

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

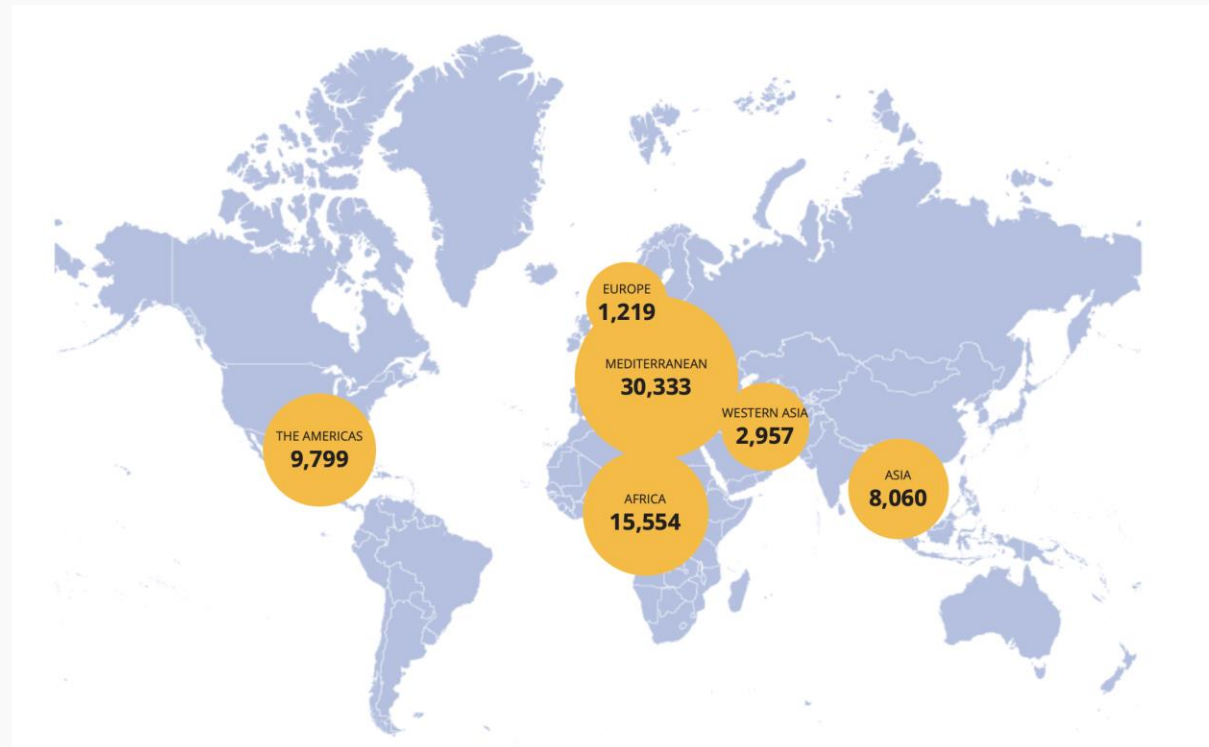
Do I Have Missing Data?

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

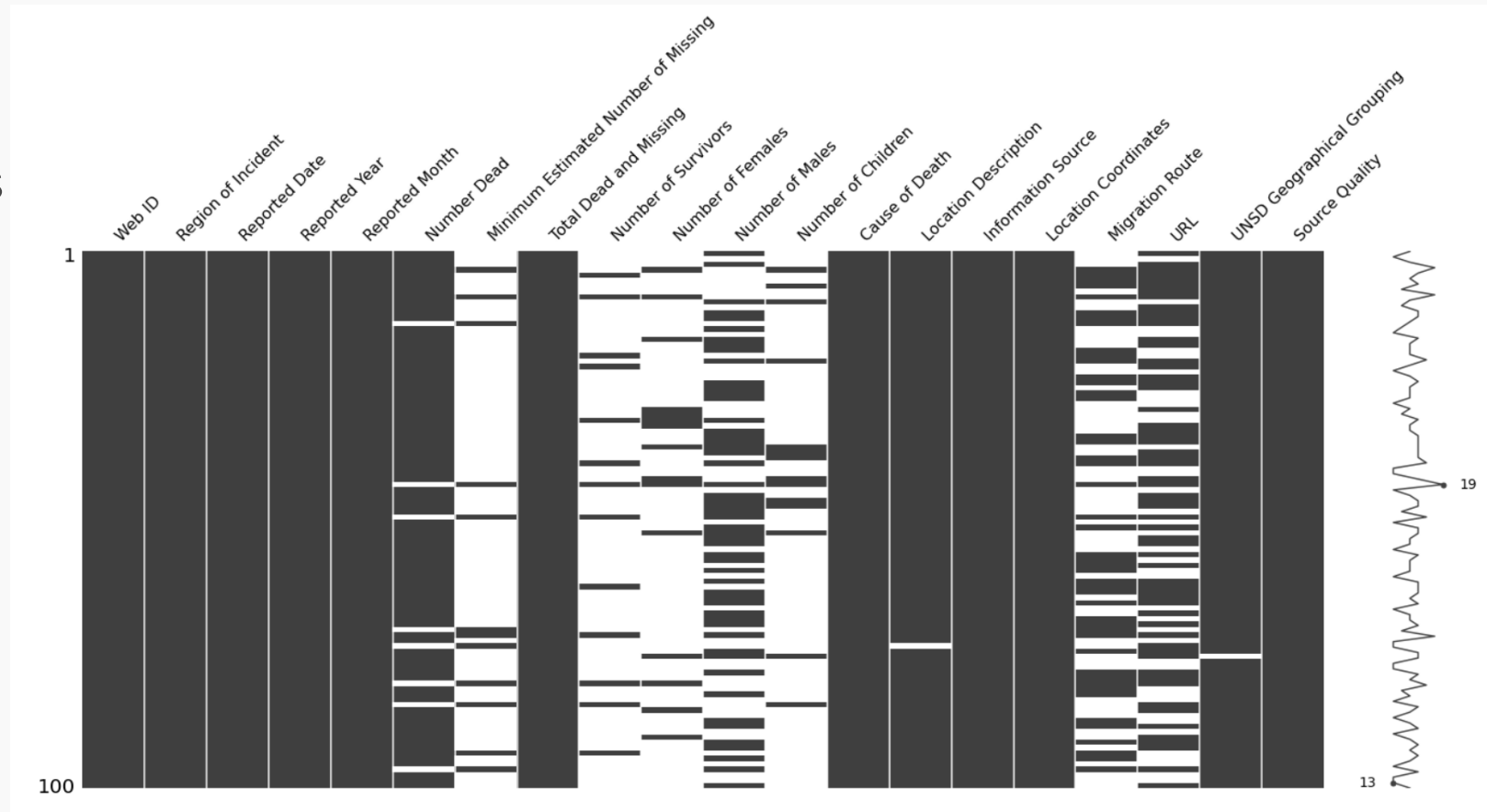
Migrants Project records since 2014 people who die in the process of migration towards an international destination, regardless of their legal status.



Example: Missing Migrants Project

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

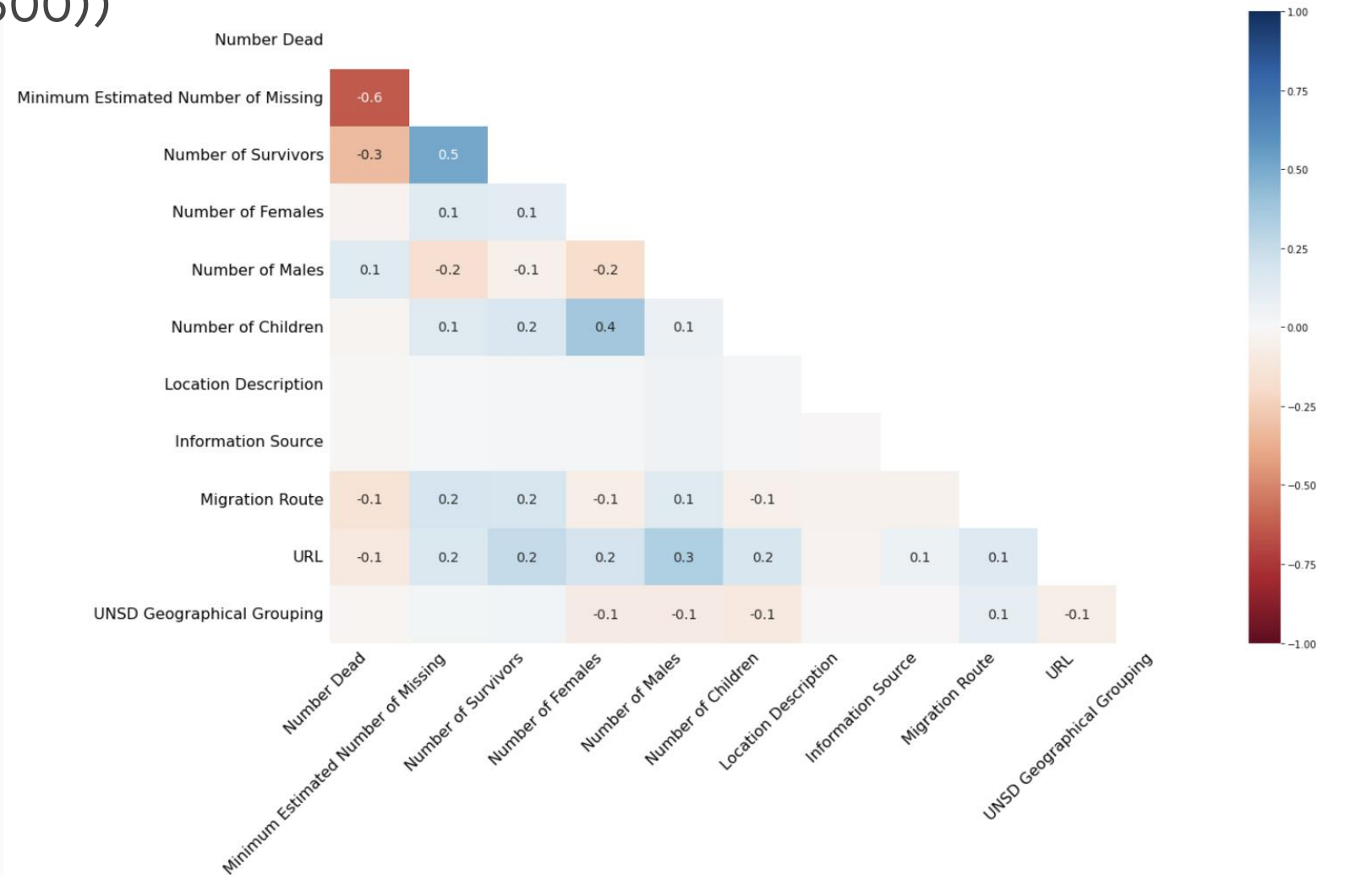
- nullity matrix: quickly visualize missing patterns
- sparkline on the right summarizes the general shape of the data completeness



Example: Missing Migrants Project

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

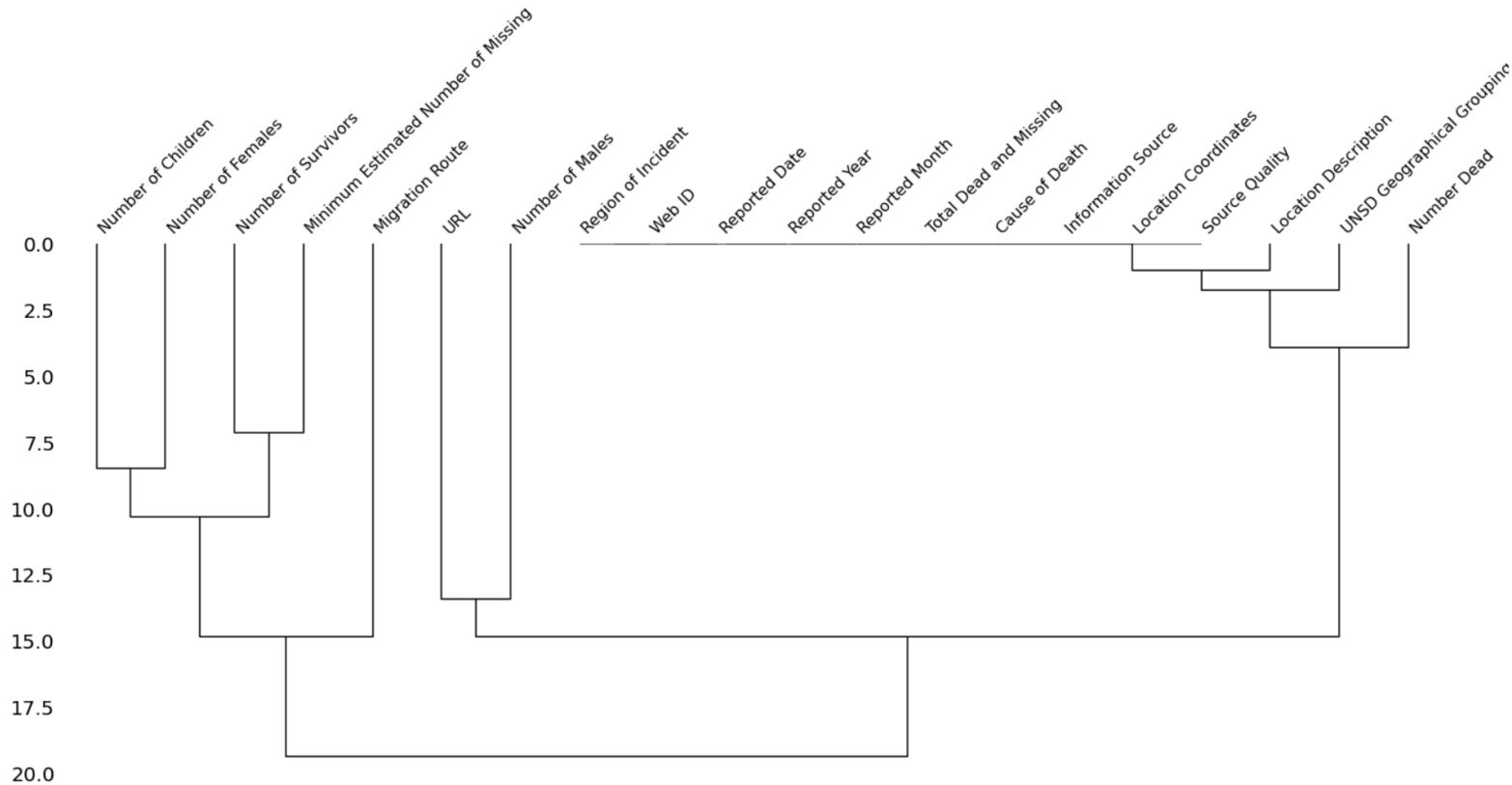
- missingno correlation heatmap: how strongly the presence or absence of one variable affects the presence of another
- sign of missing not completely at random



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

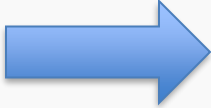
Simple Imputation Methods

- 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.

$x_1 X_1$	$x_2 X_2$		
$x_1 X_1^*$	$x_2 X_2^*$		
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

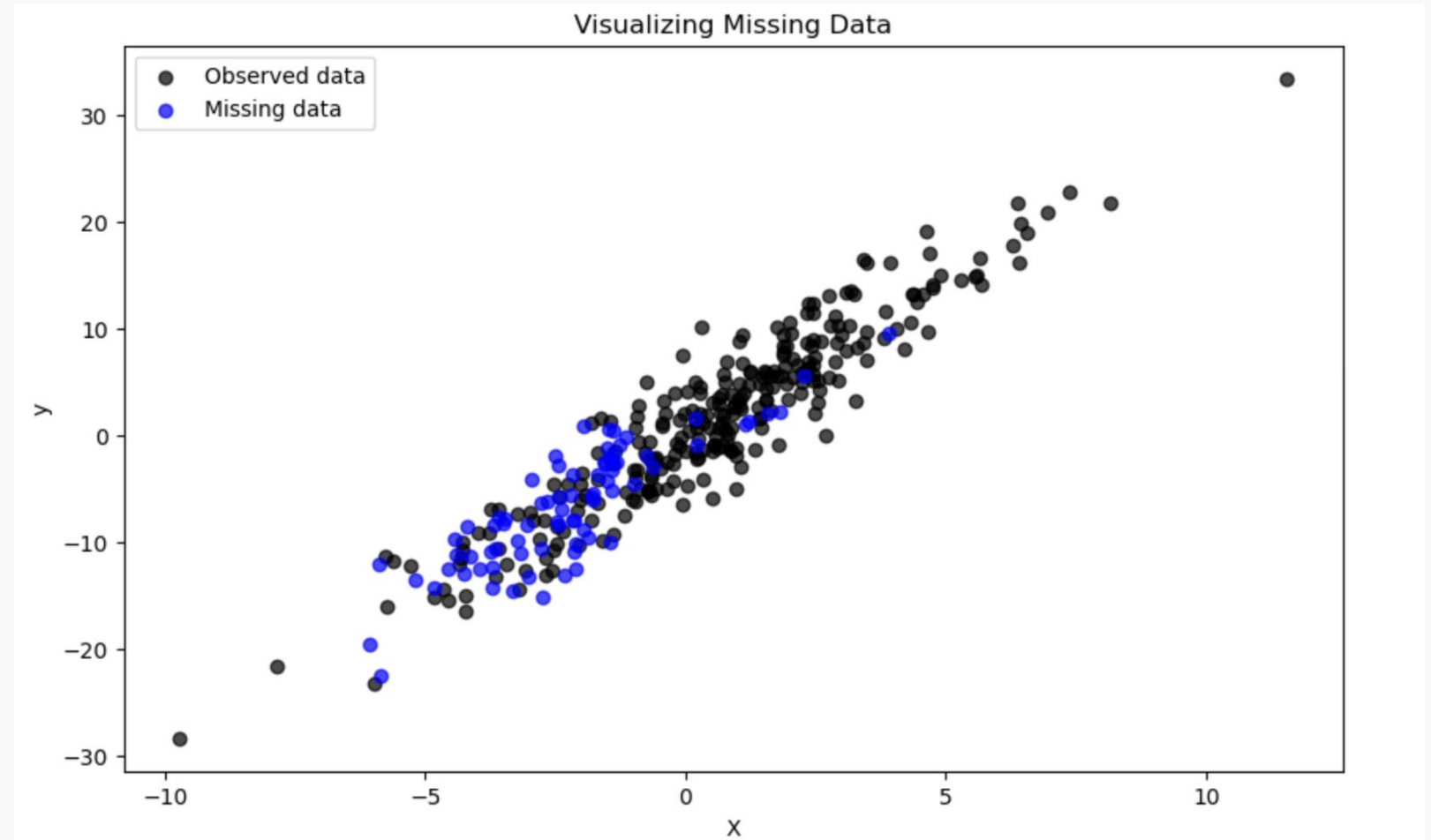
$x1X_1$	$x2X_2$		
10	.		
5	1		
21	0		
15	0		
16	.		
.	.		
21	1		
12	0		
.	1		



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

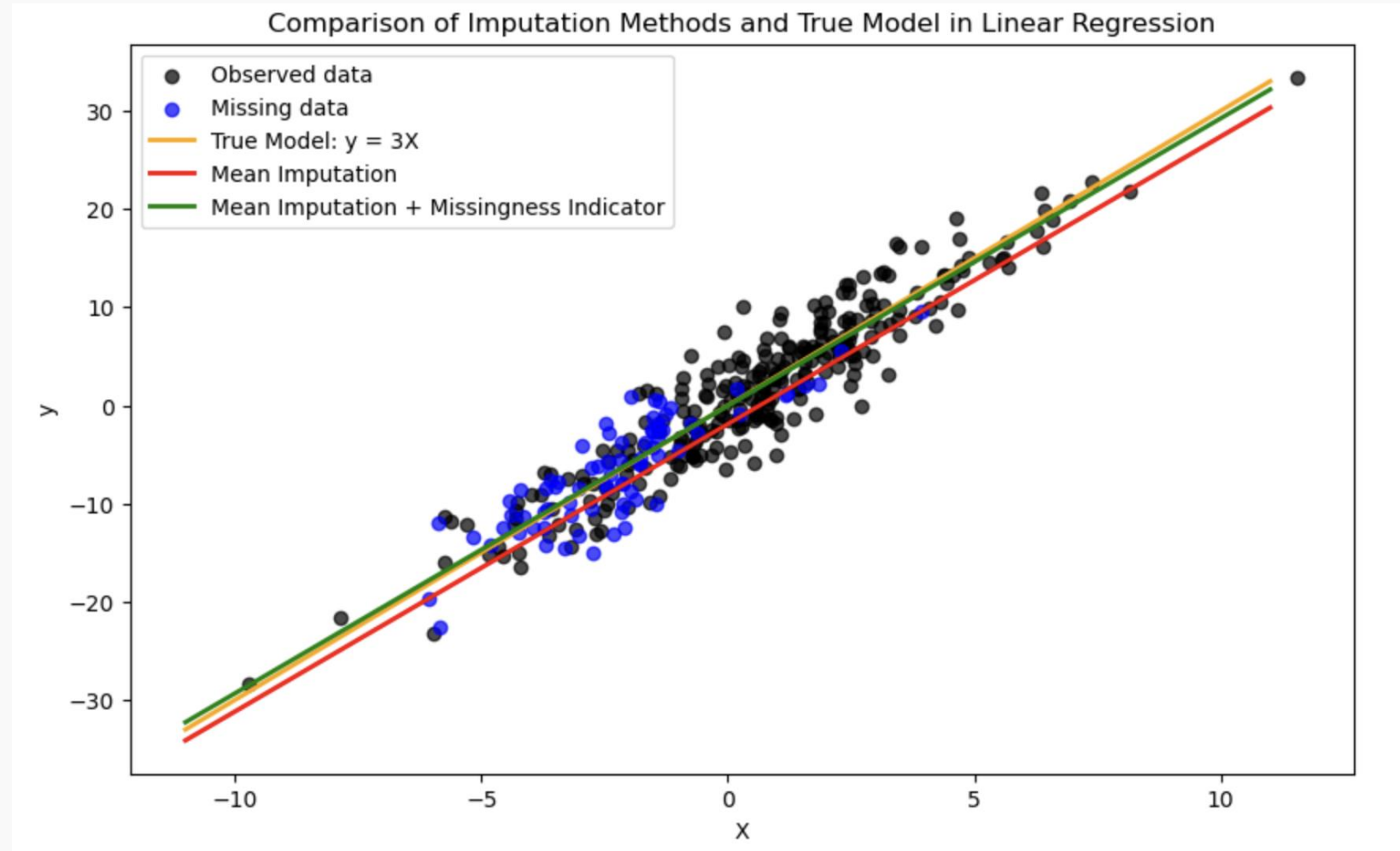
Example: Different Imputation Methods in Linear Regression

- 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?

X	Y
	1
	?
	0.5
	0.1
	?
	10
	0.03








Schematic: imputation through modeling

How do we use models to fill in missing data?

<u>X_train</u>	<u>Y_train</u>	<u>X_test</u>	<u>Y_pred</u>
	1		?
	0.5		
	0.1		
	10		?
	0.03		








Schematic: imputation through modeling

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

X	Y
	1
	? = (1 + 0.5) / 2
	0.5
	0.1
	?
	10
	0.03









Schematic: imputation through modeling

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

X	Y
	1
	? = (1 + 0.5) / 2
	0.5
	0.1
	? = (0.1 + 10) / 2
	10
	0.03

Schematic: imputation through modeling

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

X	Y
	1
	? = m  + b
	0.5
	0.1
	? = m  + b
	10
	0.03

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

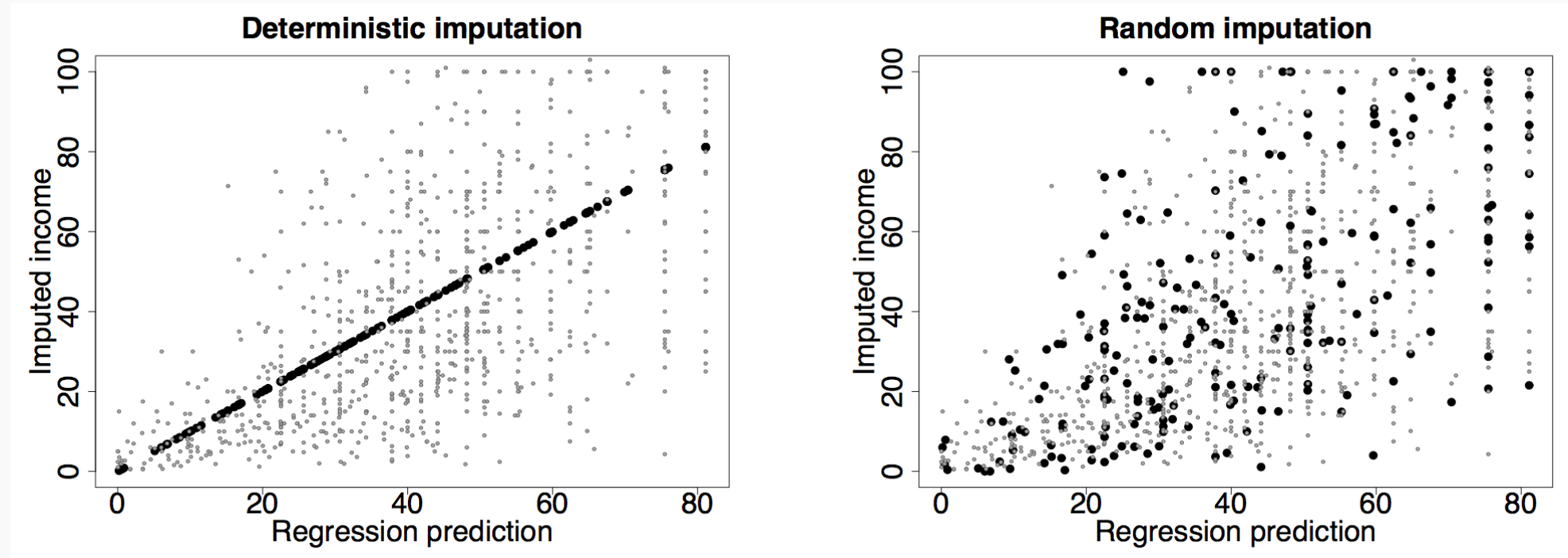
Imputation through modeling with uncertainty

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.