Four R packages for Automated Exploratory Data Analysis you might have missed

Explore useful tools to ease exploratory tasks through practical examples in R

Image by author.

Table of contents

1. Introduction

Exploratory Data Analysis (EDA) aims at performing an initial investigation on the data by summarizing their characteristics through statistical and visualization techniques, and it is a critical early step in any Data Science workflow.

In this post, we explore four R packages that facilitate this initial task and provide a substantial support in data handling, visualization and reporting.

For this example, we use the `Cardiovascular Disease Dataset`[1], composed of the following features:

1. `id`: unique patient identifier (int)
2. `gender`: gender of the patient (1: female, 2: male)
3. `age`: age in days (int)
4. `height`: height in cm (int)
5. `weight`: weight in kg (float)

6. `ap_hi`: systolic blood pressure (int)
7. `ap_lo`: diastolic blood pressure (int)
8. `cholesterol`: cholesterol (1: normal, 2: above normal, 3: well above normal)
9. `gluc`: glucose (1: normal, 2: above normal, 3: well above normal)
10. `smoke`: whether patient smokes or not (binary)
11. `alco`: alcohol intake (binary)
12. `active`: physical activity (binary)
13. `cardio`: presence or absence of cardiovascular disease (binary)

We load the dataset and observe the first rows:

```
# install.packages("tidyverse")
library(dplyr)
library(readr)


# read csv file
df <-
  read_delim(
    file = "/data/cardio_train.csv",
    col_types = "iifidiiffffff",
    delim=";")


# pre-processing
df <-
  # remove the id
  select(df, -id) %>%
  # age: days -> years
  mutate(age = round(age / 365))


# observe first rows
head(df)
```

age <dbl>	gender <fctr>	height <int>	weight <dbl>	ap_hi <int>	ap_lo <int>	cholesterol <fctr>	gluc <fctr>	smoke <fctr>	alco <fctr>	active <fctr>	cardio <fctr>
50	2	168	62	110	80	1	1	0	0	1	0
55	1	156	85	140	90	3	1	0	0	1	1
52	1	165	64	130	70	3	1	0	0	0	1
48	2	169	82	150	100	1	1	0	0	1	1
48	1	156	56	100	60	1	1	0	0	0	0
60	1	151	67	120	80	2	2	0	0	0	0

Image by author.

Base R provides `summary`[2], a generic function used to produce result overviews from different input objects, such as datasets. In particular, when a dataset is provided as input (e.g. `summary(df)`), it returns different metrics (such as mean, median, min, max, …) for numeric columns, and the distribution (counts) for categorical columns. It also returns information about missing data, if present:

```
      age              height           weight            ap_hi              ap_lo         
 Min.   :30.00   Min.   : 55.0   Min.   : 10.00   Min.   : -150.0   Min.   :  -70.00  
 1st Qu.:48.00   1st Qu.:159.0   1st Qu.: 65.00   1st Qu.:  120.0   1st Qu.:   80.00  
 Median :54.00   Median :165.0   Median : 72.00   Median :  120.0   Median :   80.00  
 Mean   :53.34   Mean   :164.4   Mean   : 74.21   Mean   :  128.8   Mean   :   96.63  
 3rd Qu.:58.00   3rd Qu.:170.0   3rd Qu.: 82.00   3rd Qu.:  140.0   3rd Qu.:   90.00  
 Max.   :65.00   Max.   :250.0   Max.   :200.00   Max.   :16020.0   Max.   :11000.00  

  gender       cholesterol     gluc        smoke        alco         active       cardio      
 2:24470   1:52385      1:59479   0:63831   0:66236   1:56261   0:35021  
 1:45530   3: 8066      2: 5190   1: 6169   1: 3764   0:13739   1:34979  
           2: 9549      3: 5331                                          
```

Output of `summary(df)`. Image by author.

Now, let us jump to packages that provide further data exploration capabilities.

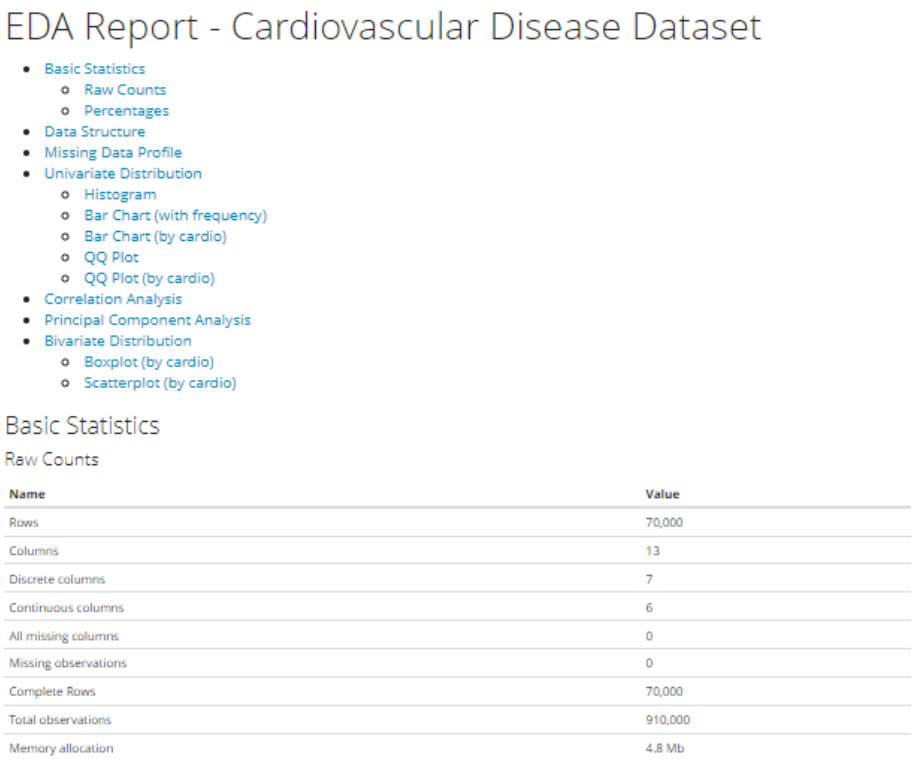
2. Automated Exploratory Data Analysis packages

2.1. DataExplorer

`DataExplorer`[3] simplifies and automates the EDA process and report generation. The package automatically scans through each variable performing data profiling, and it offers several helpful functions to generate different charts on both discrete and continuous features.

Importantly, the package allows to generate a **complete HTML report** without effort by invoking the helpful `create_report` function on a dataset (e.g. `create_report(df)`). It is possible to pass additional arguments, such a response variable `y`, to add various bivariate analyses to the report:

The snippet produces an HTML file in the working directory. When opened with a browser, it appears as follow:



EDA Report - Cardiovascular Disease Dataset

- Basic Statistics
  - Raw Counts
  - Percentages
- Data Structure
- Missing Data Profile
- Univariate Distribution
  - Histogram
  - Bar Chart (with frequency)
  - Bar Chart (by cardio)
  - QQ Plot
  - QQ Plot (by cardio)
- Correlation Analysis
- Principal Component Analysis
- Bivariate Distribution
  - Boxplot (by cardio)
  - Scatterplot (by cardio)

Basic Statistics

Raw Counts

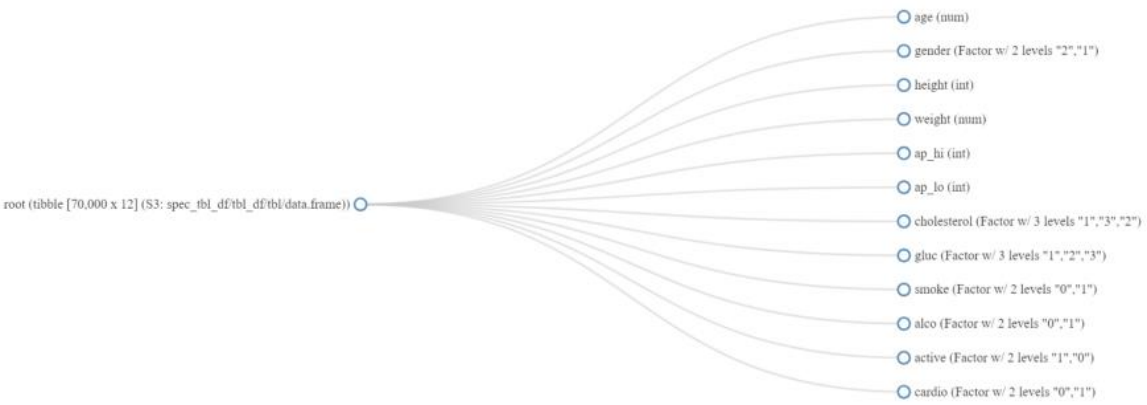
Name	Value
Rows	70,000
Columns	13
Discrete columns	7
Continuous columns	6
All missing columns	0
Missing observations	0
Complete Rows	70,000
Total observations	910,000
Memory allocation	4.8 Mb

The early part of the HTML report, comprehensive of table of contents. Image by author.

As it can be seen from the table of contents, the report covers most of the tasks performed during EDA, and it can be generated by a single line of code.

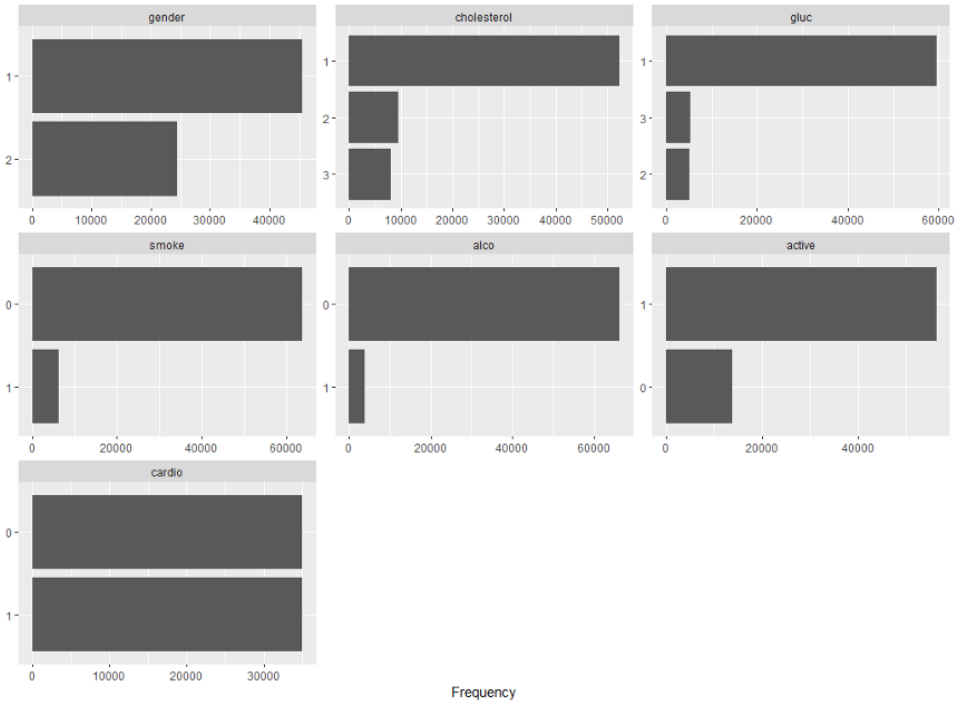
Additional package functions for specific tasks are reported as example:

- `plot_str(df)` plots the dataset structure:



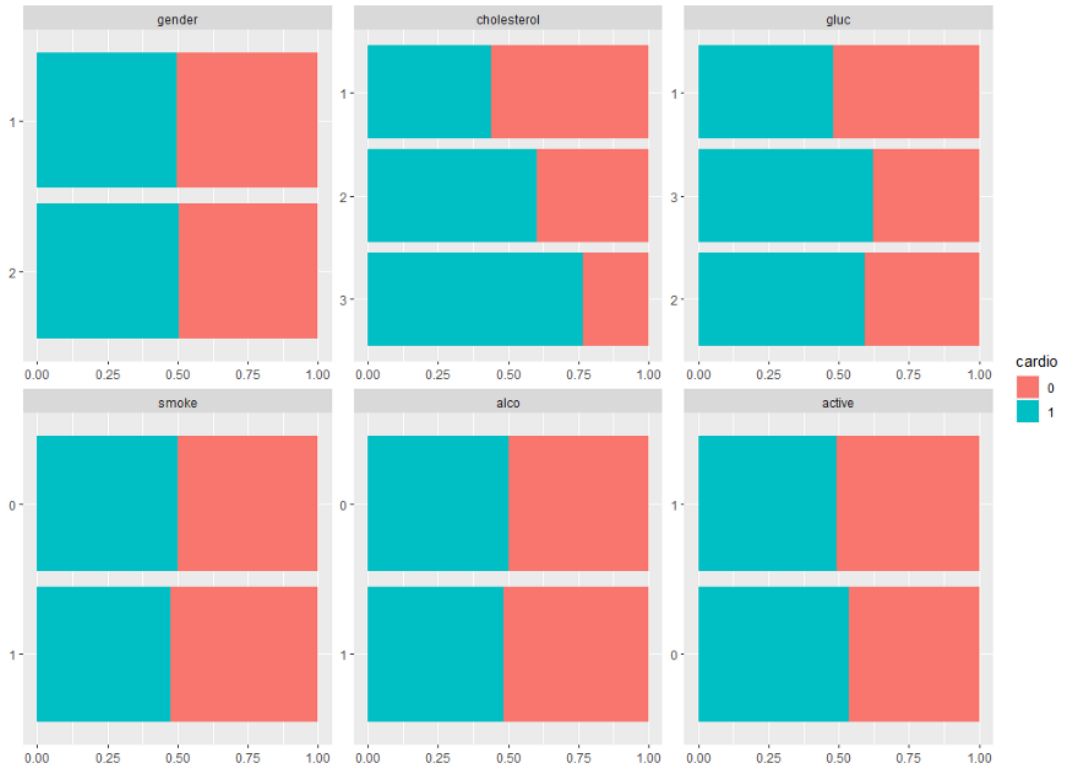
Output of `plot_str(df)`. Image by author.

- `plot_bar(df)` plots the bar chart for each discrete feature:



Output of `plot_bar(df)`. Image by author.

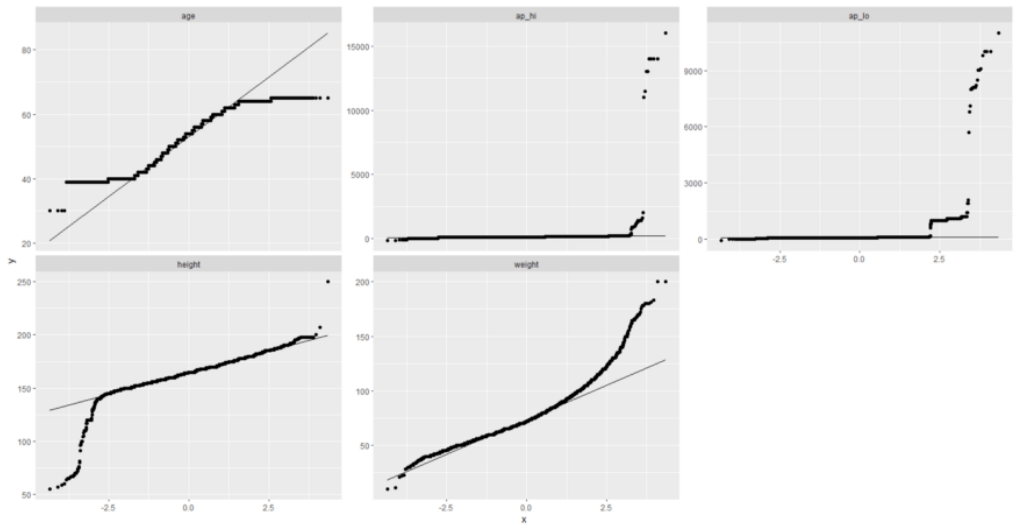
- The charts can also be grouped by a discrete variable, e.g. the presence of a cardiovascular disease, by: `plot_bar(df, by="cardio")`:



Output `of plot_bar(df, by="cardio")`. Image by author.

By observing the plots, one may immediately notice that patients with cardiovascular disease present higher values of cholesterol and may be induced to further investigate the risk of cardiovascular disease represented by higher cholesterol through statistical testing.

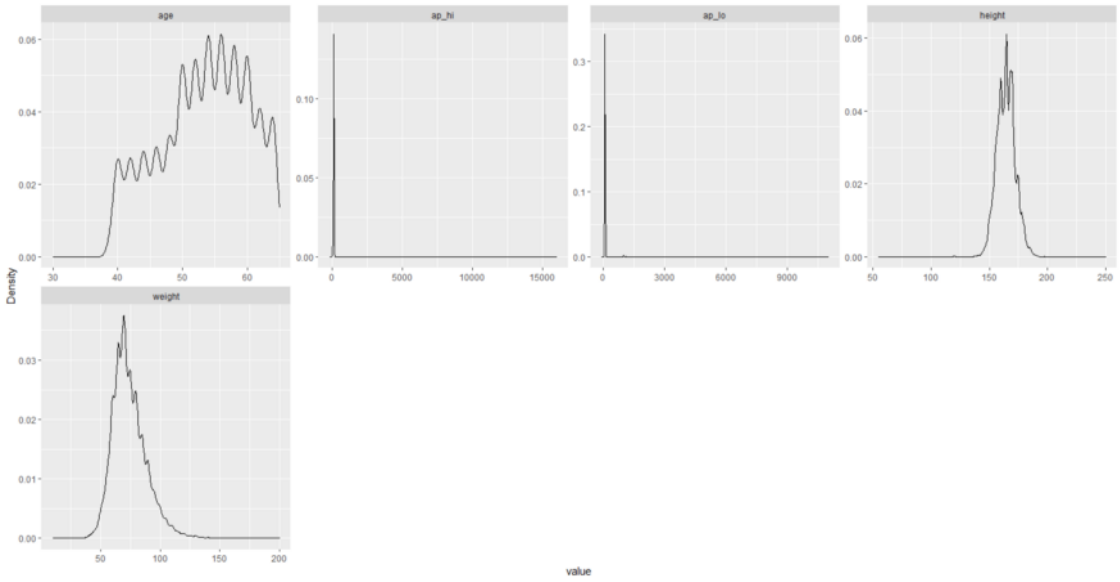
- `plot_qq(df)` plots quantile-quantile for each continuous feature:



Output of `plot_qq(df)`. Image by author.

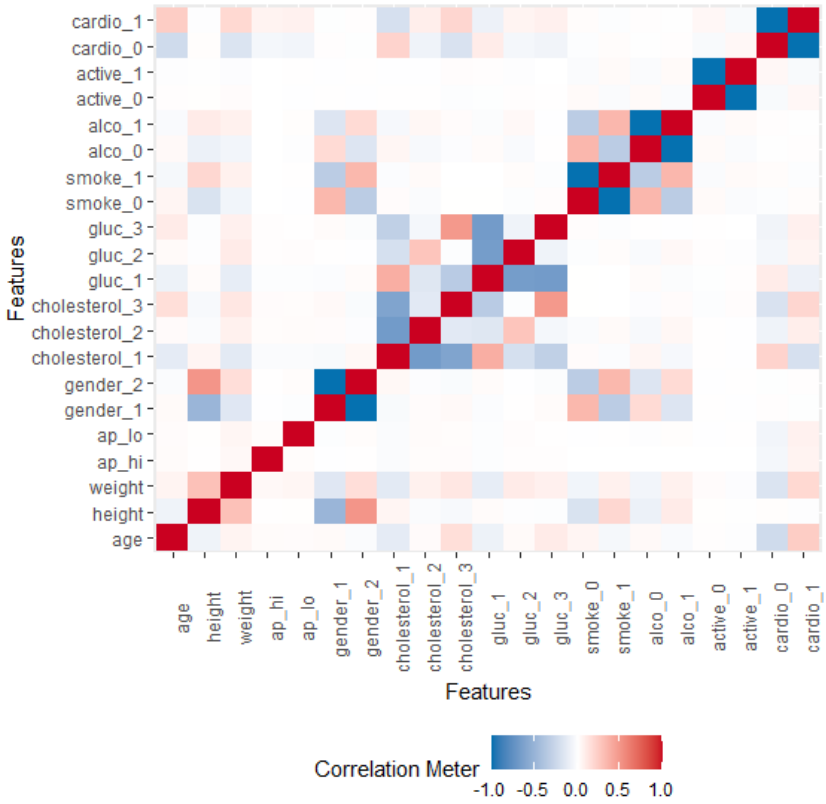
By observing the first QQ plot (age), for example, one may notice that the points tend to curve at the extremities, indicating that age has more extreme values that it would have if it had a normal distribution (this behaviour falls under the name of "*heavy tails*").

- `plot_density(df)` plots density estimates for each continuous feature:



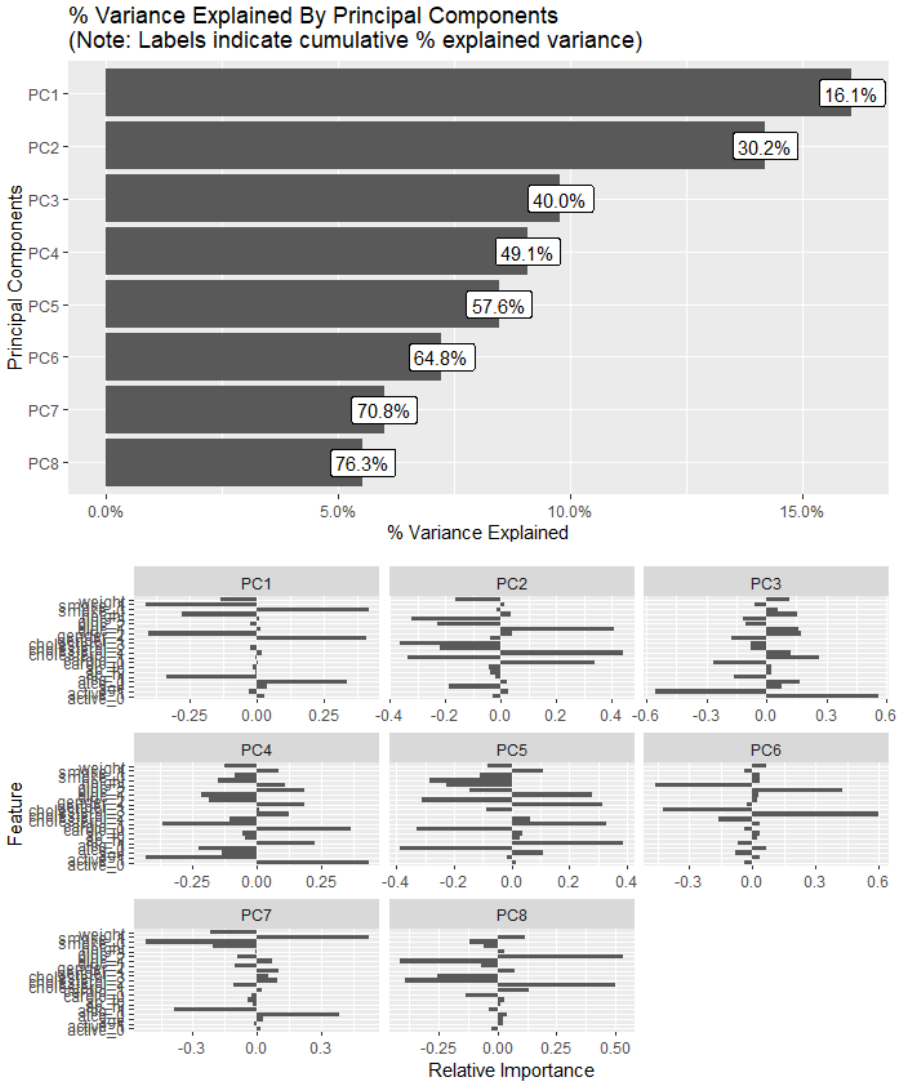
Output of `plot_density(df)`. Image by author.

- `plot_correlation(df)` to visualize correlation heatmap:



Output of `plot_correlation(df)`. Image by author.

- `plot_prcomp(df)` performs Principal Component Analysis (PCA) and plots the percentage of variance explained by each principal component:



Output of `plot_prcomp(df)`. Image by author.

The package documentation[3] contains the full list of the available functions.

2.2 GGally

`GGally`[4] extends the popular `ggplot2` plotting package by providing features to automatically visualize datasets and combine geometric objects.

In particular, one may leverage the `ggpairs`[5] function on a dataset (e.g. `ggpairs(df)`) and obtain pairs plot showing the interactions of each variable with each of the others:

```
#install.packages("GGally")
library(GGally)

# change plot size (optional)
options(repr.plot.width = 20, repr.plot.height = 10)

df %>%
  select("age", "cholesterol", "height", "weight") %>%
  ggpairs(mapping = aes(color = df$cardio, alpha = 0.5))
```

Image by author.

`ggpairs` offers an easy way to generate plots for descriptive analysis, proving useful to spot outliers, observe distributions and visually inspect differences between groups.

In our example, we chose for simplicity to generate plots for only four variables of the dataset `(df %>% select("age", "cholesterol", "height", "weight"))`, and we applied different colours based on the presence of cardiovascular disease through the `ggplot2` aesthetics `(aes(color = df$cardio, alpha = 0.5))`.

The charts can be further customized by specifying the types of plots desired in the lower, upper and diagonal panels[5].

2.3 SmartEDA

`SmartEDA`[6] is a powerful toolkit that automates most of EDA tasks by providing functions for:

- Descriptive statistics
- Data visualisation
- Custom tables
- HTML reports

Similarly to what we did with `DataExplorer`, we can generate a complete HTML report by calling the `ExpReport` function. We can specify a response variable `Target` to include further bivariate analyses to the report:

This snippet produces a file named `Report.html` in the working directory. When opened with a browser, it looks as follows:

Exploratory Data Analysis Report

2022-01-11

Exploratory Data analysis (EDA)

Analyzing the data sets to summarize their main characteristics of variables, often with visual graphs, without using a statistical model.

1. Overview of the data

Understanding the dimensions of the dataset, variable names, overall missing summary and data types of each variables

The early part of the HTML report, comprehensive of table of contents. Image by author.

The report contains a complete set of exploratory analyses, and it can be generated without coding efforts.

Additional package functions for specific tasks are reported as example:

- `ExpData(data=df, type=1)` displays an overview over a dataset:

Descriptions	Value
<chr>	<chr>
Sample size (nrow)	70000
No. of variables (ncol)	12
No. of numeric/interger variables	5
No. of factor variables	7
No. of text variables	0
No. of logical variables	0
No. of identifier variables	0
No. of date variables	0
No. of zero variance variables (uniform)	0
%. of variables having complete cases	100% (12)
%. of variables having >0% and <50% missing cases	0% (0)
%. of variables having >=50% and <90% missing cases	0% (0)
%. of variables having >=90% missing cases	0% (0)

Output of `ExpData(data=df, type=1)`. Image by author.

- `ExpData(data=df, type=2)` shows the data structure:

Index	Variable_Name	Variable_Type	Sample_n	Missing_Count	Per_of_Missing	No_of_distinct_values
<dbl>	<chr>	<chr>	<int>	<int>	<dbl>	<int>
1	age	numeric	70000	0	0	28
2	gender	factor	70000	0	0	2
3	height	numeric	70000	0	0	109
4	weight	numeric	70000	0	0	287
5	ap_hi	numeric	70000	0	0	153
6	ap_lo	numeric	70000	0	0	157
7	cholesterol	factor	70000	0	0	3
8	gluc	factor	70000	0	0	3
9	smoke	factor	70000	0	0	2
10	alco	factor	70000	0	0	2
11	active	factor	70000	0	0	2
12	cardio	factor	70000	0	0	2

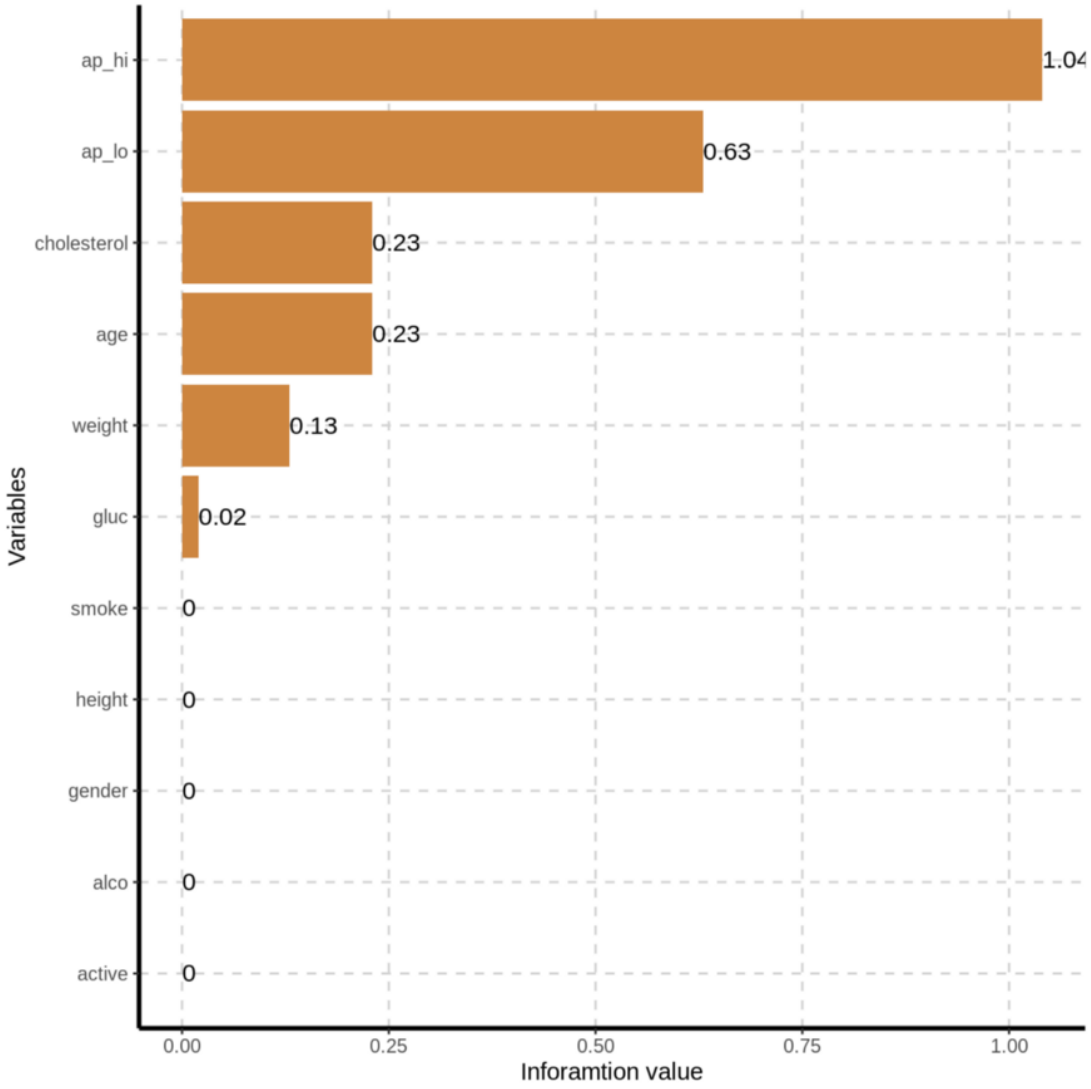
Output of `ExpData(data=df, type=2)`. Image by author.

- `ExpCatStat(df, Target="cardio", Pclass="1", plot=TRUE)` for variable importance based on information value:

Variable	Target	Unique	Chi-squared	p-value	df	IV Value	Cramers V	Degree of Association	Predictive Power
<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	<chr>
gender	cardio	2	4.570	0.033	1	0.00	0.01	Very Weak	Not Predictive
cholesterol	cardio	3	3423.439	0.000	2	0.23	0.22	Moderate	Medium Predictive
gluc	cardio	3	586.912	0.000	2	0.02	0.09	Weak	Not Predictive
smoke	cardio	2	16.678	0.000	1	0.00	0.02	Very Weak	Not Predictive
alco	cardio	2	3.697	0.055	1	0.00	0.01	Very Weak	Not Predictive
active	cardio	2	88.801	0.000	1	0.00	0.04	Very Weak	Not Predictive
age	cardio	10	4017.254	0.000	9	0.23	0.24	Moderate	Medium Predictive
height	cardio	10	44.395	0.000	9	0.00	0.03	Very Weak	Not Predictive
weight	cardio	10	2349.729	0.000	9	0.13	0.18	Weak	Somewhat Predictive
ap_hi	cardio	6	15051.773	0.000	5	1.06	0.46	Strong	Highly Predictive
ap_lo	cardio	6	9659.093	0.000	5	0.61	0.37	Strong	Highly Predictive

Output of `ExpCatStat(df, Target="cardio", Pclass="1", plot=TRUE)`. Image by author.

From this output, we gain information on the **degree of association** and **predictive power** of each covariate towards the cardiovascular disease (`Target="cardio"`). In particular, we can observe that the **systolic and diastolic blood pressure** are categorized as "***Highly predictive***" of cardiovascular disease, while **cholesterol and age** appear to be "***Medium predictive***". The function also produces a plot of the information value of each variable:



Output of `ExpCatStat(df, Target="cardio", Pclass="1", plot=TRUE)`. Image by author.

The package provides a variety of handy functions to generate plots and tables. The full list of available functions, together with examples, is available in the package documentation[7], or GitHub[6].

2.4 tableone

In biomedical journal papers, the *Table 1* provides a quantitative breakdown of baseline patients characteristics. The `tableone` package aims at producing the typical *Table 1* (hence the name) from research publications.

Before using it, we perform an optional pre-processing step to make the categories and column names more readable:

```
df <-
  df %>%
  # modify factor levels
  mutate_at(c("cholesterol", "gluc"), ~ recode(.,
                                               "1" = "normal",
                                               "2" = "above normal",
                                               "3" = "well above
normal")
```

```
  ) %>%
  mutate_at(c("smoke", "alco", "active", "cardio"), ~ recode(.,
                                                             "0" =
"no",
                                                             "1" =
"yes")
  ) %>%
  mutate_at(c("gender"), ~ recode(.,
                                  "1" = "woman",
                                  "2" = "male")
  ) %>%
  # rename columns
  rename(
    "systolic blood pressure" = "ap_hi",
    "diastolic blood pressure" = "ap_lo",
    "glucose" = "gluc",
    "smoking" = "smoke",
    "alcohol" = "alco",
    "physical activity" = "active",
    "cardiovascular disease" = "cardio"
  )


head(df)
```

age <dbl>	gender <fctr>	height <int>	weight <dbl>	systolic blood pressure <int>	diastolic blood pressure <int>	cholesterol <fctr>	glucose <fctr>	smoking <fctr>	alcohol <fctr>	physical activity <fctr>	cardiovascular disease <fctr>
50	man	168	62	110	80	normal	normal	no	no	yes	no
55	woman	156	85	140	90	well above normal	normal	no	no	yes	yes
52	woman	165	64	130	70	well above normal	normal	no	no	no	yes
48	man	169	82	150	100	normal	normal	no	no	yes	yes
48	woman	156	56	100	60	normal	normal	no	no	no	no
60	woman	151	67	120	80	above normal	above normal	no	no	no	no

Image by author

We now use the `CreateTableOne` function to generate an object that summarizes all baseline variables, both continuous and categorical, optionally stratifying by one or more variables and performing statistical tests. In our case, we decide to stratify by the presence of cardiovascular disease through the `strata` argument:

```
# install.packages("tableone")
library(tableone)

# we perform a stratification based on the presence of cardiovascular
disease
tableOne <- CreateTableOne(vars = colnames(select(df, -"cardiovascular
disease")),
                           strata = c("cardiovascular disease"),
                           data = df)


# we pass a list of continuous variables not normally distributed in
the "nonnormal" argument
print(
  tableOne,
  nonnormal = c("age", "weight", "height", "systolic blood pressure",
"diastolic blood pressure"),
  showAllLevels = TRUE)
```

		Stratified by cardiovascular disease			
	level	no	yes	p	test
n		35021	34979		
age (median [IQR])		52.00 [46.00, 57.00]	56.00 [50.00, 60.00]	<0.001	nonnorm
gender (%)	man	12107 (34.6)	12363 (35.3)	0.033	
	woman	22914 (65.4)	22616 (64.7)		
height (median [IQR])		165.00 [159.00, 170.00]	165.00 [159.00, 170.00]	0.001	nonnorm
weight (median [IQR])		70.00 [63.00, 79.00]	75.00 [66.00, 85.00]	<0.001	nonnorm
systolic blood pressure (median [IQR])		120.00 [110.00, 120.00]	130.00 [120.00, 140.00]	<0.001	nonnorm
diastolic blood pressure (median [IQR])		80.00 [70.00, 80.00]	80.00 [80.00, 90.00]	<0.001	nonnorm
cholesterol (%)	normal	29330 (83.7)	23055 (65.9)	<0.001	
	well above normal	1892 (5.4)	6174 (17.7)		
	above normal	3799 (10.8)	5750 (16.4)		
glucose (%)	normal	30894 (88.2)	28585 (81.7)	<0.001	
	above normal	2112 (6.0)	3078 (8.8)		
	well above normal	2015 (5.8)	3316 (9.5)		
smoking (%)	no	31781 (90.7)	32050 (91.6)	<0.001	
	yes	3240 (9.3)	2929 (8.4)		
alcohol (%)	no	33080 (94.5)	33156 (94.8)	0.055	
	yes	1941 (5.5)	1823 (5.2)		
physical activity (%)	yes	28643 (81.8)	27618 (79.0)	<0.001	
	no	6378 (18.2)	7361 (21.0)		

Output of `print(tableOne)`. Image by author.

From the table, it can be observed that:

- **Categorical variables** are represented as **counts and percentages**.
- **Continuous variables** are displayed as:
  - In case of **normal** distribution: **mean and standard deviation**.
  - In case of **non-normal** distribution: **median and interquartile ranges**.

The table provides immediate and intuitive overview of the baseline characteristics stratified by cardiovascular disease.

It is possible to see the categorical or continous variables only, by accessing respectively to the `CatTable` and `ContTable` elements of the `TableOne` object. For example:

```
# categorical part only
df$CatTable
```

	Stratified by cardiovascular disease			
	no	yes	p	test
n	35021	34979		
gender = woman (%)	22914 (65.4)	22616 (64.7)	0.033	
cholesterol (%)			<0.001	
normal	29330 (83.7)	23055 (65.9)		
well above normal	1892 (5.4)	6174 (17.7)		
above normal	3799 (10.8)	5750 (16.4)		
glucose (%)			<0.001	
normal	30894 (88.2)	28585 (81.7)		
above normal	2112 (6.0)	3078 (8.8)		
well above normal	2015 (5.8)	3316 (9.5)		
smoking = yes (%)	3240 (9.3)	2929 (8.4)	<0.001	
alcohol = yes (%)	1941 (5.5)	1823 (5.2)	0.055	
physical activity = no (%)	6378 (18.2)	7361 (21.0)	<0.001	

Output of df$CatTable. Image by author.

3. Conclusions

In this post, we used four R packages that accomplish different EDA tasks, from summary tables to detailed HTML reports, and significantly ease the exploration of a new dataset.

R offers several packages with features that neatly and quickly summarize numerical and categorical data. We name a few: `skimr`[9], `Hmisc`[10], `desctable`[11], `summarytools`[12], `dlookr`[13].

Putatunda et al.[14] (2019) shared an insightful comparison between different packages providing EDA capabilities and available in CRAN:

Exploratory analysis features	SmartEDA	dlookr	DataExplorer	Hmisc	exploreR	RtutoR	summarytools
Describe basic information for input data	Y	Y	Y	Y			Y
Function to provide summary statistics for all numerical variable	Y	Y		Y	Y		Y
Function to provide plots for all numerical variable	Y		Y				Y
Function to provide summary statistics and plots for all character or categorical	Y	Y	Y	Y			
Function to provide plots for all character or categorical	Y		Y				Y
Customized summary statistics - extension of data.table package	Y						
Normality / Co-ordinate plots	Y	Y	Y				
Feature binarization / Binning		Y	Y				
Standardize /missing imputation / diagnose outliers		Y	Y		Y		
HTML report using rmarkdown / Shiny	Y	Y	Y			Y	

Comparison between the EDA features of different R packages. From Putatunda et al.[14] (2019).

For curiosity, we can observe the number of downloads for the cited packages in the last year (2021) as follows:

```
# install.packages("dlstats")
library("dlstats")

stats <- cran_stats(c("SmartEDA", "DataExplorer", "tableone",
"GGally", "Hmisc",
                      "exploreR", "dlookr", "desctable",
"summarytools"))

stats %>%
  filter(start >= "2021-01-01" & end < "2022-01-01") %>%
  select(package, downloads) %>%
  group_by(package) %>%
  summarize(downloads = sum(downloads)) %>%
  arrange(desc(downloads))
```

package <fctr>	downloads <int>
Hmisc	9037380
GGally	885701
summarytools	159334
DataExplorer	140822
tableone	95302
dlookr	42134
SmartEDA	17206
desctable	7897
exploreR	6017

9 rows

Image by author.

We can also observe the packages download trend over time:

```
# log10 scale is used as the difference between downloads is
comparatively large
ggplot(stats, aes(end, log10(downloads), group=package,
color=package)) +
       geom_line()
```

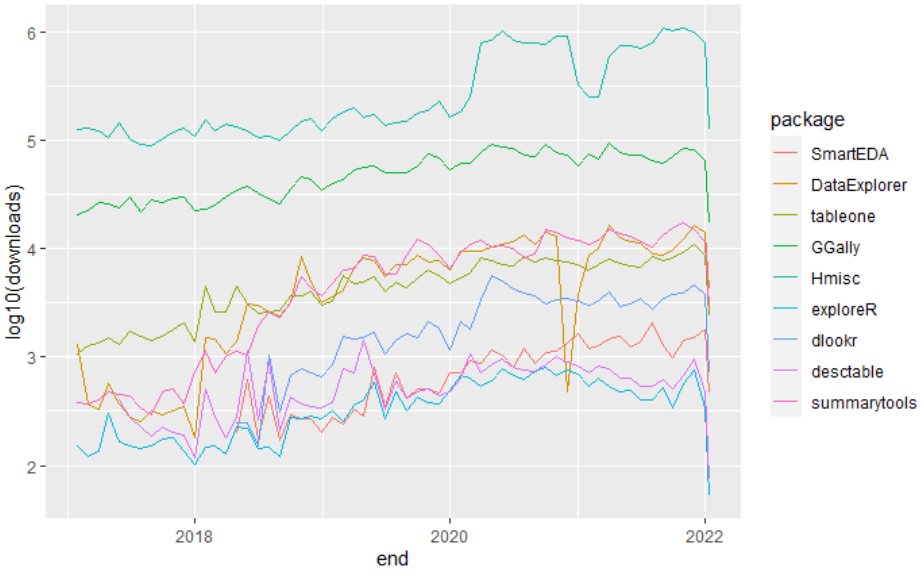


Image by author.

4. References

[1] kaggle.com/sulianova/cardiovascular-disease-dataset

[2] rdocumentation.org/packages/base/versions/3.6.2/topics/summary

[3] cran.r-project.org/package=DataExplorer

[4] cran.r-project.org/package=GGally

[5] rdocumentation.org/packages/GGally/versions/1.5.0/topics/ggpairs

[6] github.com/daya6489/SmartEDA

[7] cran.r-project.org/package=SmartEDA

[8] cran.r-project.org/package=tableone

[9] cran.r-project.org/package=skimr

[10] cran.r-project.org/package=Hmisc

[11] cran.r-project.org/package=desctable

[12] cran.r-project.org/package=summarytools

[13] cran.r-project.org/package=dlookr

[14] Putatunda, Sayan and Ubrangala, Dayananda and Rama, Kiran and Kondapalli, Ravi, "*SmartEDA: An R Package for Automated Exploratory Data Analysis*", Journal of Open Source Software, vol. 4, 2019, link.