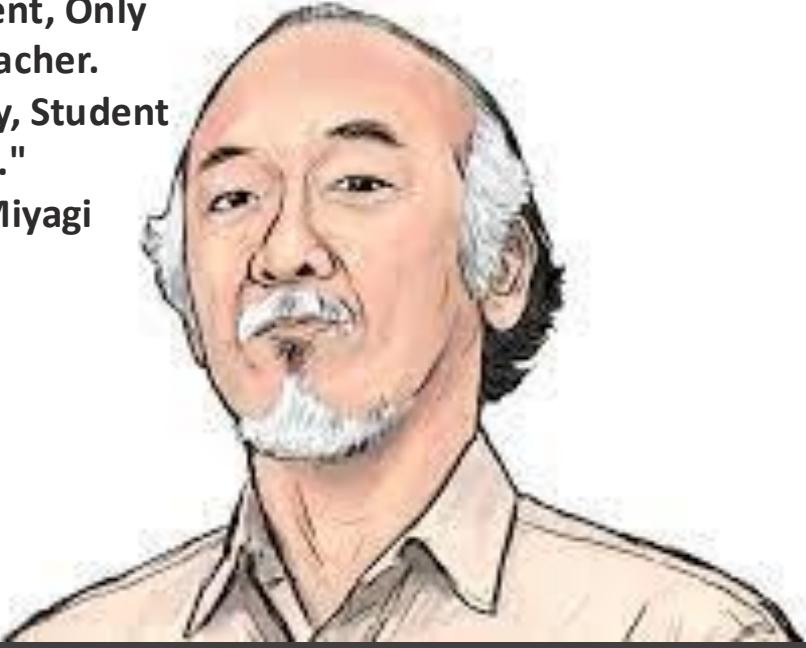
I have a project in mind. Can I use it for the course?

Yes, as long as the data are public and you're willing to work with other students.

Lecture Outline

- What is data science?
- Why data science?
- How to learn and why take CS109A?
- What is this class: who, how, what?
- **Demo**

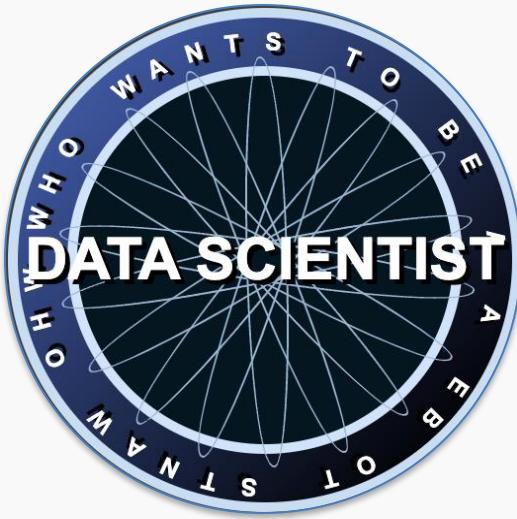
**"No Such Thing As
Bad Student, Only
Bad Teacher.
Teacher Say, Student
Do."**
- Mr. Miyagi



Breakout rooms and in-class exercises







CS109A

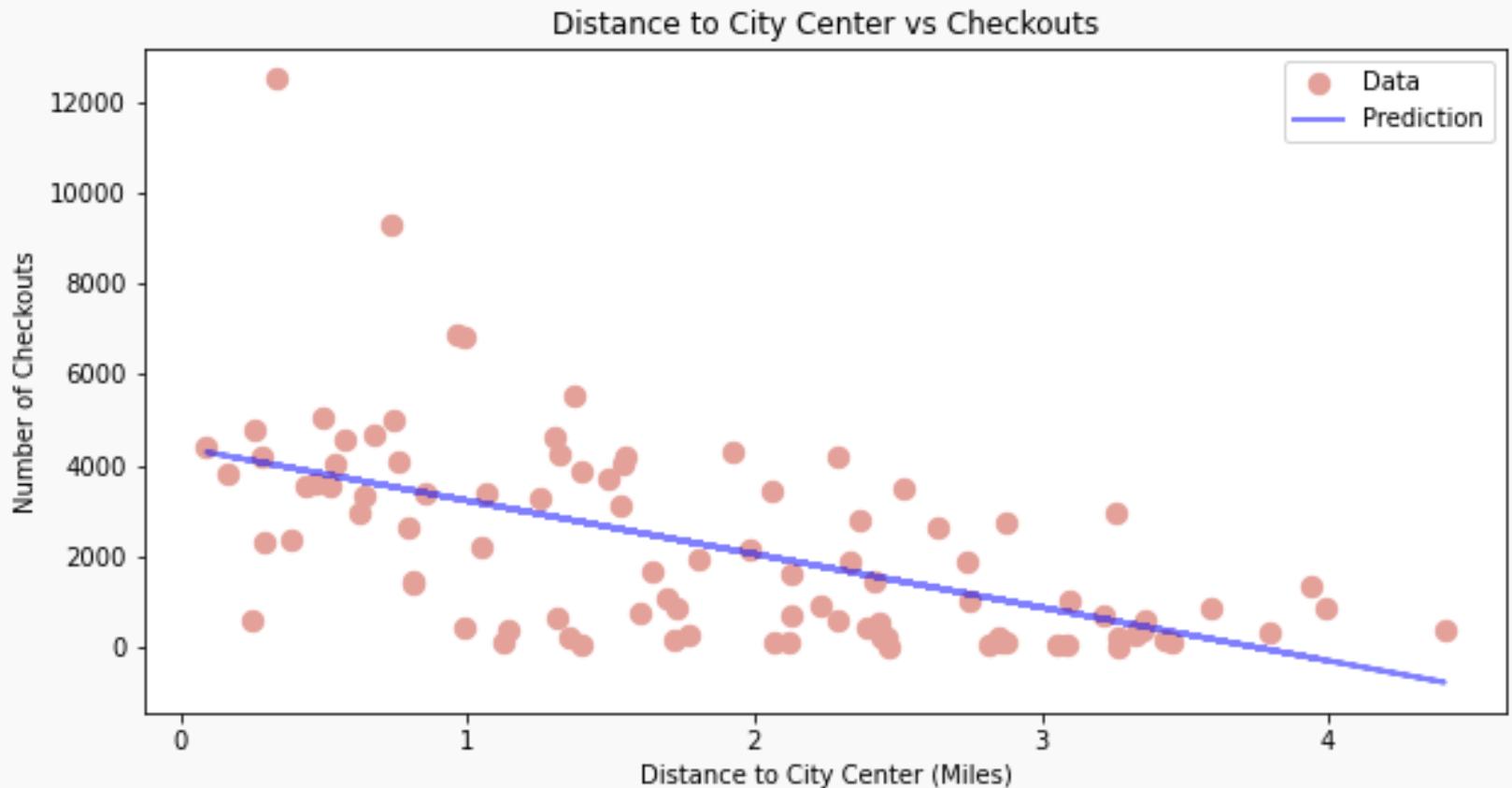
GAME Time



Based on our "linear" model, what would most likely be the number of checkouts for a distance of 2.5 miles from the city center?

Options

- A. 45000
- B. 12530
- C. 1450
- D. 650

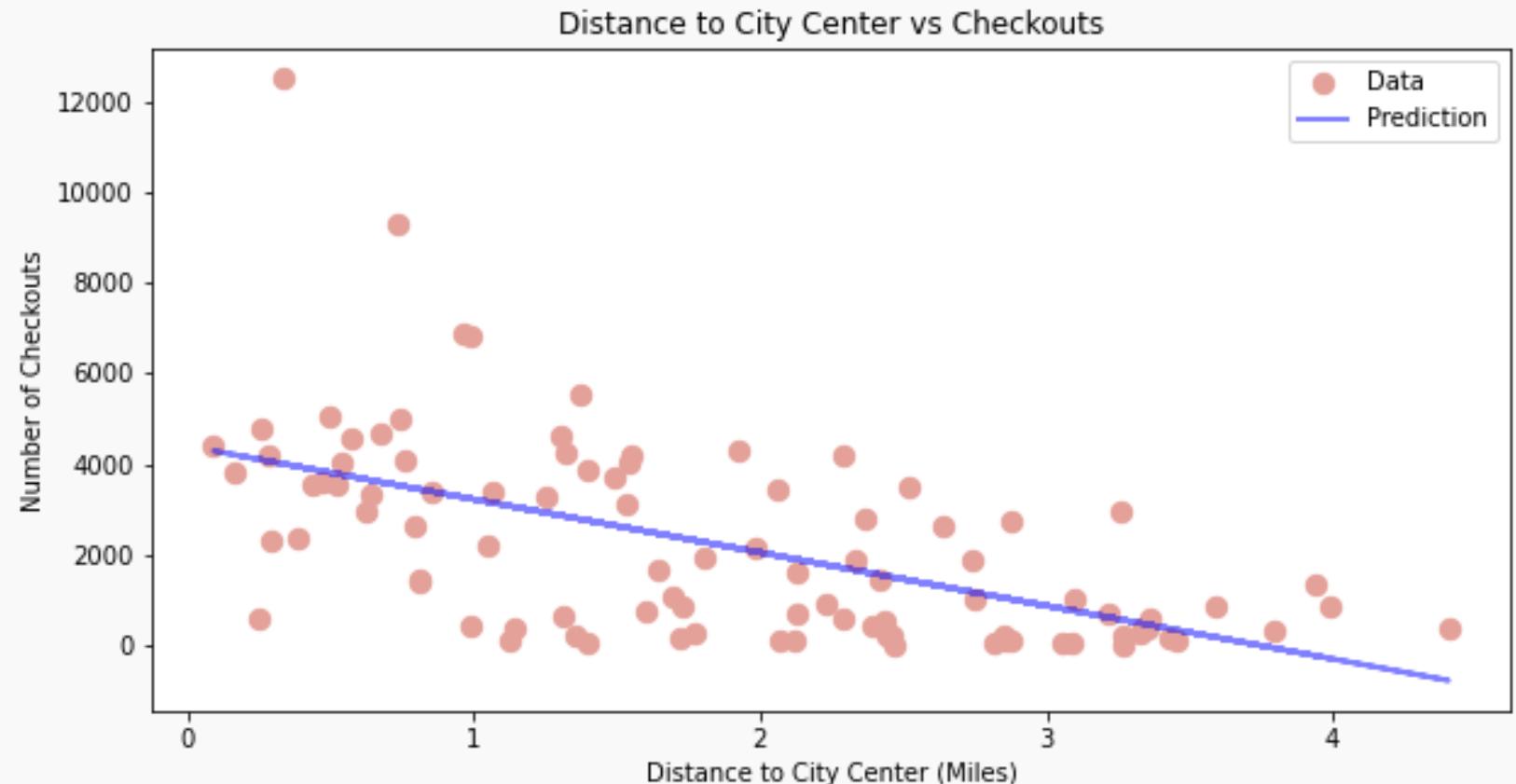




Based on our "linear" model, what would most likely be the number of checkouts for a distance of 2.5 miles from the city center?

Options

- A. 45000
- B. 12530
- C. 1450
- D. 650





What is the goal of CS109A?

Options

- A. To teach you data science.
- B. To make your life difficult and painful.
- C. To predict the next stock price crash.
- D. To enable computers to talk.



What is the goal of CS109A?

Options

- A. To teach you data science.
- B. To make your life difficult and painful.
- C. To predict the next stock price crash.
- D. To enable computers to talk.

THANK YOU

Course staff available to answer questions after class today in:

Pierce Hall Room 209
from
10:30 AM - 12:30 PM