Missingness

Bookmark this page

In all of the datasets we've looked at so far, every observation has always had a recorded value for each predictor variable as well as the response. That is, there were no **missing values**. But in the real world we are usually not so lucky. Missing data can arise in many ways. For example:

- A survey was conducted, and some values were just randomly missed when being entered in the computer
- Someone completing a survey chooses not to respond to a question like, "Have you ever used illegal drugs?"
- You decide to start collecting a new variable partway through the data collection of a study
- You want to measure the speed of meteors, and some of them are just 'too quick' to be measured properly

In what ways are some of the examples above similar? How do they differ? We'll find it useful to categorize all instances of missingness into 3 major types:

1. Missing Completely at Random (MCAR)

For this scenario, the probability of missingness is the same for every observation. This would be like randomly poking holes in a data set. Examples:

- A coin is flipped to determine whether an entry is removed.
- Values were just randomly missed when being entered in the computer.

This is the best-case scenario for missingness, and the easiest to handle. Because the missingness does not depend on any other predictors, either included in our data or not, we can simply remove (or 'drop') observations with missing values. This would be equivalent to removing a random subset of the data, which should not bias the remaining dataset toward observations of a certain kind. This should then have no effect on our inference (that is, the estimate of the betas).

Inserting or 'imputing' predictions for the missing values is also an option and will be discussed in more detail below.

2. Missing at Random (MAR)

For this scenario, the probability of missingness depends only on available information (in other predictors). While less ideal than MCAR, MAR is still a case that can be handled.

An example for MAR would be men and women responding at different rates to the question, "have you ever been harassed at work?"In this hypothetical example let's assume that women were more likely than men to leave the question about harassment blank. In this scenario, dropping observations with this response missing would bias our data as we would drop more women than men.

Effect if you ignore: inferences (estimates of the β's) are biased and predictions are usually worsened.

How to handle: use the values in the other predictors to build a model to impute values for the missing entries.

3. Missing Not at Random (MNAR)

For this scenario, the probability of missingness depends on information that has not been recorded. Missing Not at Random is the **worst-case** scenario, and impossible to resolve. Some examples are:

- Patients drop out of a study because they experience some really bad side effect that was not measured.
- Cheaters are less likely to respond when asked if they've ever cheated.

Effect if you ignore: inferences (estimates of betas) become biased, and predictions are usually worsened.

How to handle: you can attempt to 'improve' things with an imputation method similar to that recommended for MAR, but you may never completely fix the bias.

The bad news is that we can never be certain about what type of missingness we are dealing with by inspecting the data. Generally, it cannot be determined whether data are missing completely at random (MCAR) or whether the missingness depends on unobserved predictors or the *missing data themselves* (MNAR). The problem is that these potential "lurking variables" are unobserved (by definition)

never be completely ruled out.

In practice, a model with as many predictors as possible is used so that the 'missing at random' (MAR) assumption is reasonable.

Identifying and Handling Missingness

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.

Pandas has a helpful method to quickly discover missingness in your DataFrame. The `isna()` method will return a boolean matrix with `True` in each index that had either `NaN` or `None` and `False` everywhere else.

This boolean matrix can be used for selecting just those indices that have missing values. Or, if you use the negation operator, you can select just those indices without missingness.

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?

Option 1: Dropping

The simplest ways to handle missing data it is by just throwing away or '**dropping**' the observations that have any missing values.

Pandas `dropna(axis=0,1)` will drop any rows or columns with one or more missing values from your Pandas DataFrame.

Dropping is 'easy,' but when is it advisable?

It is probably a good idea to drop a **predictor column** if it is missing large proportion of observations.

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

Option 2: 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

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

Imputing average values is easy in Pandas. First, calculate the average value you wish to impute. For example, the mean. Next, use the `fillna()` method on the column with the missingness you want to fill and pass the value to impute as an argument.

Let's see an example!

```
df['x1'].fillna(df['x1'].mean())
```

This code first calculates the mean of the column `x1`. Note the mean method ignores any missing values in the calculation. It then fills all missing values in `x1` with this mean!

> **⚠ Don't Overdo It!**
>
> If there is too much missingness in a predictor, you may be doing more harm than good by attempting to impute an average!

Imputation Example

Here's a visual explanation for imputation for a dataset with two predictors: X1, which is quantitative, and X2, which is categorical

Skip to after table

X1	X2
10	.
5	1
21	0
15	0
16	.
.	.
21	1
12	0
.	0

For X1 we can replace its missing values by imputing with the column's mean value.

For X2 we can impute using its mode, or most common value (i.e., mode).

Let's call these new predictors with the imputed values $\hat{X_1}$ and $\hat{X_2}$. Now, we *could* proceed with $\hat{X_1}$ and $\hat{X_2}$ as our only 2 predictors. But doing so would in effect be throwing away some potentially useful information.

What information could that be? **The fact that these values were originally missing!**

Missingness Indicator Variable

We can capture the fact that a value was missing by creating 2 new **indicator variables**.

For example, the indicator variable $X_{1,miss}$, takes the value 1 for all rows where X_1 originally had a missing value, and 0 otherwise.

Similarly,$X_{2,miss}$, takes the value 1 for all rows where X_2 originally had a missing value, and 0 otherwise.

Skip to after table

X_1	X_2	$X_{1,miss}$	$X_{2,miss}$
10	.	0	1
5	1	0	0
21	0	0	0
15	0	0	0
16	.	0	1
.	.	1	1
21	1	0	0
12	0	0	0
.	0	1	0

Our final design matrix would then be:

- X_1^* - continuous variable with mean imputation
- X^* - categorical variable with mode imputation

- X_2 categorical variable with mode imputation
- $X_{1,miss}$ - indicating previously missing values in the continuous variable X_1
- $X_{2,miss}$ - indicating previously missing values in the categorical variable X_2

Why Use a Missingness Indicator Variable?

Q: How does this missingness indicator variable improve the model?

A: Because the group of observations with a missing entry may be systematically different than those with the variable measured.

Treating them equivalently could lead to bias in quantifying relationships (the βs) and underperformance in prediction.

Example: Imagine a survey question that asks whether or not someone has ever recreationally used opioids, and some people chose not to respond.

Does the fact that they did not respond *itself* 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.

Model-Based Imputation

With simple imputation, we were only using information about the particular predictor we wanted to perform the imputation on. But each observation may have many other predictors, and these other predictors may be related to the missing predictor!

Model-Based imputation uses information contained in the other predictors to help us learn a value to impute.

Here we have two predictors, X_1 and X_2.

Skip to after table

X_1	X_2
red-orange	1
orange	**?**
gold	0.5
yellow	0.1
green-yellow	**?**
green	10
teal	0.03

How can we use a model to fill in the missing values in X_2?

Think of X_1 as the predictor and X_2 as the response. An imputation model can then learn the relationship between X_1 and X_2 from those observations with no missingness.

kNN Imputation

kNN is perhaps the simplest model-based imputation method. Let's see a kNN imputation model in action with k=2.

To impute a missing value, find the missing observation's 2 nearest neighbors in terms of the data we do have (that is, X_1). We only consider observations without missingness as potential neighbors.We then average these neighbors' values for the missing predictor and impute! You can repeat this process for each observation that needs imputation.

Skip to after table

X_1	X_2
red-orange	1
orange	**0.75 = (1 + 0.5) / 2**
gold	0.5

yellow	0.1
green-yellow	**5.05 = (10 + 0.1) / 2**
green	10
teal	0.03

Different Imputation Models

We can use all kinds of models as imputers:

- A regression model for imputing continuous variables
- A classification model for imputing categorical variables

The trick is to treat the column to be imputed as a new response variable.

> **⚠ Don't use the response variable as a predictor!**
>
> You must not use the true response variable as a predictor when imputing. This would cause information about the true response to 'leak' into the predictors.

‹ Previous Next ›