eda-report

Release 2.8.1

Abwao

Oct 13, 2023

CONTENTS

Speed up the exploratory data analysis and reporting process. Automatically analyze a dataset, and get:

CHAPTER

ONE

1. STATISTICAL PROPERTIES

Descriptive statistics, bivariate analysis, tests for normality and more:

```
>>> eda_report.summarize(range(50))

Name: var_1
Type: numeric
Non-null Observations: 50
Unique Values: 50 -> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, [...]
Missing Values: None

                  Summary Statistics
                  ------------------
        Average:                      24.5000
        Standard Deviation:           14.5774
        Minimum:                       0.0000
        Lower Quartile:               12.2500
        Median:                       24.5000
        Upper Quartile:               36.7500
        Maximum:                      49.0000
        Skewness:                      0.0000
        Kurtosis:                     -1.2000

                  Tests for Normality
                  -------------------
                               p-value Conclusion at  = 0.05
D'Agostino's K-squared test  0.0015981  Unlikely to be normal
Kolmogorov-Smirnov test      0.0000000  Unlikely to be normal
Shapiro-Wilk test            0.0580895        Possibly normal
```

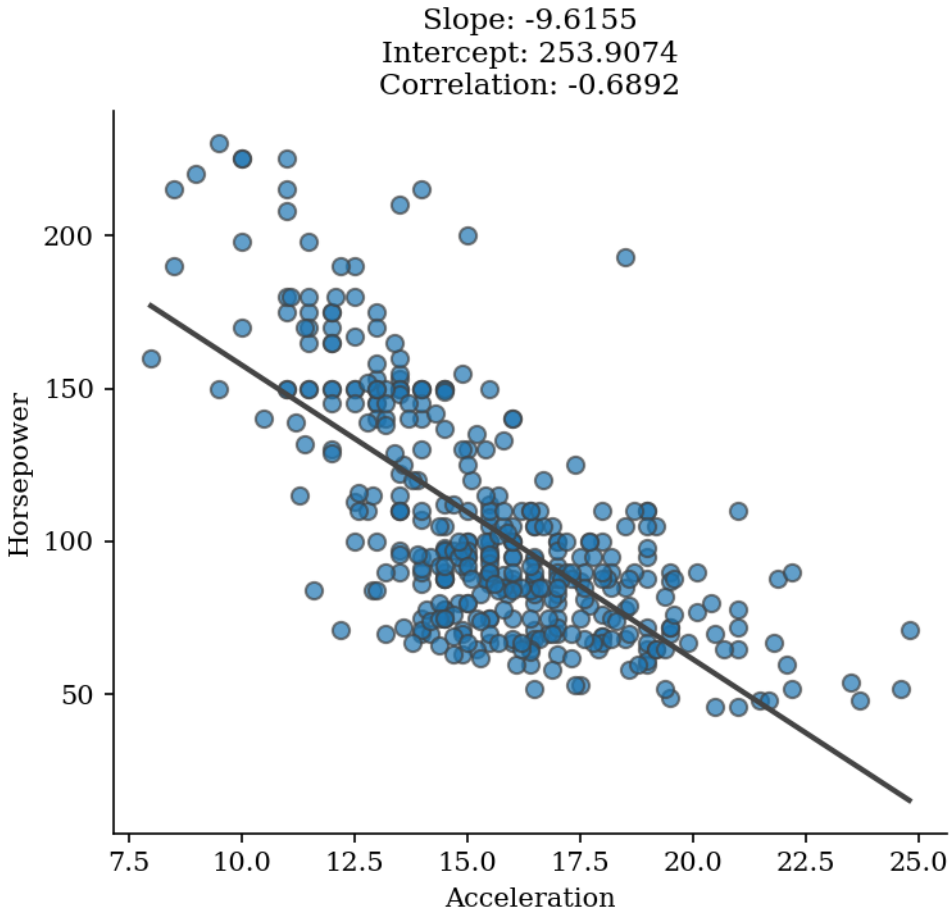
CHAPTER

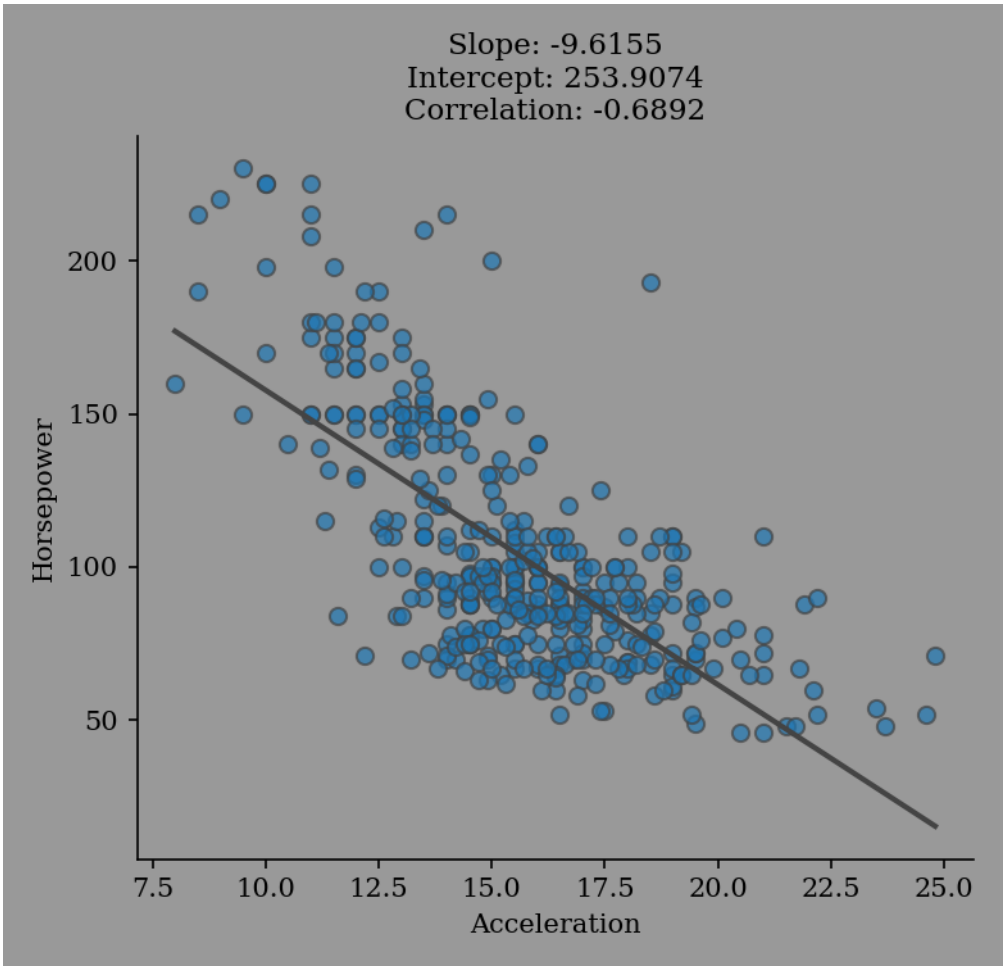
TWO

2. REVEALING VISUALIZATIONS

- *Box-plots*, *kde-plots*, *normal-probability-plots*, *scatter-plots* and a *correlation bar-chart* for numeric variables.
- *Bar-plots* for categorical variables.

```
>>> import eda_report.plotting as ep
>>> ax = ep.regression_plot(mpg_data["acceleration"], mpg_data["horsepower"],
...                         labels=("Acceleration", "Horsepower"))
>>> ax.figure.savefig("regression-plot.png")
```




Slope: -9.6155
Intercept: 253.9074
Correlation: -0.6892
Horsepower
200
150
100
50
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
Acceleration

CHAPTER

THREE

3. A REPORT IN *WORD* (.DOCX) FORMAT

An exploratory data analysis report document complete with variable descriptions, summary statistics, statistical plots, contingency tables and more:

```
>>> import eda_report
>>> eda_report.get_word_report(iris_data)
Analyze variables:  100%|| 5/5
Plot variables:     100%|| 5/5
Bivariate analysis: 100%|| 6/6 pairs.
[INFO 16:14:53.648] Done. Results saved as 'eda-report.docx'
<eda_report.document.ReportDocument object at 0x7f196753bd60>
```

3.1 Installation

Important: Only **Python3.9 to 3.11** are currently supported.

Tip: Consider using a virtual environment. Virtual environments are a great way to ensure that you install the right versions of dependencies, while avoiding breaking other Python packages in your system.

You can install `eda-report` from the Python Package Index using `pip`:

```
$ pip install eda-report
```

You can also install the latest stable version right from the GitHub repository using:

```
$ pip install https://github.com/tim-abwao/eda-report/archive/main.tar.gz
```

1. Univariate Analysis

1.1. Petal_Length

Petal_length is a numeric variable with 43 unique values. None of its values are missing.

Summary Statistics

Number of observations	150
Average	3.758
Standard Deviation	1.7653
Minimum	1
Lower Quartile	1.6
Median	4.35
Upper Quartile	5.1
Maximum	6.9
Skewness	-0.2749
Kurtosis	-1.4021

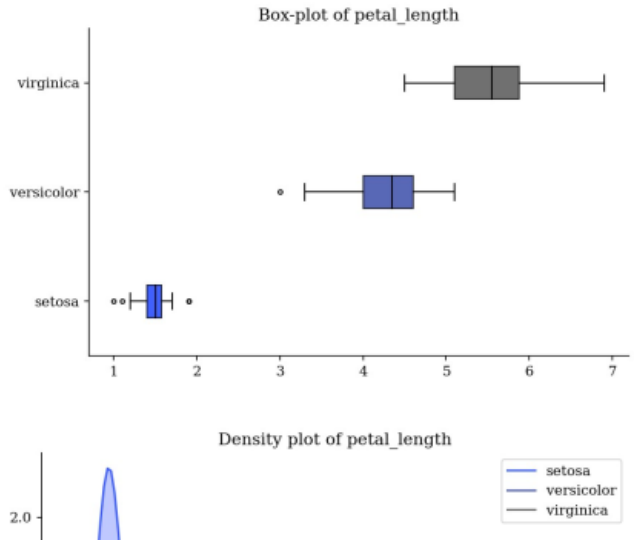


Fig. 1: A report generated from the *iris dataset*.

3.2 Quickstart

3.2.1 Using the Graphical User Interface

The command `eda-report` launches a graphical window to help select a *csv* or *excel* file to analyze:

```
$ eda-report
```

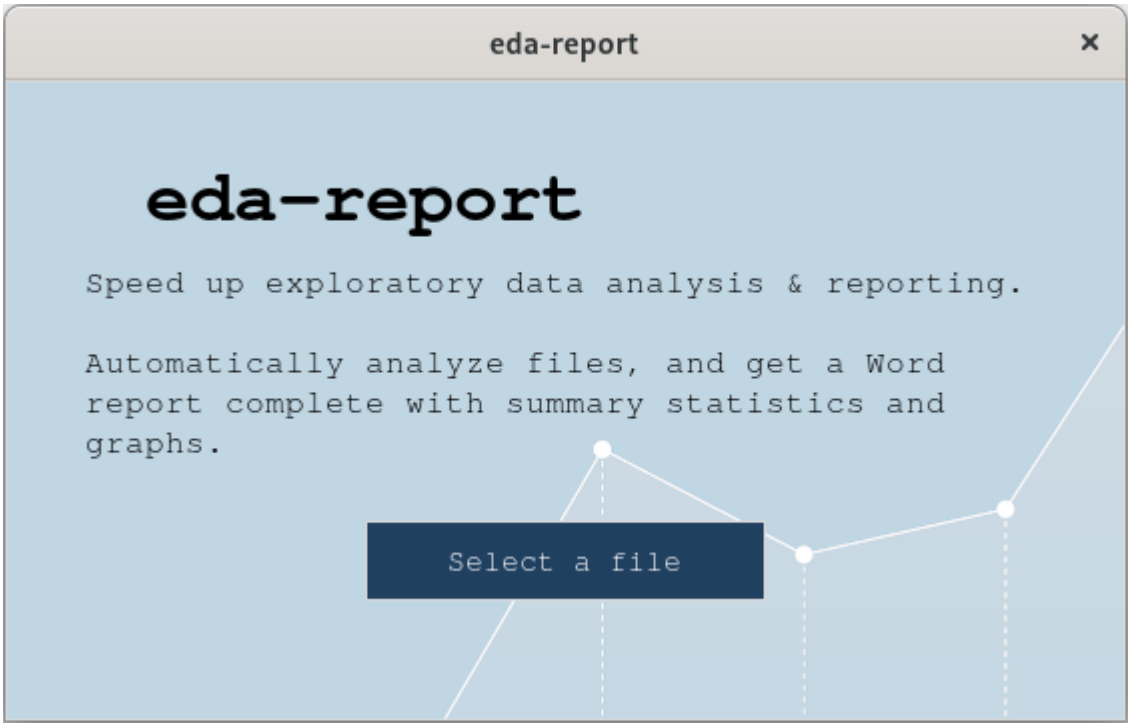


Fig. 2: A `tkinter`-based graphical user interface to the application

You will be prompted to enter your desired *title*, *groupby/target variable*, *graph color* & *output file-name*. Afterwards, a report is generated, as specified, from the contents of the selected file.

Hint: For help with *Tk* - related issues, consider visiting TkDocs.

3.2.2 Using the Command Line Interface

You can specify an input file and an output file-name:

```
$ eda-report -i data.csv -o some_name.docx
```

```
$ eda-report -h
usage: eda-report [-h] [-i INFILE] [-o OUTFILE] [-t TITLE] [-c COLOR]
                  [-g GROUPBY]

Automatically analyze data and generate reports. A graphical user interface
```

(continues on next page)

(continued from previous page)

```
will be launched if none of the optional arguments is specified.

optional arguments:
  -h, --help            show this help message and exit
  -i INFILE, --infile INFILE
                        A .csv or .xlsx file to analyze.
  -o OUTFILE, --outfile OUTFILE
                        The output name for analysis results (default: eda-
                        report.docx)
  -t TITLE, --title TITLE
                        The top level heading for the report (default:
                        Exploratory Data Analysis Report)
  -c COLOR, --color COLOR
                        The color to apply to graphs (default: cyan)
  -g GROUPBY, -T GROUPBY, --groupby GROUPBY, --target GROUPBY
                        The variable to use for grouping plotted values. An
                        integer value is treated as a column index, whereas a
                        string is treated as a column label.
```

3.2.3 From an Interactive Session

You can use the *get_word_report()* function to generate reports:

```
>>> import eda_report
>>> eda_report.get_word_report(iris_data)
Analyze variables:  100%|| 5/5
Plot variables:     100%|| 5/5
Bivariate analysis: 100%|| 6/6 pairs.
[INFO 16:14:53.648] Done. Results saved as 'eda-report.docx'
<eda_report.document.ReportDocument object at 0x7f196753bd60>
```

You can use the *summarize()* function to analyze datasets:

```
>>> eda_report.summarize(range(50))

Name: var_1
Type: numeric
Non-null Observations: 50
Unique Values: 50 -> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, [...]
Missing Values: None

                  Summary Statistics
                  ------------------
        Average:                     24.5000
        Standard Deviation:          14.5774
        Minimum:                      0.0000
        Lower Quartile:              12.2500
        Median:                      24.5000
        Upper Quartile:              36.7500
        Maximum:                     49.0000
        Skewness:                     0.0000
```

(continues on next page)

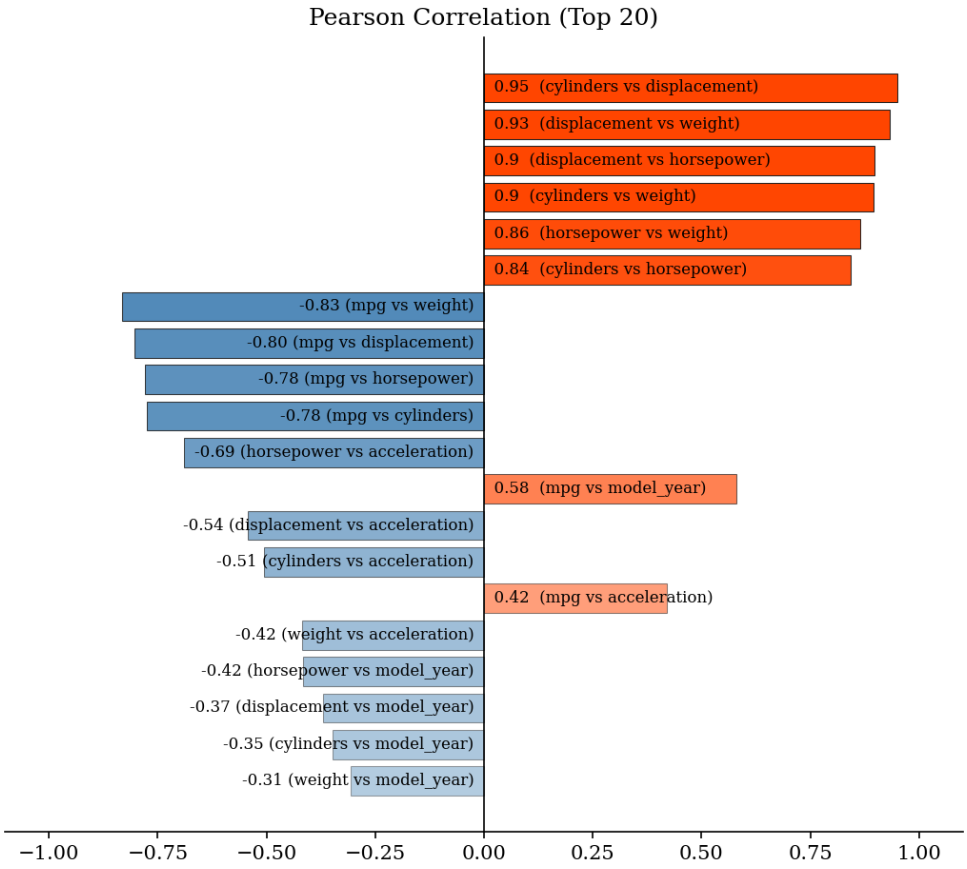
(continued from previous page)

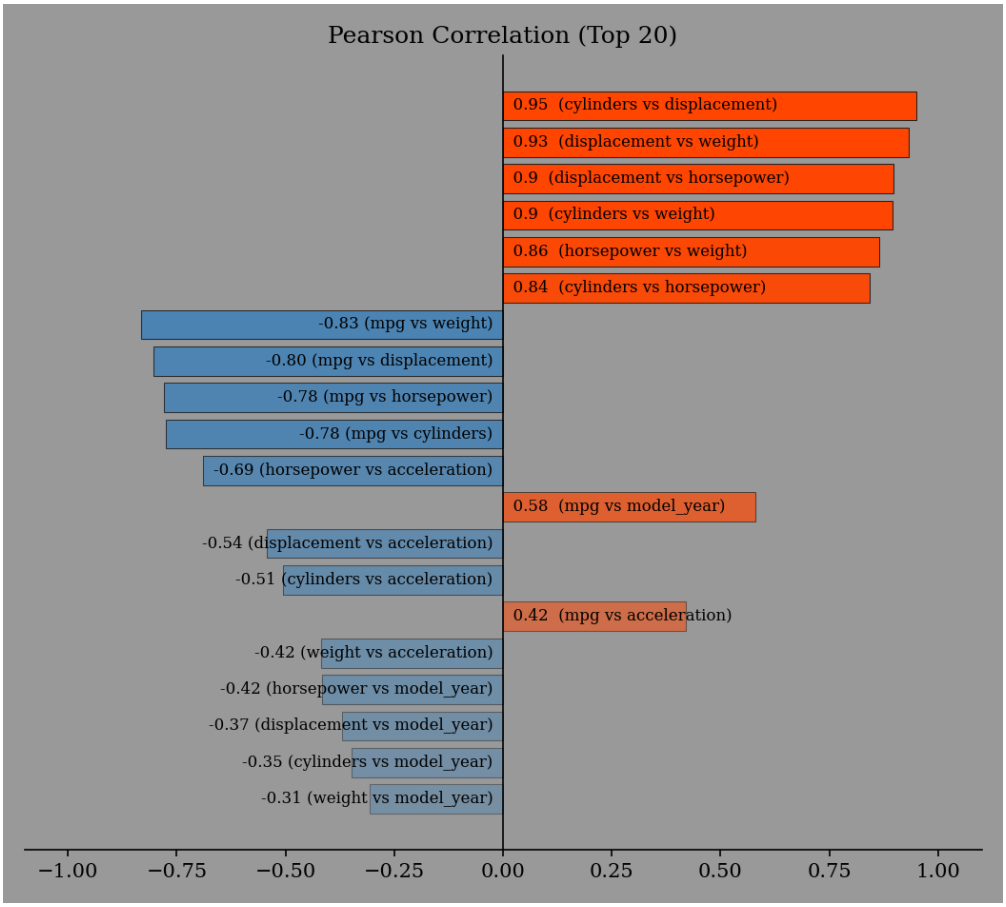
```
       Kurtosis:                    -1.2000

                 Tests for Normality
                 -------------------
                              p-value Conclusion at  = 0.05
D'Agostino's K-squared test  0.0015981  Unlikely to be normal
Kolmogorov-Smirnov test      0.0000000  Unlikely to be normal
Shapiro-Wilk test            0.0580895        Possibly normal
```

```
>>> eda_report.summarize(iris_data)

                 Summary Statistics for Numeric features (4)
                 -------------------------------------------
               count     avg  stddev  min  25%   50%  75%  max  skewness  kurtosis
  sepal_length   150  5.8433  0.8281  4.3  5.1  5.80  6.4  7.9    0.3149   -0.5521
  sepal_width    150  3.0573  0.4359  2.0  2.8  3.00  3.3  4.4    0.3190    0.2282
  petal_length   150  3.7580  1.7653  1.0  1.6  4.35  5.1  6.9   -0.2749   -1.4021
  petal_width    150  1.1993  0.7622  0.1  0.3  1.30  1.8  2.5   -0.1030   -1.3406

               Summary Statistics for Categorical features (1)
               -----------------------------------------------
                  count unique     top freq relative freq
          species   150      3  setosa   50        33.33%


                      Pearson's Correlation (Top 20)
                      ------------------------------
     petal_length & petal_width -> very strong positive correlation (0.96)
    sepal_length & petal_length -> very strong positive correlation (0.87)
     sepal_length & petal_width -> very strong positive correlation (0.82)
     sepal_width & petal_length -> moderate negative correlation (-0.43)
      sepal_width & petal_width -> weak negative correlation (-0.37)
     sepal_length & sepal_width -> very weak negative correlation (-0.12)
```

You can plot several statistical graphs (see *Plotting Examples*):

```
>>> import eda_report.plotting as ep
>>> ax = ep.plot_correlation(mpg_data)
>>> ax.figure.savefig("correlation-plot.png")
```


Pearson Correlation (Top 20)
0.95 (cylinders vs displacement)
0.93 (displacement vs weight)
0.9 (displacement vs horsepower)
0.9 (cylinders vs weight)
0.86 (horsepower vs weight)
0.84 (cylinders vs horsepower)
-0.83 (mpg vs weight)
-0.80 (mpg vs displacement)
-0.78 (mpg vs horsepower)
-0.78 (mpg vs cylinders)
-0.69 (horsepower vs acceleration)
0.58 (mpg vs model_year)
-0.54 (displacement vs acceleration)
-0.51 (cylinders vs acceleration)
0.42 (mpg vs acceleration)
-0.42 (weight vs acceleration)
-0.42 (horsepower vs model_year)
-0.37 (displacement vs model_year)
-0.35 (cylinders vs model_year)
-0.31 (weight vs model_year)
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00



3.3 API Reference

3.3.1 eda_report

eda_report.**get_word_report**(*data: Iterable*, *, *title: str = 'Exploratory Data Analysis Report'*, *graph_color: str = 'cyan'*, *groupby_variable: str | int = None*, *output_filename: str = 'eda-report.docx'*, *table_style: str = 'Table Grid'*) → *ReportDocument*

Analyze *data*, and generate a report document in *Word* (*.docx*) format.

Parameters

- **data** (*Iterable*) – The data to analyze.
- **title** (*str, optional*) – The title to assign the report. Defaults to "Exploratory Data Analysis Report".
- **graph_color** (*str, optional*) – The color to apply to the graphs. Defaults to "cyan".
- **groupby_variable** (*Union[str, int], optional*) – The label/index for the column to use to group values. Defaults to None.
- **output_filename** (*str, optional*) – The name/path to save the report document. Defaults to "eda-report.docx".
- **table_style** (*str, optional*) – The style to apply to the tables created. Defaults to "Table Grid".

Returns
Document object with analysis results.

Return type
ReportDocument

Example

```
>>> import eda_report
>>> eda_report.get_word_report(iris_data)
Analyze variables:  100%|| 5/5
Plot variables:     100%|| 5/5
Bivariate analysis: 100%|| 6/6 pairs.
[INFO 16:14:53.648] Done. Results saved as 'eda-report.docx'
<eda_report.document.ReportDocument object at 0x7f196753bd60>
```

eda_report.**summarize**(*data: Iterable*) → *Variable* | *Dataset*

Get summary statistics for the supplied data.

Parameters
data (*Iterable*) – The data to analyze.

Returns
Analysis results.

Return type
Union[*Variable*, *Dataset*]

Example

```
>>> eda_report.summarize(iris_data)

                Summary Statistics for Numeric features (4)
                -------------------------------------------
               count     avg  stddev  min  25%   50%  75%  max  skewness  kurtosis
  sepal_length   150  5.8433  0.8281  4.3  5.1  5.80  6.4  7.9    0.3149   -0.5521
  sepal_width    150  3.0573  0.4359  2.0  2.8  3.00  3.3  4.4    0.3190    0.2282
  petal_length   150  3.7580  1.7653  1.0  1.6  4.35  5.1  6.9   -0.2749   -1.4021
  petal_width    150  1.1993  0.7622  0.1  0.3  1.30  1.8  2.5   -0.1030   -1.3406

              Summary Statistics for Categorical features (1)
              -----------------------------------------------
                  count unique     top freq relative freq
          species   150      3  setosa   50        33.33%


                     Pearson's Correlation (Top 20)
                     ------------------------------
    petal_length & petal_width -> very strong positive correlation (0.96)
   sepal_length & petal_length -> very strong positive correlation (0.87)
    sepal_length & petal_width -> very strong positive correlation (0.82)
    sepal_width & petal_length -> moderate negative correlation (-0.43)
```

(continues on next page)

(continued from previous page)

```
      sepal_width & petal_width -> weak negative correlation (-0.37)
     sepal_length & sepal_width -> very weak negative correlation (-0.12)
```

3.3.2 eda_report.bivariate

class eda_report.bivariate.**Dataset**(*data: Iterable*)

Analyze two-dimensional datasets to obtain descriptive statistics and correlation information.

Input data is stored as a `pandas.DataFrame` in order to leverage pandas' built-in statistical methods.

Parameters

data (*Iterable*) – The data to analyze.

Example

```
>>> Dataset(iris_data)
                 Summary Statistics for Numeric features (4)
                 -------------------------------------------
               count     avg  stddev  min  25%   50%  75%  max  skewness  kurtosis
  sepal_length   150  5.8433  0.8281  4.3  5.1  5.80  6.4  7.9    0.3149   -0.5521
  sepal_width    150  3.0573  0.4359  2.0  2.8  3.00  3.3  4.4    0.3190    0.2282
  petal_length   150  3.7580  1.7653  1.0  1.6  4.35  5.1  6.9   -0.2749   -1.4021
  petal_width    150  1.1993  0.7622  0.1  0.3  1.30  1.8  2.5   -0.1030   -1.3406

               Summary Statistics for Categorical features (1)
               -----------------------------------------------
                   count unique     top freq relative freq
           species   150      3  setosa   50        33.33%


                      Pearson's Correlation (Top 20)
                      ------------------------------
    petal_length & petal_width -> very strong positive correlation (0.96)
   sepal_length & petal_length -> very strong positive correlation (0.87)
    sepal_length & petal_width -> very strong positive correlation (0.82)
    sepal_width & petal_length -> moderate negative correlation (-0.43)
     sepal_width & petal_width -> weak negative correlation (-0.37)
    sepal_length & sepal_width -> very weak negative correlation (-0.12)
```

3.3.3 eda_report.document

class eda_report.document.**ReportDocument**(*data: Iterable*, *, *title: str = 'Exploratory Data Analysis Report'*, *graph_color: str = 'cyan'*, *groupby_variable: str | int = None*, *output_filename: str = 'eda-report.docx'*, *table_style: str = 'Table Grid'*)

Bases: `_ReportContent`

Creates a report `Document` with analysis results.

The report consists of 3 main sections:

1. An **Overview** of the data and its features.
2. **Univariate Analysis**: Summary statistics and graphs for each feature.
3. **Bivariate Analysis**: Pair-wise comparisons of numerical features.

Parameters

- **data** (*Iterable*) – The data to analyze.
- **title** (*str, optional*) – The title to assign the report. Defaults to "Exploratory Data Analysis Report".
- **graph_color** (*str, optional*) – The color to apply to the graphs. Defaults to "cyan".
- **groupby_variable** (*Union[str, int], optional*) – The column to use to group values. Defaults to None.
- **output_filename** (*str, optional*) – The name/path to save the document to. Defaults to "eda-report.docx".
- **table_style** (*str, optional*) – The style to apply to the tables created. Defaults to "Table Grid".

3.3.4 eda_report.exceptions

exception eda_report.exceptions.**EmptyDataError**(*message: str*)

Bases: *InputError*

Exception raised when an iterable input object has length zero or has no more items to yield.

exception eda_report.exceptions.**Error**

Bases: Exception

The base class for exceptions in this package.

exception eda_report.exceptions.**GroupbyVariableError**(*message: str*)

Bases: *InputError*

Exception raised when the specified group-by variable is invalid.

exception eda_report.exceptions.**InputError**(*message: str*)

Bases: *Error*

Exception raised when a given input object is *not of the expected type* or is otherwise *invalid*.

In most cases, an attempt is made to cast the erroneous input into the proper type, and this *Exception* is raised if it fails.

Parameters

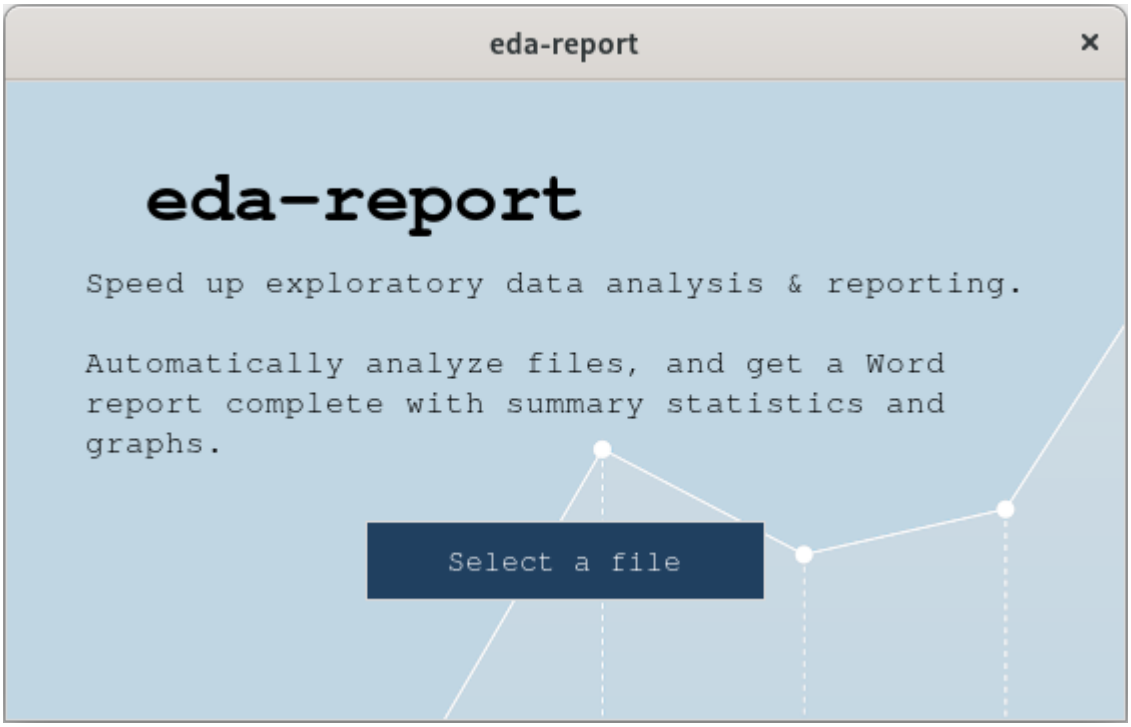
message (*str*) – A brief description of the mishap detected.

3.3.5 eda_report.gui

class eda_report.gui.**EDAGUI**(*master=None*, ***kwargs*)

Bases: `Frame`

The blueprint for the `tkinter` - based *graphical user interface* to the application.



The "Select a file" button launches a *file-dialog* to navigate to and select a file to analyze.

If a valid file is selected, *text-input widgets* and a *color-picker tool* pop up to help set the report's *title*, *target/groupby variable(optional)* and *graph color*.

Afterwards, a final file-dialog appears to help set the destination for the generated report.

Tip: For help with *Tk* - related issues, consider visiting TkDocs.

3.3.6 eda_report.plotting

You can find a wealth of plotting libraries at the PyViz website.

The plotting functions below are implemented using matplotlib. In the interest of efficiency, especially for large datasets with numerous columns; these plotting functions use a *non-interactive* matplotlib backend. This was inspired by Embedding in a web application server, which says in part:

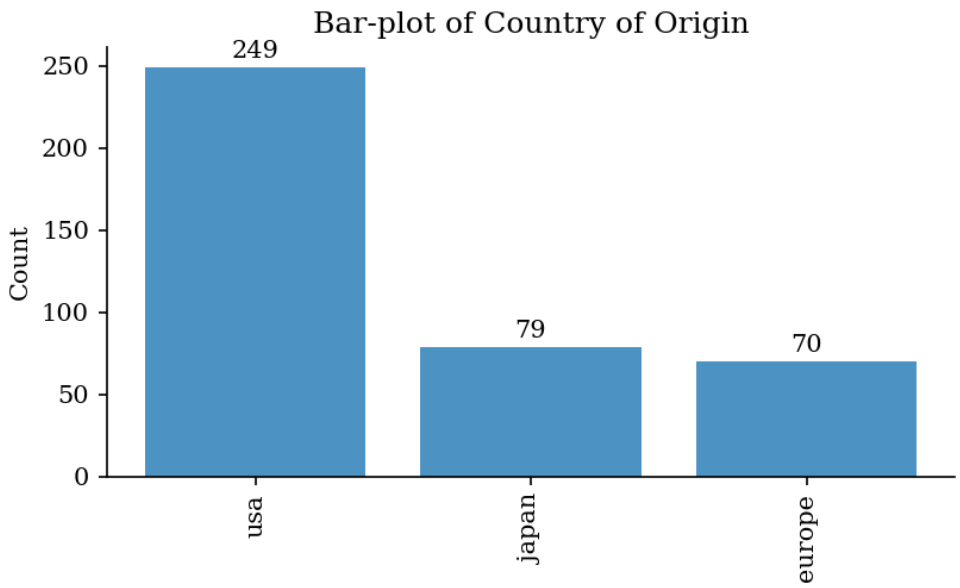
> When using Matplotlib in a web server [GUI application, in this case] it is strongly recommended to not use `pyplot` (pyplot maintains references to the opened figures to make show work, but this will cause memory leaks unless the figures are properly closed).

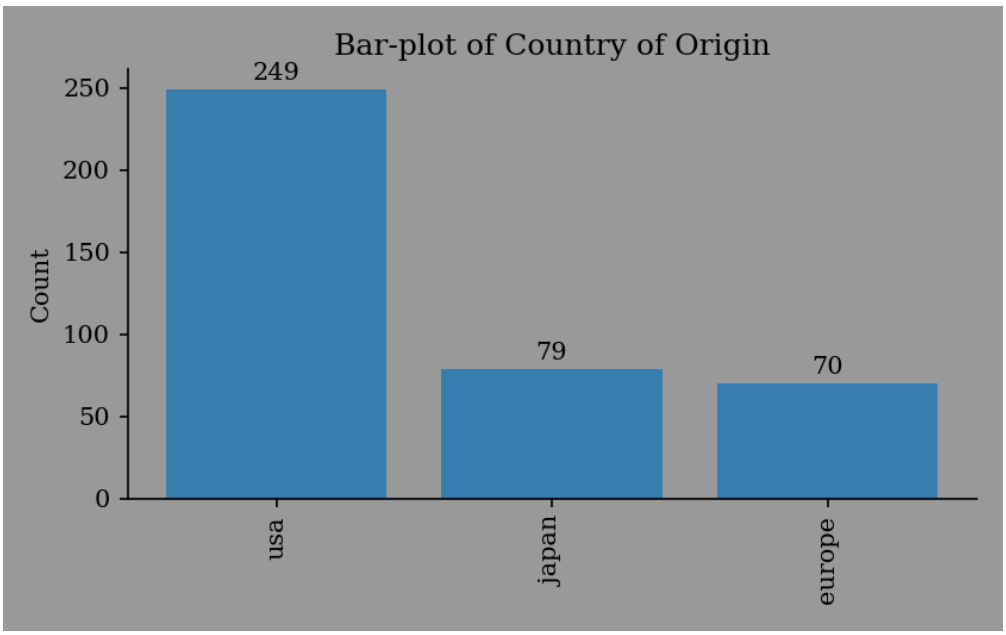
You can conveniently view the generated figures in a *jupyter notebook* using `%matplotlib inline`, as shown in this demo notebook.

Otherwise, you'll probably need to export them as images.

Plotting Examples

```
>>> import eda_report.plotting as ep
>>> ax = ep.bar_plot(mpg_data["origin"], label="Country of Origin")
>>> ax.figure.savefig("bar-plot.png")
```



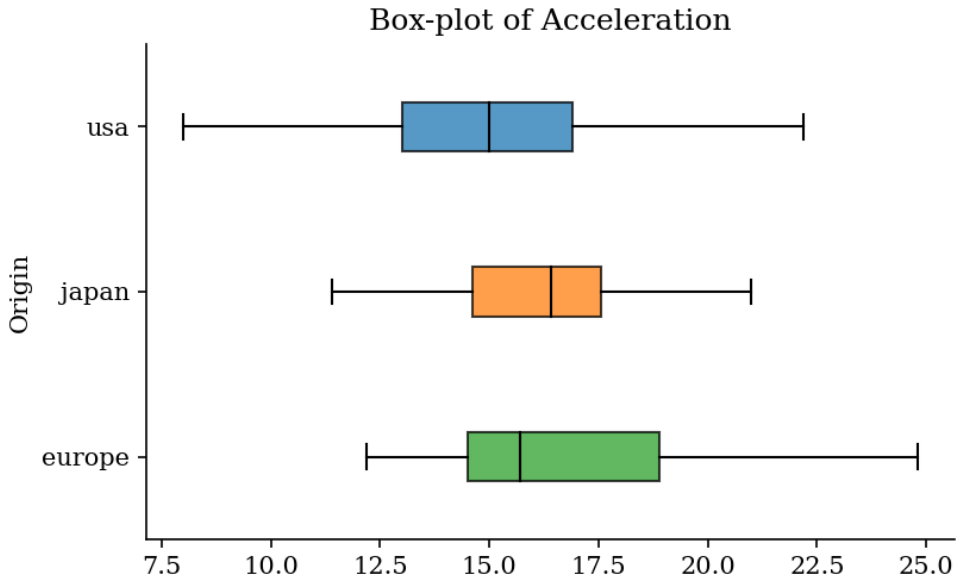


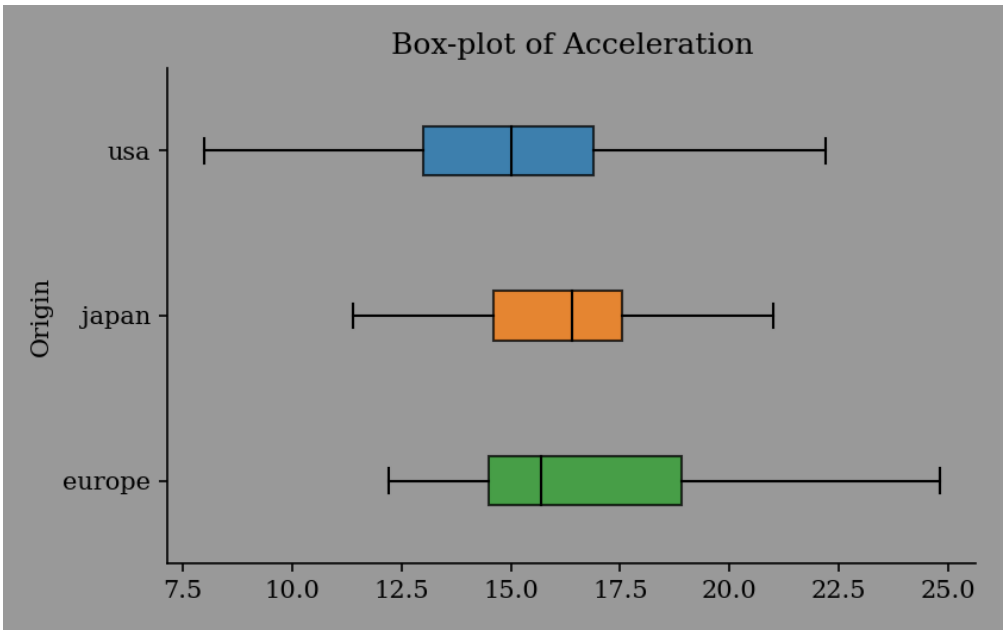

```
>>> ax = ep.box_plot(mpg_data["acceleration"], label="Acceleration", hue=mpg_data["origin
↪"])
```

(continues on next page)

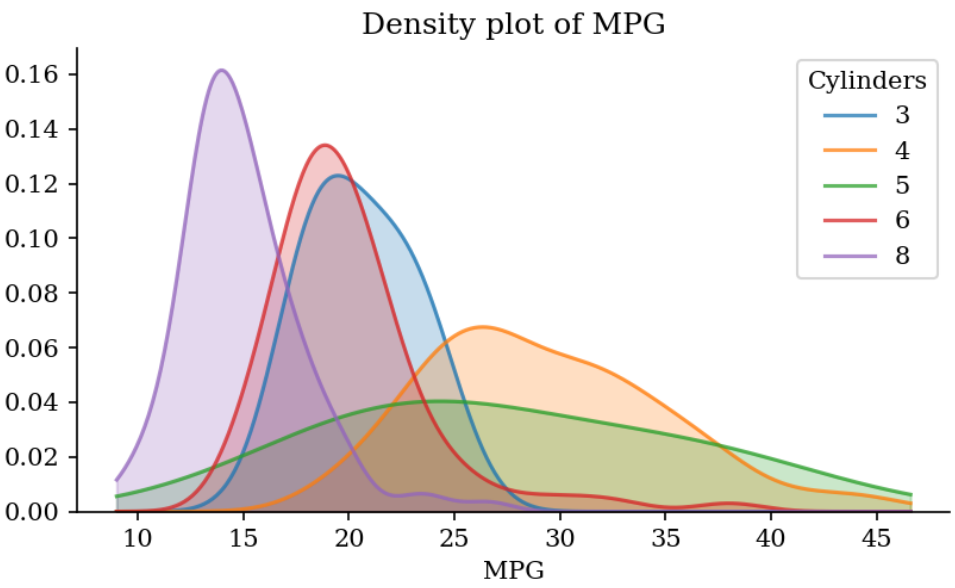
(continued from previous page)

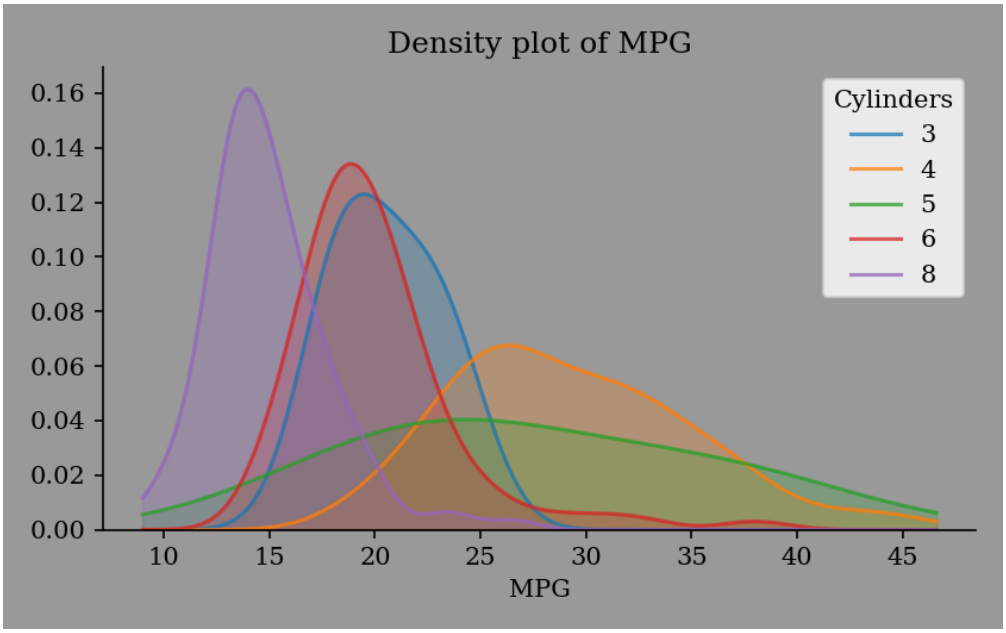
```
>>> ax.figure.savefig("box-plot.png")
```



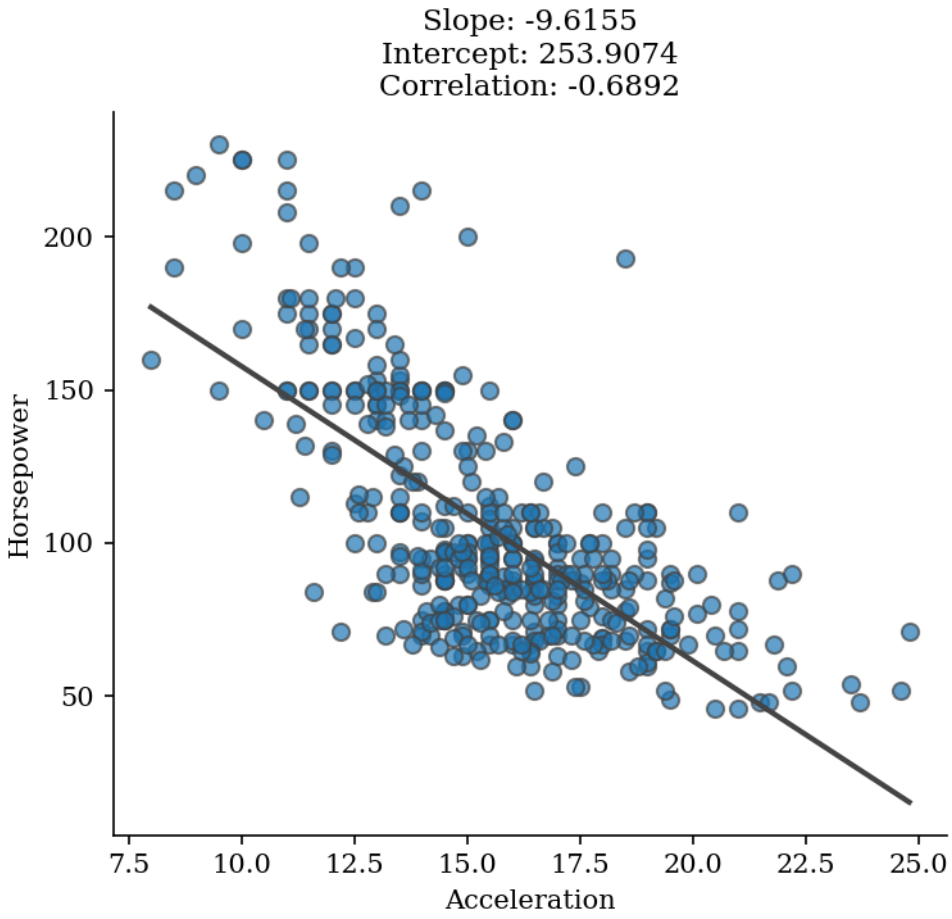


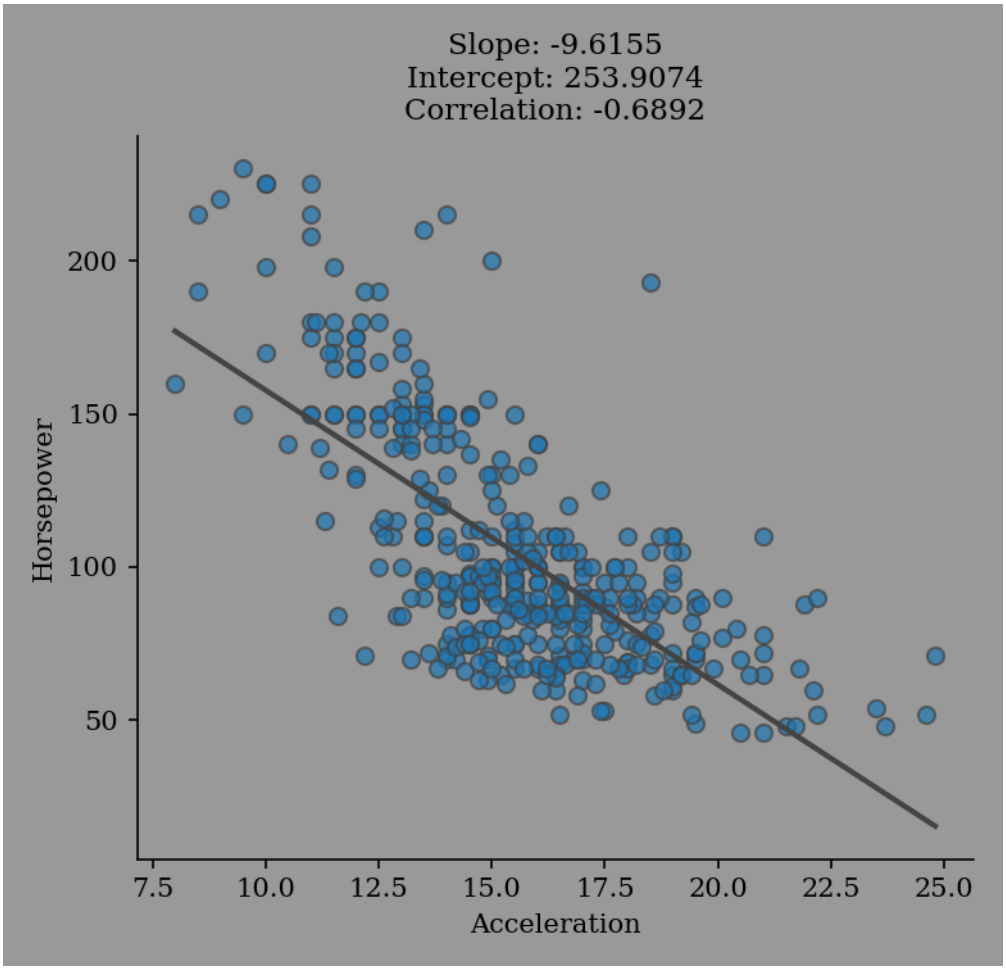

```
>>> ax = ep.kde_plot(mpg_data["mpg"], label="MPG", hue=mpg_data["cylinders"])
>>> ax.figure.savefig("kde-plot.png")
```



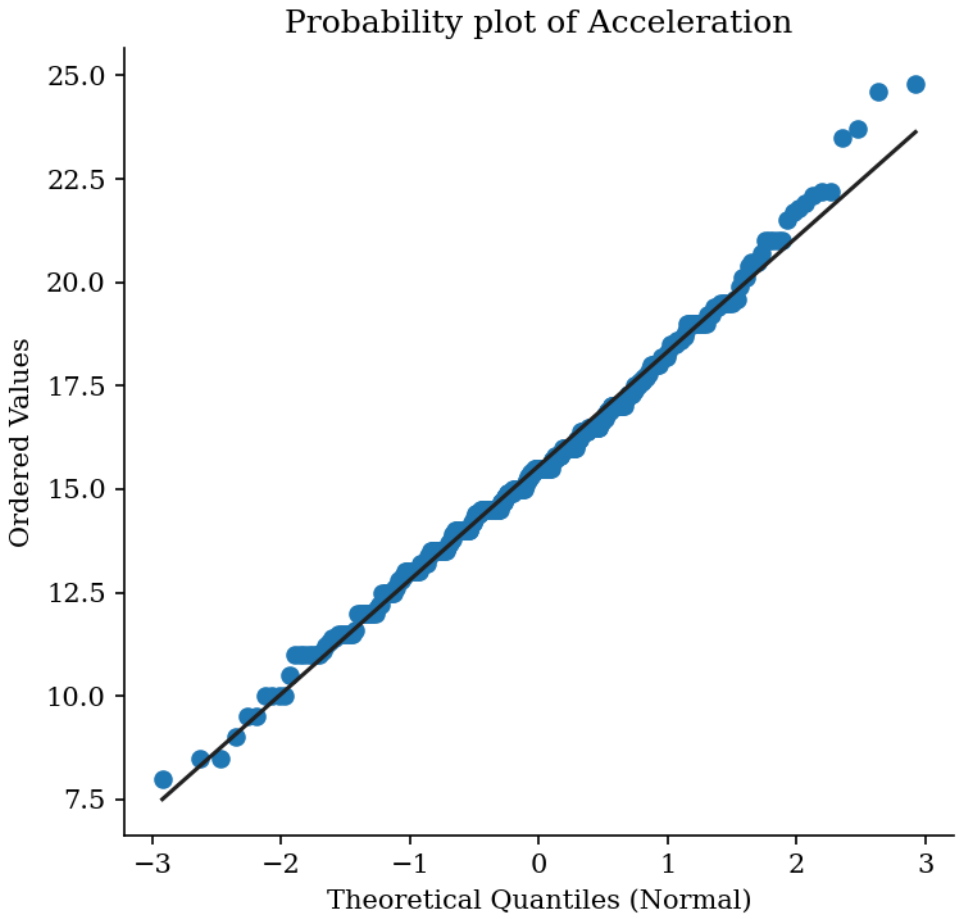


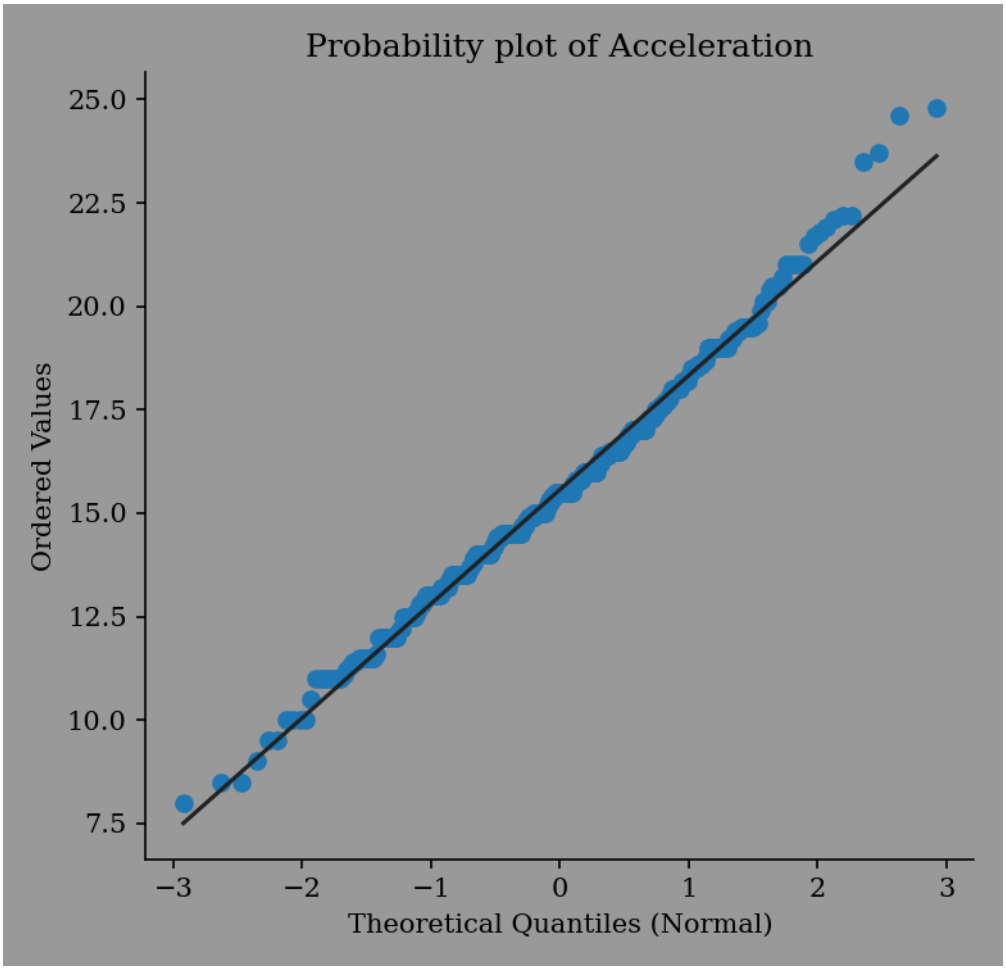

```
>>> ax = ep.regression_plot(mpg_data["acceleration"], mpg_data["horsepower"],
...                         labels=("Acceleration", "Horsepower"))
>>> ax.figure.savefig("regression-plot.png")
```


Slope: -9.6155
Intercept: 253.9074
Correlation: -0.6892
Horsepower
200
150
100
50
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
Acceleration

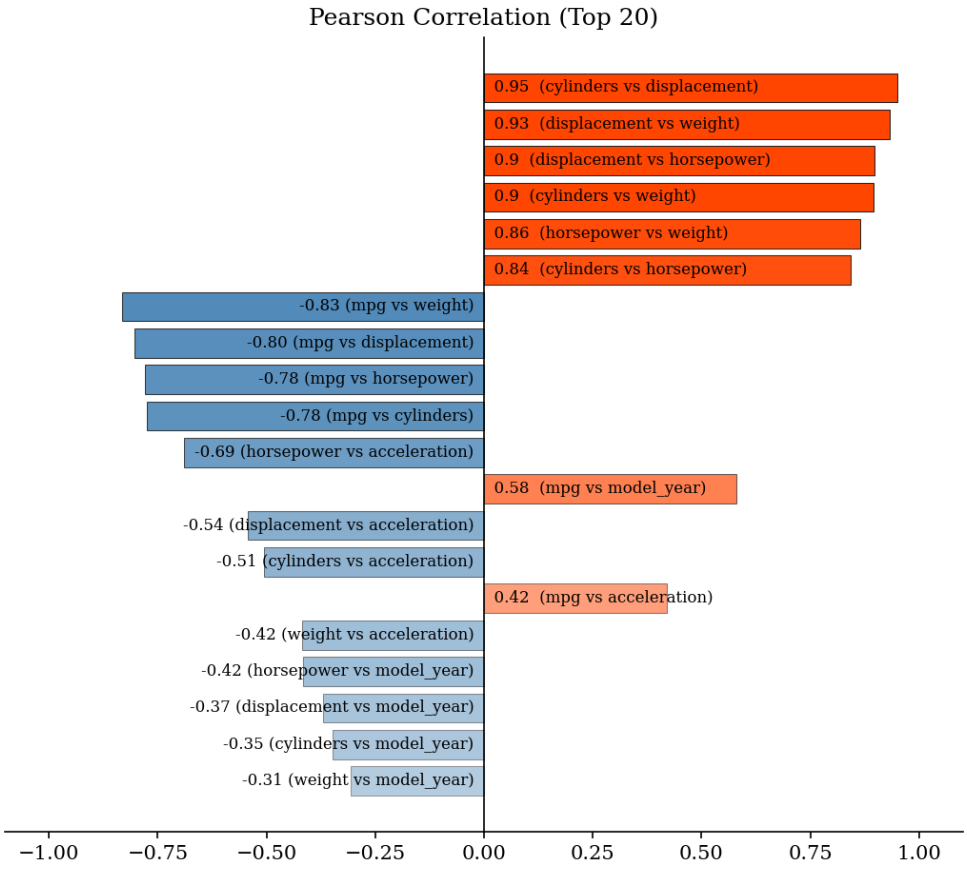


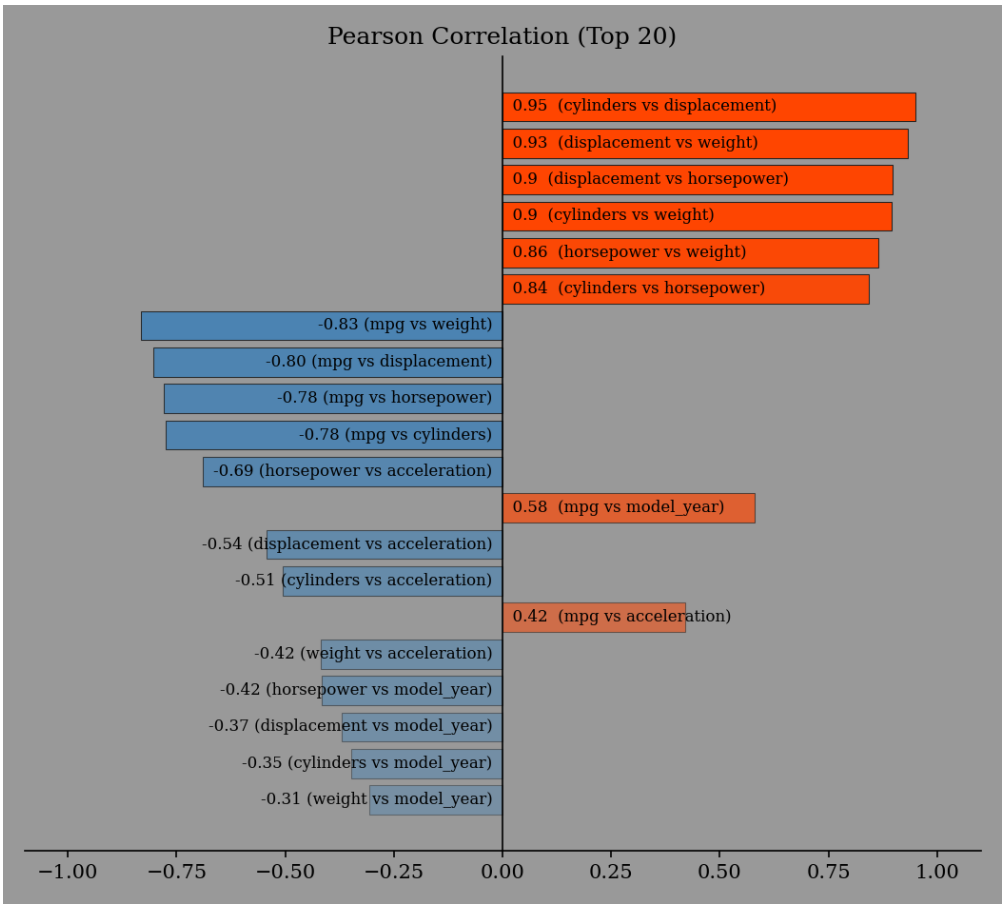

```
>>> ax = ep.prob_plot(mpg_data["acceleration"], label="Acceleration")
>>> ax.figure.savefig("probability-plot.png")
```


Probability plot of Acceleration
Ordered Values
Theoretical Quantiles (Normal)
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
−3
−2
−1
0
1
2
3




```
>>> ax = ep.plot_correlation(mpg_data)
>>> ax.figure.savefig("correlation-plot.png")
```


Pearson Correlation (Top 20)
0.95 (cylinders vs displacement)
0.93 (displacement vs weight)
0.9 (displacement vs horsepower)
0.9 (cylinders vs weight)
0.86 (horsepower vs weight)
0.84 (cylinders vs horsepower)
-0.83 (mpg vs weight)
-0.80 (mpg vs displacement)
-0.78 (mpg vs horsepower)
-0.78 (mpg vs cylinders)
-0.69 (horsepower vs acceleration)
0.58 (mpg vs model_year)
-0.54 (displacement vs acceleration)
-0.51 (cylinders vs acceleration)
0.42 (mpg vs acceleration)
-0.42 (weight vs acceleration)
-0.42 (horsepower vs model_year)
-0.37 (displacement vs model_year)
-0.35 (cylinders vs model_year)
-0.31 (weight vs model_year)
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00



eda_report.plotting.**bar_plot**(*data: Iterable*, *, *label: str*, *color: str | Sequence = None*, *ax: Axes = None*) → Axes

Get a bar-plot from a sequence of values.

Parameters

- **data** (*Iterable*) – Values to plot.
- **label** (*str*) – A name for the data, shown in the title.
- **color** (*Union[str, Sequence]*) – A valid matplotlib color specifier.
- **ax** (*matplotlib.axes.Axes, optional*) – Axes instance. Defaults to None.

Returns

Matplotlib axes with the bar-plot.

Return type

matplotlib.axes.Axes

eda_report.plotting.**box_plot**(*data: Iterable*, *, *label: str*, *hue: Iterable = None*, *color: str | Sequence = None*, *ax: Axes = None*) → Axes

Get a box-plot from numeric values.

Parameters

- **data** (*Iterable*) – Values to plot.
- **label** (*str*) – A name for the data, shown in the title.

- **hue** (*Iterable, optional*) – Values for grouping the `data`. Defaults to None.
- **color** (*Union[str, Sequence]*) – A valid matplotlib color specifier.
- **ax** (*matplotlib.axes.Axes, optional*) – Axes instance. Defaults to None.

Returns
Matplotlib axes with the box-plot.

Return type
matplotlib.axes.Axes

eda_report.plotting.**kde_plot**(*data: Iterable*, *, *label: str*, *hue: Iterable = None*, *color: str | Sequence = None*, *ax: Axes = None*) → Axes

Get a kde-plot from numeric values.

Parameters

- **data** (*Iterable*) – Values to plot.
- **label** (*str*) – A name for the `data`, shown in the title.
- **hue** (*Iterable, optional*) – Values for grouping the `data`. Defaults to None.
- **color** (*Union[str, Sequence]*) – A valid matplotlib color specifier.
- **ax** (*matplotlib.axes.Axes, optional*) – Axes instance. Defaults to None.

Returns
Matplotlib axes with the kde-plot.

Return type
matplotlib.axes.Axes

eda_report.plotting.**plot_correlation**(*variables: Iterable*, *max_pairs: int = 20*, *color_pos: str | Sequence = 'orangered'*, *color_neg: str | Sequence = 'steelblue'*, *ax: Axes = None*) → Axes

Create a bar chart showing the top `max_pairs` most correlated variables. Bars are annotated with variable pairs and their respective Pearson correlation coefficients.

Parameters

- **variables** (*Iterable*) – 2-dimensional numeric data.
- **max_pairs** (*int*) – The maximum number of numeric pairs to include in the plot. Defaults to 20.
- **color_pos** (*Union[str, Sequence]*) – Color for positive correlation bars. Defaults to "orangered".
- **color_neg** (*Union[str, Sequence]*) – Color for negative correlation bars. Defaults to "steelblue".
- **ax** (*matplotlib.axes.Axes, optional*) – Axes instance. Defaults to None.

Returns
A bar-plot of correlation data.

Return type
matplotlib.axes.Axes

eda_report.plotting.**prob_plot**(*data: Iterable*, *, *label: str*, *marker_color: str | Sequence = 'C0'*, *line_color: str | Sequence = '#222'*, *ax: Axes = None*) → Axes

Get a probability-plot from numeric values.

Parameters

- **data** (*Iterable*) – Values to plot.
- **label** (*str*) – A name for the `data`, shown in the title.
- **marker_color** (*Union[str, Sequence]*) – Color for the plotted points. Defaults to "C0".
- **line_color** (*Union[str, Sequence]*) – Color for the line of best fit. Defaults to "#222".
- **ax** (*matplotlib.axes.Axes, optional*) – Axes instance. Defaults to None.

Returns
Matplotlib axes with the probability-plot.

Return type
matplotlib.axes.Axes

eda_report.plotting.**regression_plot**(*x: Iterable*, *y: Iterable*, *labels: Tuple[str, str]*, *marker_color: str | Sequence = 'C0'*, *line_color: str | Sequence = '#444'*, *ax: Axes = None*) → Axes

Get a regression-plot from the provided pair of numeric values.

Parameters

- **x** (*Iterable*) – Numeric values.
- **y** (*Iterable*) – Numeric values.
- **labels** (*Tuple[str, str]*) – Names for *x* and *y* respectively, shown in axis labels.
- **marker_color** (*Union[str, Sequence]*) – Color for the plotted points. Defaults to "C0".
- **line_color** (*Union[str, Sequence]*) – Color for the line of best fit. Defaults to "#444".
- **ax** (*matplotlib.axes.Axes, optional*) – Axes instance. Defaults to None.

Returns
Matplotlib axes with the regression-plot.

Return type
matplotlib.axes.Axes

3.3.7 eda_report.univariate

class eda_report.univariate.**Variable**(*data: Iterable*, *, *name: str = None*)

Obtain summary statistics and properties such as data type, missing value info & cardinality from one-dimensional datasets.

Parameters

- **data** (*Iterable*) – The data to analyze.
- **name** (*str, optional*) – The name to assign the variable. Defaults to None.

Examples

```
>>> from eda_report.univariate import Variable
>>> Variable(range(1, 51), name="1 to 50")

Name: 1 to 50
Type: numeric
Non-null Observations: 50
Unique Values: 50 -> [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, [...]
Missing Values: None

                  Summary Statistics
                  ------------------
        Average:                      25.5000
        Standard Deviation:           14.5774
        Minimum:                       1.0000
        Lower Quartile:               13.2500
        Median:                       25.5000
        Upper Quartile:               37.7500
        Maximum:                      50.0000
        Skewness:                      0.0000
        Kurtosis:                     -1.2000

                  Tests for Normality
                  -------------------
                               p-value Conclusion at  = 0.05
D'Agostino's K-squared test  0.0015981  Unlikely to be normal
Kolmogorov-Smirnov test      0.0000000  Unlikely to be normal
Shapiro-Wilk test            0.0580895        Possibly normal
```

```
>>> Variable(["mango", "apple", "pear", "mango", "pear", "mango"], name="fruits")

Name: fruits
Type: categorical
Non-null Observations: 6
Unique Values: 3 -> ['apple', 'mango', 'pear']
Missing Values: None
Mode (Most frequent): mango
Maximum frequency: 3

                Most Common Items
                -----------------
                   mango: 3 (50.00%)
                    pear: 2 (33.33%)
                   apple: 1 (16.67%)
```

```
>>> import pandas as pd
>>> dt = pd.date_range("2022-03-08", periods=20, freq="D")
>>> Variable(dt, name="dttm")

Name: dttm
Type: datetime
```

(continues on next page)

(continued from previous page)

```
Non-null Observations: 20
Unique Values: 20 -> [Timestamp('2022-03-08 00:00:00'), [...]
Missing Values: None

                  Summary Statistics
                  ------------------
        Average:              2022-03-17 12:00:00
        Minimum:              2022-03-08 00:00:00
        Lower Quartile:       2022-03-12 18:00:00
        Median:               2022-03-17 12:00:00
        Upper Quartile:       2022-03-22 06:00:00
        Maximum:              2022-03-27 00:00:00
```

missing

The number of *missing values* in the form `number (% of total count)` e.g "4 (16.67%)".

Type
str

name

The variable's *name*. If no name is specified, the name will be set the value of the `name` attribute of the input data, or `None`.

Type
str

num_unique

The *number of unique values* present in the variable.

Type
int

rename(*name: str*) → None

Update the variable's name.

Parameters
name (*str*) – New name.

summary_stats

Descriptive statistics

Type
dict

unique_values

The *unique values* present in the variable.

Type
list

var_type

The type of variable — one of *"boolean"*, *"categorical"*, *"datetime"*, *"numeric"* or *"numeric (<=10 levels)"*.

Type
str

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex

PYTHON MODULE INDEX

e

INDEX