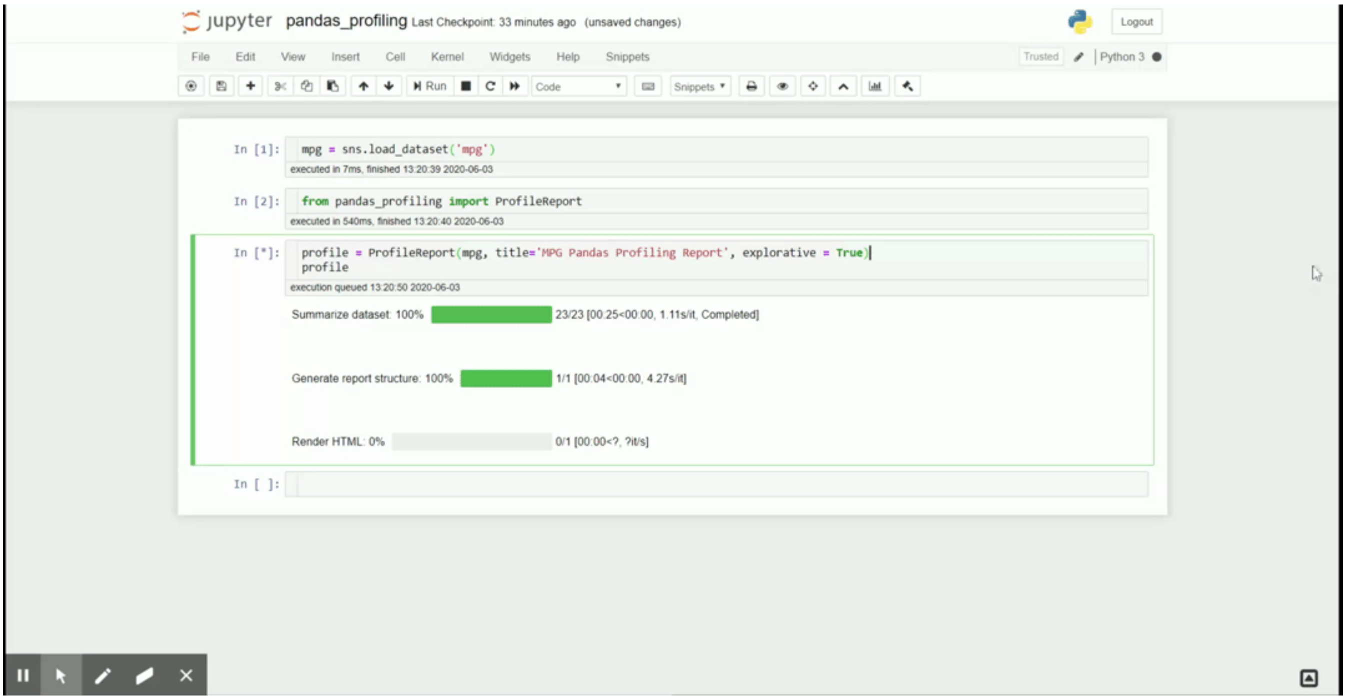
Published in Towards Data Science

Cornellius Yudha Wijaya Follow
Jun 3, 2020 · 5 min read · · Listen

Fantastic Pandas Data Frame Report with Pandas Profiling

Enhance your basic reporting to the next level

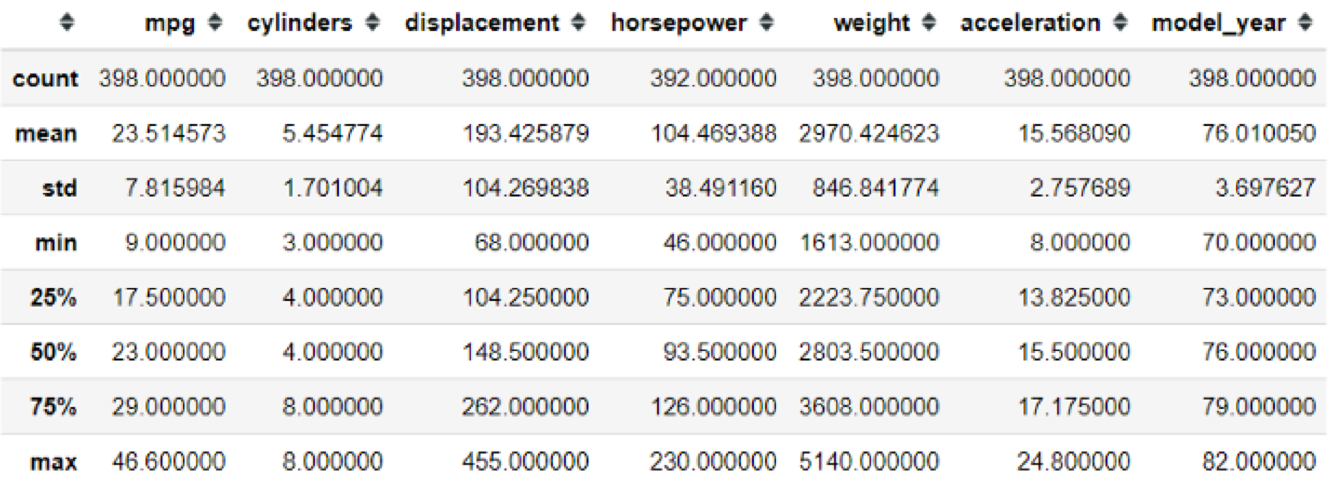

jupyter pandas_profiling Last Checkpoint: 33 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help Snippets
Trusted
Python 3
Run
Code
Snippets

```
In [1]: mpg = sns.load_dataset('mpg')
executed in 7ms, finished 13:20:39 2020-06-03

In [2]: from pandas_profiling import ProfileReport
executed in 540ms, finished 13:20:40 2020-06-03

In [*]: profile = ProfileReport(mpg, title='MPG Pandas Profiling Report', explorative = True)
        profile
execution queued 13:20:50 2020-06-03

Summarize dataset: 100%            23/23 [00:25<00:00, 1.11s/it, Completed]

Generate report structure: 100%    1/1 [00:04<00:00, 4.27s/it]

Render HTML: 0%                    0/1 [00:00<?, ?it/s]

In [ ]:
```

Source: Created by Author

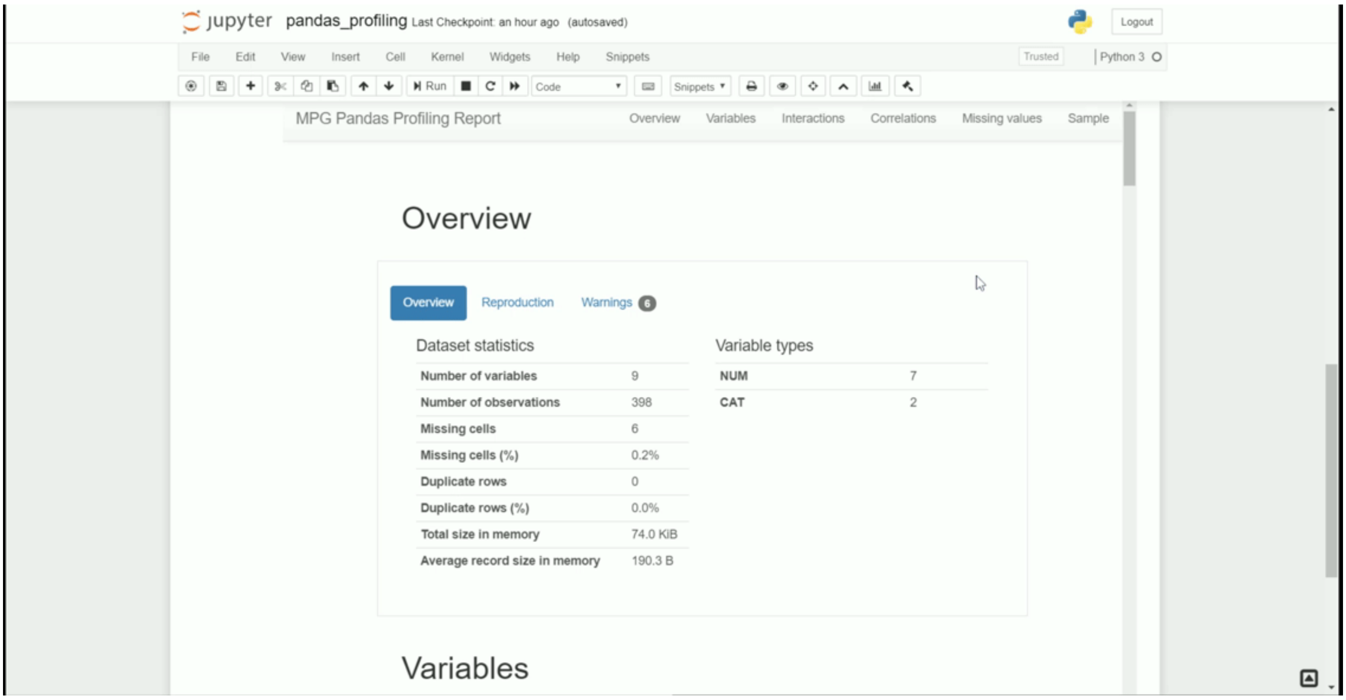
As a Data Scientist, we would explore data for our everyday work. For Pythonist, using the Pandas module is a must. While compelling, sometimes we find the report is just too basic. Let me show it by an example below.

```
#Loading dataset
mpg = sns.load_dataset('mpg')
mpg.describe()
```



	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year
count	398.000000	398.000000	398.000000	392.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	104.469388	2970.424623	15.568090	76.010050
std	7.815984	1.701004	104.269838	38.491160	846.841774	2.757689	3.697627
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000
25%	17.500000	4.000000	104.250000	75.000000	2223.750000	13.825000	73.000000
50%	23.000000	4.000000	148.500000	93.500000	2803.500000	15.500000	76.000000
75%	29.000000	8.000000	262.000000	126.000000	3608.000000	17.175000	79.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000

We could produce the fundamental statistic using `.describe()` attribute, but instead of a basic report like the sample above, we could have our report way more attractive like below.



Overview

Dataset statistics

Number of variables	9
Number of observations	398
Missing cells	6
Missing cells (%)	0.2%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	74.0 KiB
Average record size in memory	190.3 B

Variable types

NUM	7
CAT	2

Variables

Just look at how different the report becomes. It makes our daily exploration way easier. Furthermore, you could save the report into HTML and share it with

Pandas Profiling

We could create a fantastic report like above with the help of Pandas Profiling module. This module is the best to work in the Jupyter environment so that this article would cover the report generated in the Jupyter Notebook. Now, to use this module, we need to install the module.

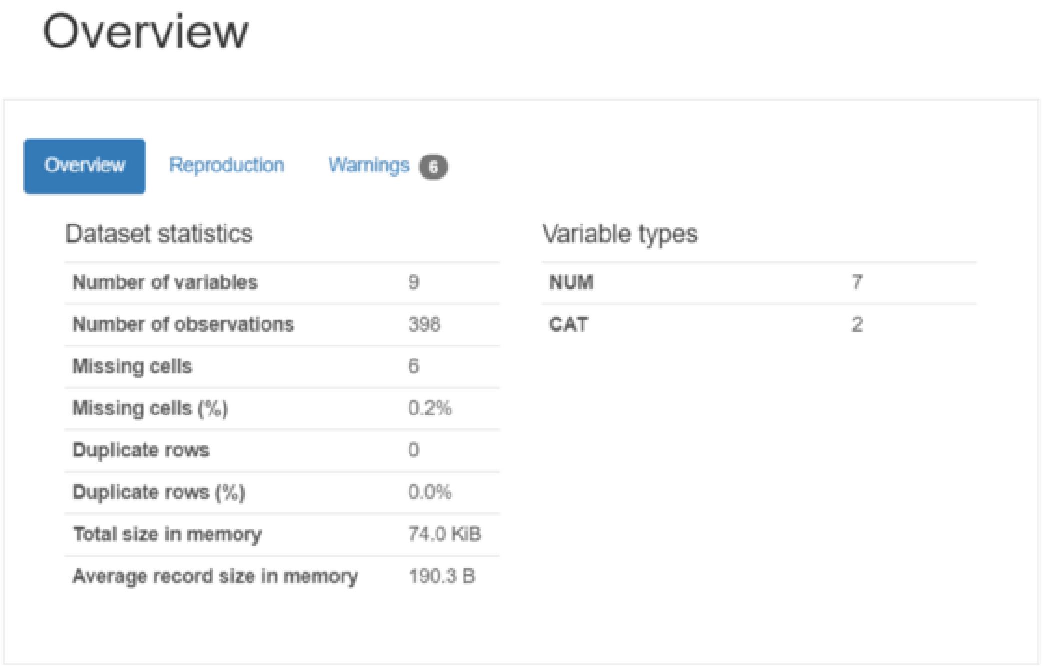
```
#Installing via pip
pip install -U pandas-profiling[notebook]

#Enable the widget extension in Jupyter
jupyter nbextension enable --py widgetsnbextension

#or if you prefer via Conda
conda env create -n pandas-profiling
conda activate pandas-profiling
conda install -c conda-forge pandas-profiling

#or if you prefer installing directly from the source
pip install https://github.com/pandas-profiling/pandas-
profiling/archive/master.zip

#in any case, if the code raise an error, it probably need
permission from user. To do that, add --user in the end of the line.
```

With that, we are ready to generate the report. We would use the Pandas Profiling function, just like the code below.

```
#Importing the function
from pandas_profiling import ProfileReport

#Generate the report. We would use the mpg dataset as sample, title
parameter for naming our report, and explorative parameter set to
True for Deeper exploration.

profile = ProfileReport(mpg, title='MPG Pandas Profiling Report',
explorative = True)

profile
```

After waiting a while, we would end up with an HTML report like below.

Complete set of
report



Overview

Overview Reproduction Warnings 6

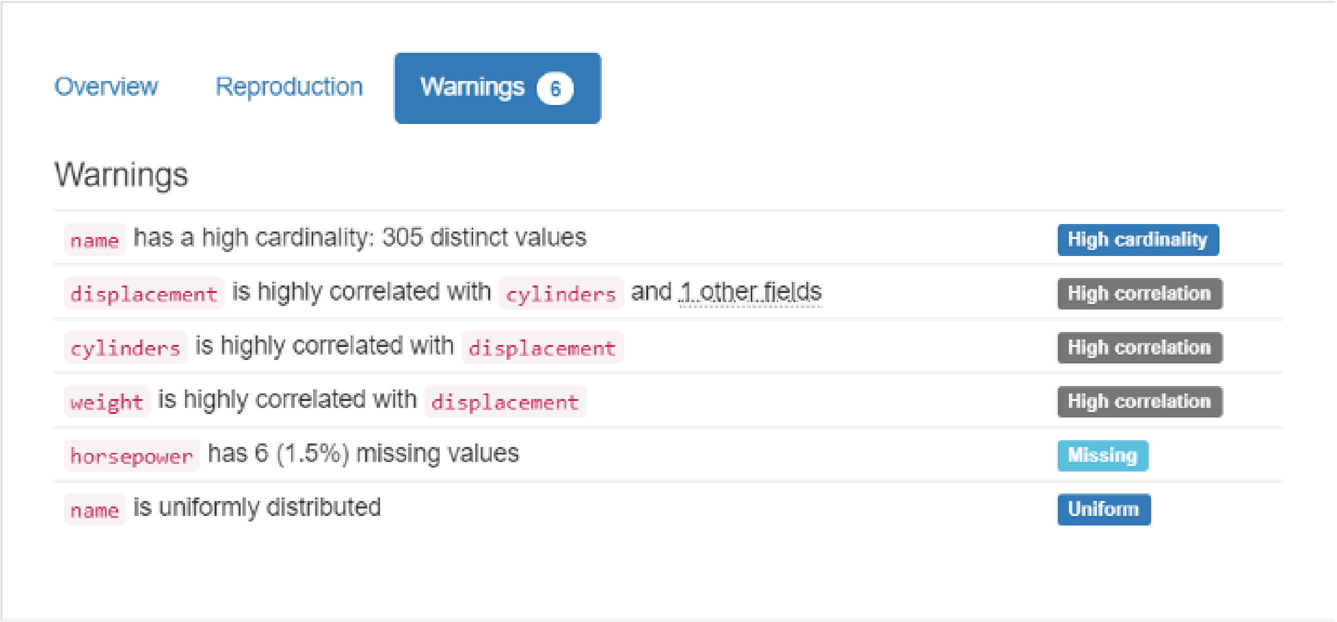
Dataset statistics

Number of variables	9
Number of observations	398
Missing cells	6
Missing cells (%)	0.2%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	74.0 KiB
Average record size in memory	190.3 B

Variable types

NUM	7
CAT	2

In the first part, we would get the overview information of our Data Frame. It is similar if we use the `.info()` attribute from the Pandas Data Frame object, but the Pandas Profiling offer more information. For example, the Warnings section.

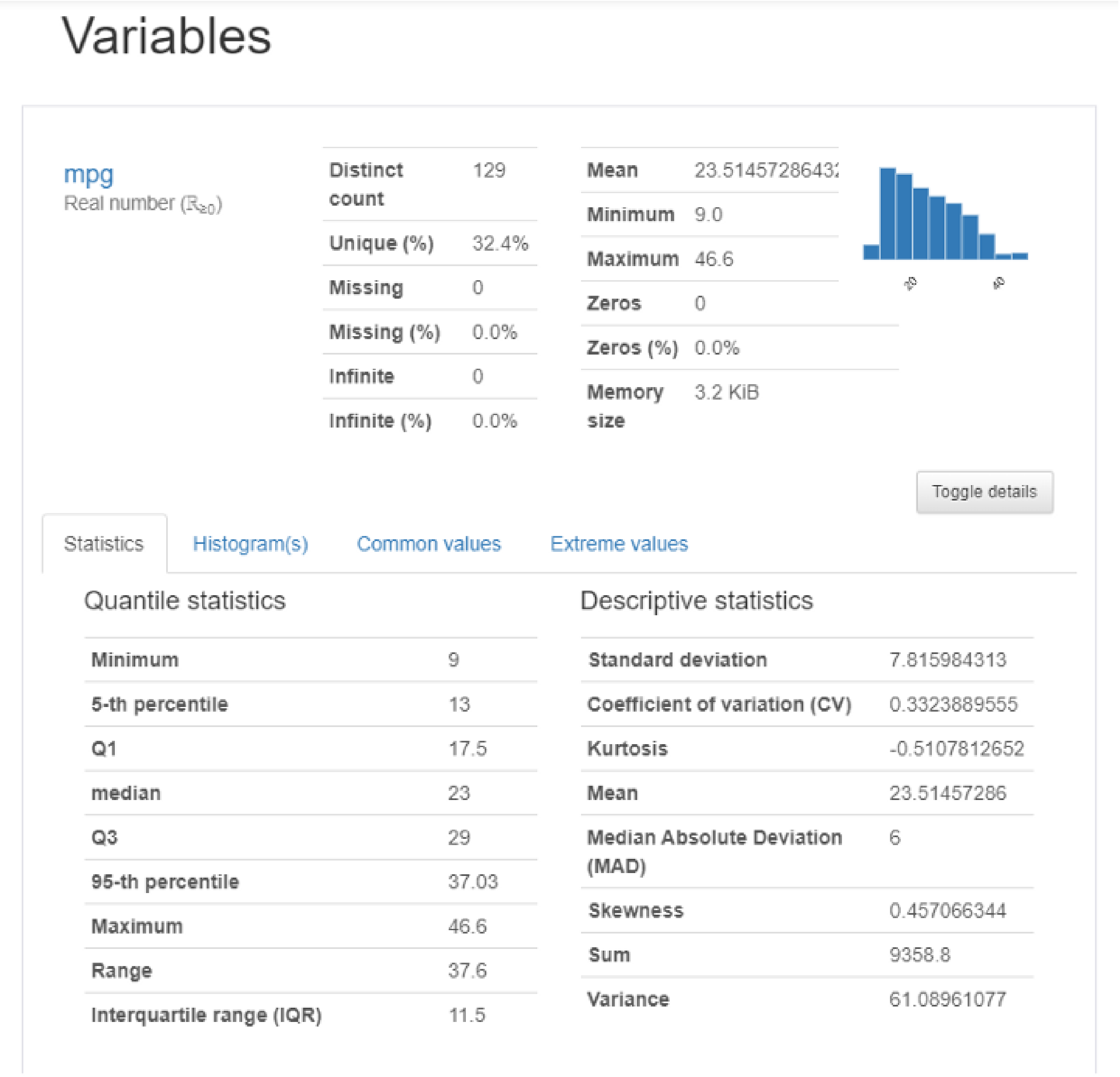


Overview Reproduction Warnings 6

Warnings

`name` has a high cardinality: 305 distinct values	High cardinality
`displacement` is highly correlated with `cylinders` and 1 other fields	High correlation
`cylinders` is highly correlated with `displacement`	High correlation
`weight` is highly correlated with `displacement`	High correlation
`horsepower` has 6 (1.5%) missing values	Missing
`name` is uniformly distributed	Uniform

What is excellent about the Warnings section is that the information given are not just basic info such as missing data, but more complex one such as high correlation, high cardinality, etc. We could modify how high it is to consider what is ‘High numerical variable.



Variables

mpg
Real number (ℝ≥0)

Distinct count	129
Unique (%)	32.4%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	23.51457286432
Minimum	9.0
Maximum	46.6
Zeros	0
Zeros (%)	0.0%
Memory size	3.2 KiB

Quantile statistics

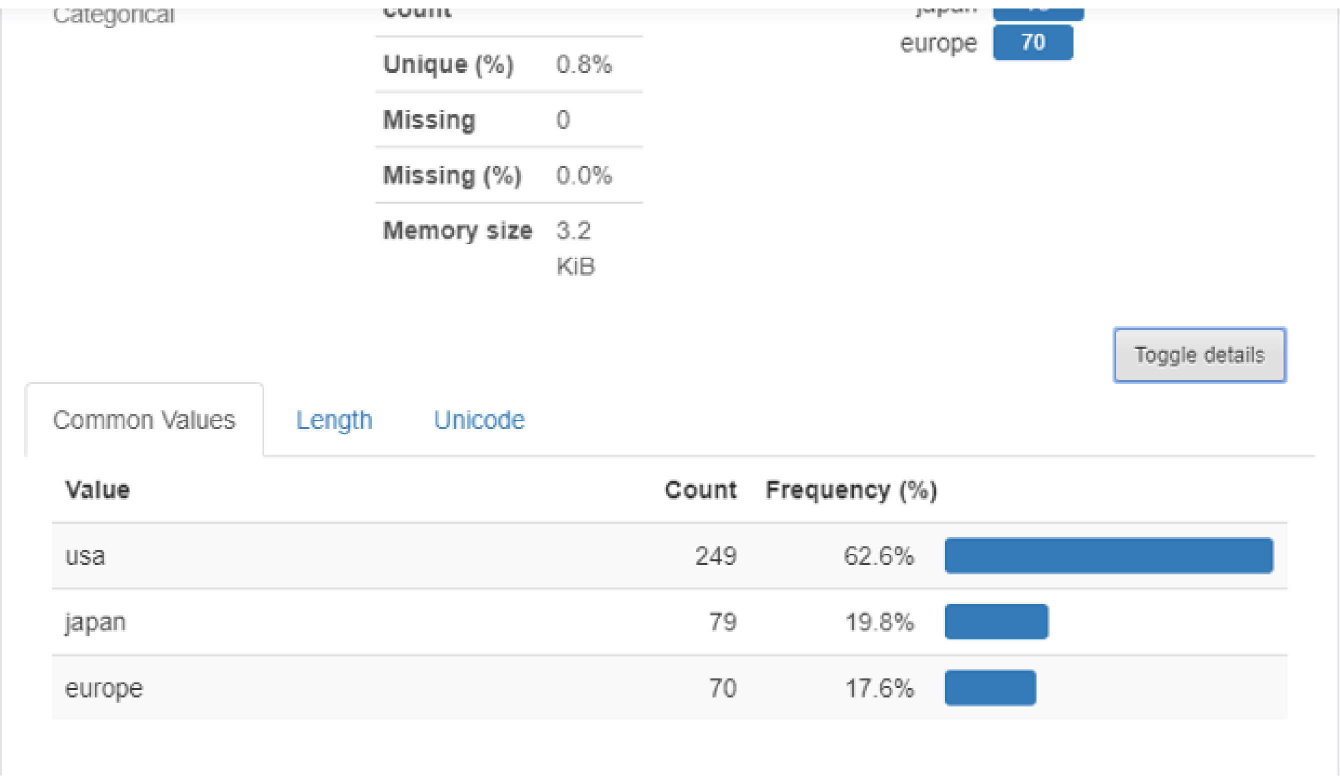
Minimum	9
5-th percentile	13
Q1	17.5
median	23
Q3	29
95-th percentile	37.03
Maximum	46.6
Range	37.6
Interquartile range (IQR)	11.5

Descriptive statistics

Standard deviation	7.815984313
Coefficient of variation (CV)	0.3323889555
Kurtosis	-0.5107812652
Mean	23.51457286
Median Absolute Deviation (MAD)	6
Skewness	0.457066344
Sum	9358.8
Variance	61.08961077

We could see that for each variable, we are served with complete statistic information. Furthermore, there are sections where we could get information for the most common values and extreme values.

How about Categorical variable? Let me show you in the image below.

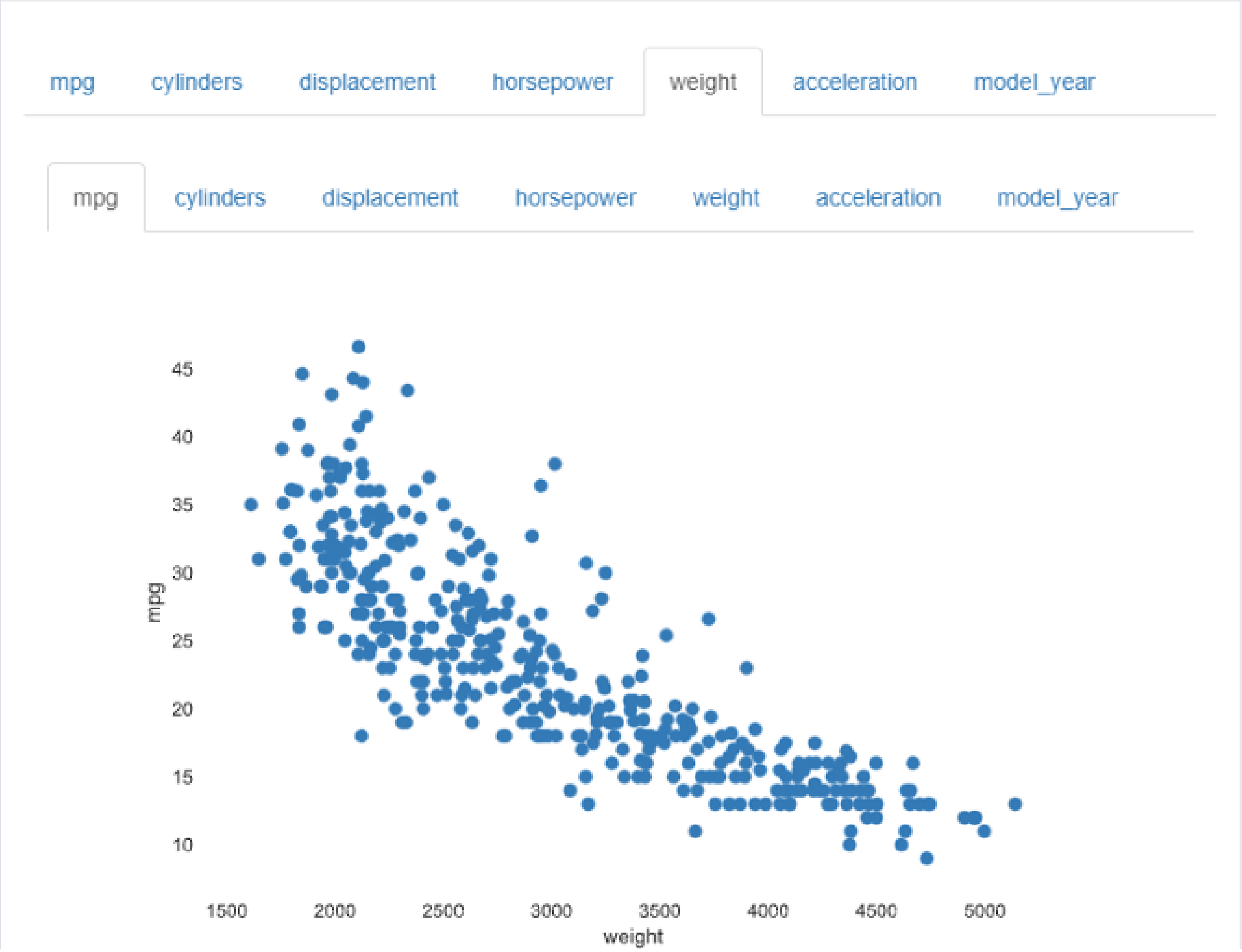


Categorical

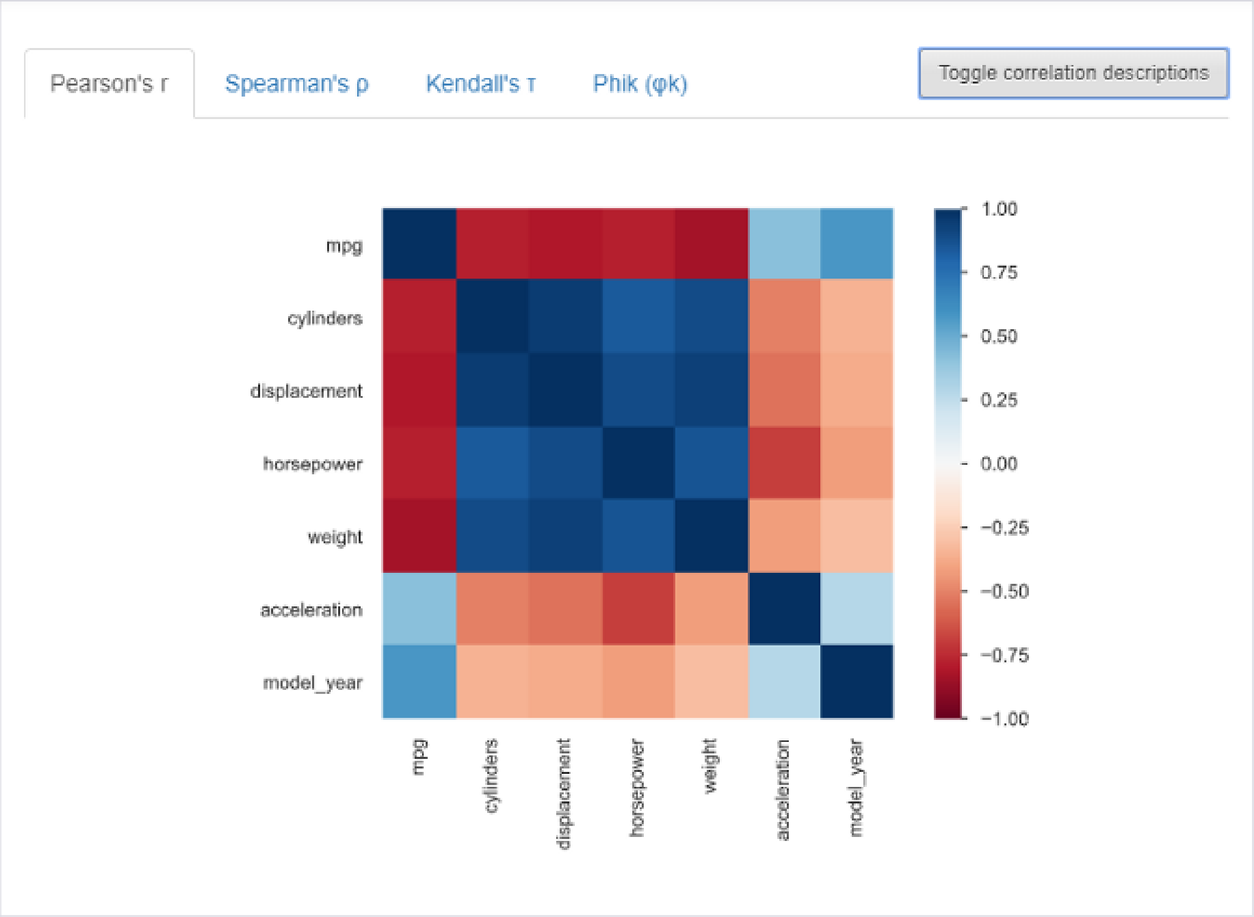
Unique (%)	0.8%
Missing	0
Missing (%)	0.0%
Memory size	3.2 KiB

Value	Count	Frequency (%)
usa	249	62.6%
japan	79	19.8%
europe	70	17.6%

Just like the numerical variable, we acquired complete information about the variable. Scroll down even further; we would arrive in the Interactions section. This is the section where we could get a Scatter Plot between two numerical variables.

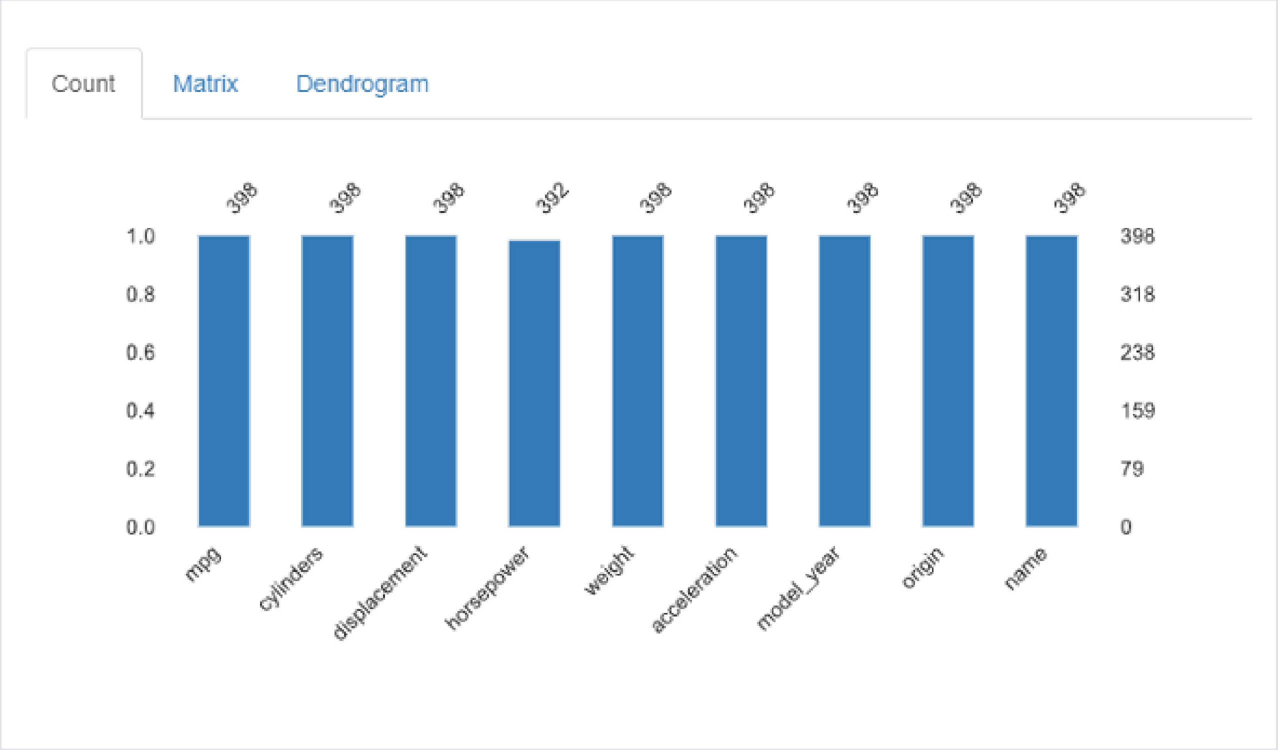


And just below it is the Correlations section.



This section is showing the correlation values between numerical variables in the form of a heatmap. Only four correlation calculation available here and if you need the correlation descriptions, you could click the “Toggle correlation descriptions button”.

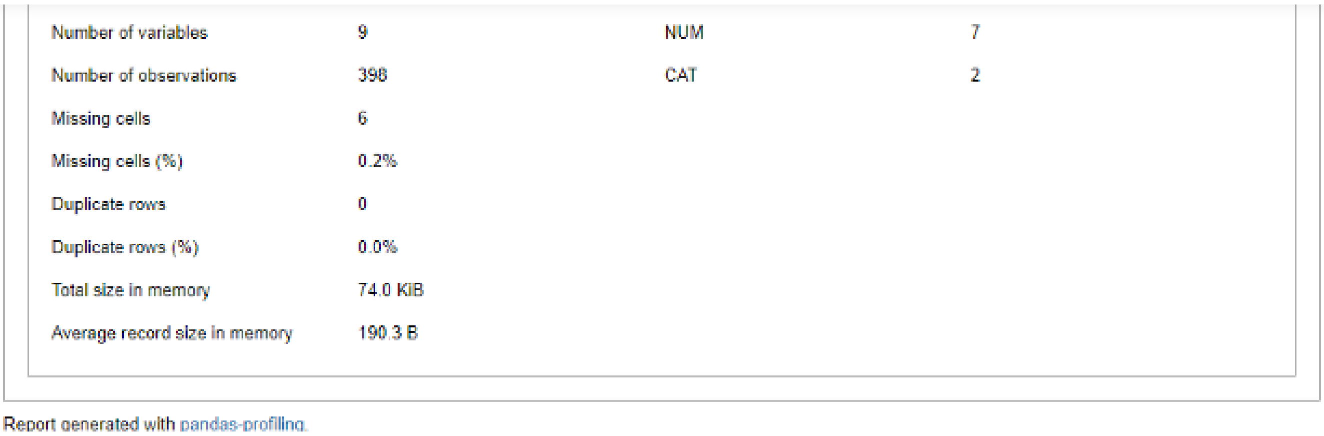
There is also a section dedicated to the Missing values, just like the example below.



And the last section would only show the data samples — nothing interesting there.

If you need a more simple way to show the report, we could use the following code to transform the report.

```
profile.to_widgets()
```



Number of variables	9	NUM	7
Number of observations	398	CAT	2
Missing cells	6		
Missing cells (%)	0.2%		
Duplicate rows	0		
Duplicate rows (%)	0.0%		
Total size in memory	74.0 KiB		
Average record size in memory	190.3 B		

Report generated with pandas-profiling.

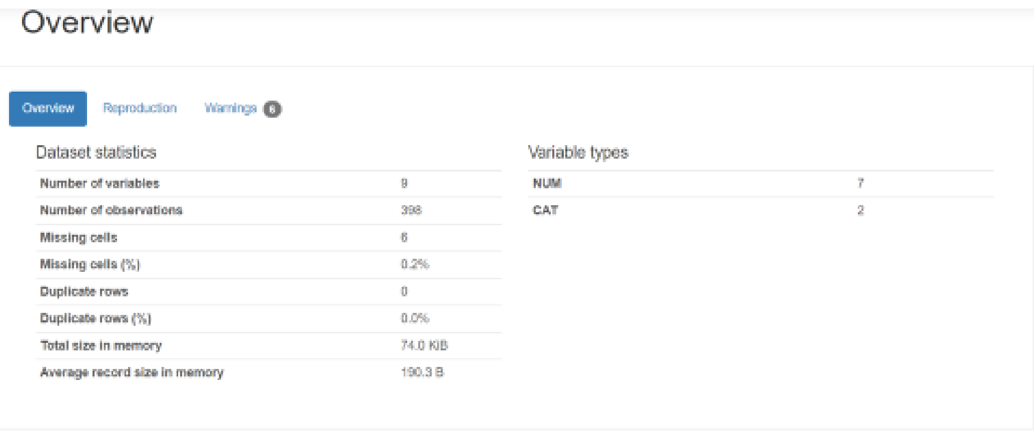
With one line of code, we get the same information from what I showed you above. The only differences are just the UI becomes more straightforward. The information, although, would still the same.

Lastly, if you want to export the report into an external HTML file, we could use the following code.

```
profile.to_file('your_report_name.html')
```

□ your_report_name.html

You could find the HTML file in the same folder with your Jupyter Notebook. If you open the file, it would automatically open on your default browser with beautiful UI similar to the one in our Jupyter Notebook.



Overview

Dataset statistics

Number of variables	9
Number of observations	398
Missing cells	6
Missing cells (%)	0.2%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	74.0 KiB
Average record size in memory	190.3 B

Variable types

NUM	7
CAT	2

Conclusion

I have shown you how to transform our basic report in the Pandas Data Frame to a more interactive form by using the Pandas Profiling Module. I hope it helps.

If you enjoy my content and want to get more in-depth knowledge regarding data or just daily life as a Data Scientist, please consider subscribing to my newsletter here.

> If you are not subscribed as a Medium Member, please consider subscribing through my referral.

Enjoy the read? Reward the writer. Beta

Your tip will go to Cornellius Yudha Wijaya through a third-party platform of their choice, letting them know you appreciate their story.