towards
data science

Visualizing Missing Values in Python is Shockingly Easy

How to Use the Missingno Library to See All Your Missing Values

Eirik Berge 1 day ago · 7 min read★

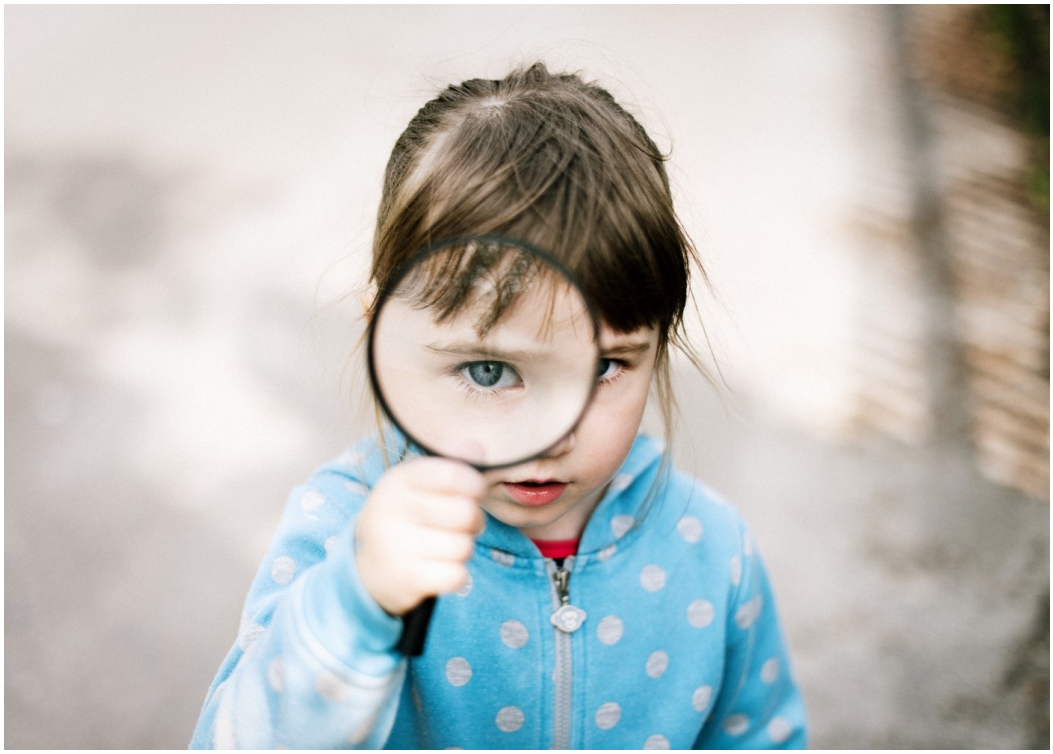

Photo by Elizaveta Dushechkina on Unsplash

Overview of Your Journey

. . .

1 — Setting the Stage

Missing values are a fact of life. If you are a data scientist or a data engineer and receives data, then missing values abound. How you should deal with missing values is highly context-dependent:

- Maybe remove all the rows with missing values?
- Maybe drop an entire feature that has too many missing values?
- Maybe fill in the missing values in a clever way?

The first step should always be to understand what is missing and why it is missing. To start this discovery, there is nothing better than to obtain a good visualization of the missing values! Which of the two options below are easier to comprehend?

```
0   survived     891 non-null    int64
1   pclass       891 non-null    int64
2   sex          891 non-null    object
3   age          714 non-null    float64
4   sibsp        891 non-null    int64
5   parch        891 non-null    int64
6   fare         891 non-null    float64
7   embarked     889 non-null    object
8   class        891 non-null    category
9   who          891 non-null    object
10  adult_male   891 non-null    bool
11  deck         203 non-null    category
12  embark_town  889 non-null    object
13  alive        891 non-null    object
14  alone        891 non-null    bool
```


Bar chart

It's definitely the bar chart, right? 😊

Both options give you information about the missing values in the famous **Titanic dataset**. By a single look at the bar chart, you can see that there are two features (`age` and `deck`) where you are missing a serious amount of data.

In this blog post, I will show you how to work with the Python library missingno. This library gives you a few utility functions that plot the missing values of a pandas dataframe. If you are more of a visual learner, then I have also made a video on the topic

😀

. . .

2 — What is Missingno?

Missingno is a Python library that helps you to visualize missing values in a pandas dataframe. The authors of the library describe missingno in the following way:

> Messy datasets? Missing values? `missingno` provides a small toolset of flexible and easy-to-use missing data visualizations and utilities that allows you to get a quick visual summary of the completeness (or lack thereof) of your dataset. — Missingno Documentation

In this blog post, you will use missingno to understand the missing values in the famous Titanic dataset. The dataset comes preinstalled with the library seaborn, so there is no need to download it separately.

First of all, let's install missingno. I will be using Anaconda, and have hence installed missingno with the simple command:

```
conda install -c conda-forge missingno
```

If you are using PIP, then you can use the command:

```
pip install missingno
```

Since I am using Jupyter Notebooks through Anaconda, I already have pandas and seaborn installed. Make sure you have these installed if you want to follow the code in

this blog post ☺

. . .

3 — Loading the Data

You should start by importing the packages:

```
# Package imports
import seaborn as sns
import pandas as pd
import missingno as msno
%matplotlib inline
```

Importing missingno with the alias `msno` is the recommended way.

Now you can use seaborn to import the Titanic dataset. This dataset comes preinstalled with seaborn, and you can simply run the command:

```
# Load the Titanic data set
titanic = sns.load_dataset("titanic")
```

Now the Titanic dataset is stored in the pandas dataframe `titanic`.

It is difficult to visualize the missing values with pandas. The only thing you can really do is to use the pandas method `.info()` to get a summary of the missing values:

```
titanic.info()

Output:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

The method `.info()` is great for checking out the data types of the different features. However, it is not great for getting a visual picture of what is missing for the different features. You will use missingno for this 😍

. . .

4 — Bar Charts

The most basic plot for visualizing missing values is the **bar chart**. To get this, you can simply use the function `bar` in the missingno library:

```
# Gives a bar chart of the missing values
msno.bar(titanic)
```

This displays the image:

Bar chart

Here you can immediately see that the `age` and `deck` features are seriously missing values. A closer look also reveals that the features `embarked` and `embark_town` are missing two values each.

How you should deal with missing values depends on the context. In this setting, it should be possible to fill in the features `age`, `embarked`, and `embark_town` with appropriate values. However, for the `deck` feature, there is so much missing that I would consider dropping the feature entirely.

Although a bar char is simple, there is no way to see which parts of a feature that is missing. In the next section, I will show you how to see this with missingno's `matrix` function.

. . .

5 — Matrix Plots

Another utility visualization that missingno provides is the **matrix plot**. Simply use the `matrix()` function as follows:

```
# Gives positional information of the missing values
msno.matrix(titanic)
```

This displays the image:

Matrix plot

From the matrix plot, you can see where the missing values are located. For the Titanic dataset, the missing values are located all over the place. However, for other datasets (such as time-series), the missing data is often bundled together (due to e.g. server crashes).

The matrix plot reaffirms our initial assumption that it will be hard to save anything regarding the `deck` features ☹

. . .

6 — Heatmaps

A final visualization you can use is the **heatmap**. This is slightly more complicated than the bar chart and the matrix plot. However, it can sometimes reveal interesting connections between missing values of different features.

To get a heatmap, you can simply use the function `heatmap()` in the missingno library:

```
# Gives a heatmap of how missing values are related
msno.heatmap(titanic)
```

This displays the image:

Heatmap

First of all, notice that there are only four features present in the heatmap. This is because there are only four features that are missing values. All the other features are discarded from the plot.

To understand the heatmap, look at the value that corresponds to `embarked` and `embark_town`. The value is 1. This means that there is a perfect correspondence between missing values in `embarked` and missing values in `embark_town` . You can also see this from the matrix plot you made before.

The values in the heatmap range between -1 and 1. A value of -1 indicates a negative correspondence: A missing value in *feature A* implies that there is not a missing value in *feature B*.

Finally, a value of 0 indicates that there is no obvious correspondence between missing values in *feature A* and missing values in *feature B*. This is (more or less) the case for all the remaining features.

For the Titanic dataset, the heatmap reveals that there is no obvious correspondence between missing values in the `age` feature and missing values in the `deck` feature.

. . .

7 —What have you Learned?

From the visualizations you have done, the following conclusions can be drawn.

- **Bar Chart** — The Titanic dataset is mostly missing values from the features `age` and `deck`.
- **Matrix Plot** — The missing values in `age` and `deck` are spread out all over the rows.
- **Heatmap** — There is no strong correlation between missing values in the `age` and `deck` features.

This gives you a lot more intuition than you started with. Visualizing the missing data is just the first step in a long process. You have far to go, but at least now you have started the journey

Photo by Daniel Cheung on Unsplash

. . .

8 — Wrapping Up

If you need to learn more about missingno, then check out the missingno Github or my video on missingno.

Like my writing? Check out my blog posts

- Modernize Your Sinful Python Code with Beautiful Type Hints
- A Quick Guide to Symbolic Mathematics with SymPy
- 5 Awesome NumPy Functions That Can Save You in a Pinch
- 5 Expert Tips to Skyrocket Your Dictionary Skills in Python

for more Python content. If you are interested in data science, programming, or anything in between, then feel free to add me on LinkedIn and say hi 👋

Missing Values | Python | Pandas | Visualization