tgm
Die Schule der Technik

JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

Systemtechnik Theorie

5CHIT 2020/21

SYT-Vertiefung

Explorative Datenanalyse

Jan R. Borensky

12. Januar 2021

Bewertung:		Version:	0.2
Betreuer:	Michael Borko	Begonnen:	17.11.2020
		Beendet:	12. Januar 2021

Inhaltsverzeichnis

1 Abstract

exploratory data analysis (in R and Python)

1.1 Introduction

EDA (exploratory data analysis) is used to understand different data sets by looking at their main characteristics and even plotting them visually to understand them better.

In this document, I would like to discuss the fundamental techniques as well as the concrete implementations of these techniques in R and Python.

Data Analysis is and will get much more important in the future. In 2007 it was assumed that the total quantity of data that was produced until then was about five exabytes ($5 \cdot 10^{18} bytes$) . At that point of time it was more than the estimated number of words spoken by humans. [12]

Because there are some open-source-software-solutions for nearly everything, there is also a solution for data analysis: the R programming language.

Basically there are three motivations for analyzing data:

> "1. to understand what has happened or what is happening;
> 2. to predict what is likely to happen, either in the future or in other circumstances we haven't seen yet;
> 3. to guide us in making decisions."
>
> [12, S. 1]

In order to analyze our data/the data of our company or something else, we have to begin with the simple sounding first step: understanding our data. And there it is necessary to use exploratory data analysis on every dataset we want to analyze in the future in order to understand it and make decision regarding certain predictions models for example.

When it comes to the size of data performance and speed is an important factor because the first step - loading the data - will take a lot of time if you use a slow implementation with a big data set. Therefore it is necessary to determine the time a certain implementation (R and Python) will take to complete the different exploratory data analysis steps.

1.2 Milestones

milestone	date
Introduction and Sources complete	17.11.2020
theoretical explanations complete	(~~01.12.2020~~) 12.01.2021
simple practical example complete	(~~08.12.2020~~) 12.01.2021
two more extensive examples and the comparsion of the complete	23.02.2020

2 Theoretische Ausführungen

2.1 Grundlagen

In diesem Kapitel werden die grundsätzlichen (theoretischen) Aspekte der explorativen Datenanalyse sowie die grundsätzliche Umsetzung von verschiedenen Methoden beziehungsweise den verbreitetsten Methoden.

Es gibt im Allgemeinen kein „Kochrezept" für Explorative Datenanalyse (EDA), das ein genaues Vorgehen vorgibt, da die zu verwendenden Methoden sehr von dem Datensatz abhängt, mit dem gearbeitet wird. Dennoch gibt es einige Grundsätze, die man anwenden kann, um sich sozusagen in den Datensatz einzuleben bzw. ein gewisses Verständnis für diesen Datensatz zu bekommen.

Im Folgenden werde ich daher auf zwei mögliche Herangehensweisen, die sich natürlich auch überschneiden, eingehen:

1. "the four R's of exploratory data analysis"
2. Peng's 10 Steps

Die **4 Schritte der explorativen Datenanalyse** stammen aus einem Buch, dass bereits 1991 veröffentlicht wurde. Trotzdem sind diese Ideen auch heute noch präsent.
Das erste R - *revelation* - bezieht sich auf die auf die Visualisierung eines Datensatzes in Bezug auf dessen Analyse.
Das zweite R - *residuals* - bezieht sich, wie der Name schon sagt, auf die tatsächlichen Werte in einem Datensatz und die mithilfe eines Modells generierten Prädiktionswerte und deren Differenzen.
Das dritte R - *re-expression* - bezieht sich auf die Umformung der Werte einer oder mehrerer Spalten bzw. Variablen eines Datensatzes zum besseren Verständnis.
Das vierte R - *resistance* - bezieht sich auf die Analyse eines Datensatzes unter Berücksichtigung der verschiedenen Arten von Beeinflussungen wie Aureißern oder Ähnlichem sowie dem Entfernen dieser Anomalien.

Die **10 Schritte für Explorative Datenanalyse**, die Roger Peng in seinem Buch *Exploratory Data Analysis with R* können durchaus als kleine Anleitung für den Einstieg in die EDA genutzt werden. Nachfolgend werde ich die einzelnen Schritte aufzählen und gewisse Informationen ergänzen bzw. kleine Beispiele hinzufügen. (Dazu wird sowohl das soeben genannte Buch [11] und die Website [4] verwendet.)

1. **Formulate your question**
   Grundsätzlich benutzt man die EDA oder grundsätzlich Datenanalyse dazu, um etwas über die Beschaffenheit der Daten herauszufinden, also um eine bestimmte Frage in Bezug auf die Daten zu beantworten. Das bedeutet, dass man sich grundsätzlich die Frage stellen muss, ob man die Daten für seine spezifische Frage verwenden kann, um sich dann mit seiner eigentlichen Frage oder den Fragen, die man zu dem Datensatz hat, beschäftige kann.

2. **Read in your data**
   Zuvor sollte festgehalten werden, dass man sich mit einem Datensatz nur dann gut beschäftigen kann, wenn dieser bereinigt wurde. Da das Bereinigen von Datensätzen jedoch sehr umfangreich werden kann, je nachdem wie ein Datensatz beschaffen ist, werden wir nachfolgend noch ein eigenes Unterkapitel dazu einfügen.
   Doch in jedem Fall müssen die Daten vor dem Bereinigen eingelesen werden. Sollte es möglich sein die Datentypen selbst zu erkennen, sollte man diese zuweisen, um ein falsches Interpretieren der jeweiligen Funktion zum Einlesen zu vermeiden.

3. **Check the packaging**
   Peng vergleicht bei diesem Schritt den Datensatz ein Geschenk, dass verpackt bleiben muss. Man will trotzdem wissen, was der Inhalt ist.
   Deswegen kann man hier ein folgende Dinge abrufen, um den Datensatz „von außen“ ein wenig zu analysieren:

   - Anzahl der Zeilen abrufen
   - Anzahl der Spalten abrufen

4. **Run str()** Also Fortsetzung des letzten Schrittes verwendet man nur detailliertere Informationen, um sich den Datensatz anzusehen:

   - Spaltennamen abrufen
   - Datentypen abrufen
   - Beispieldaten der Variablen abrufen

   (in `R` ganz einfach mit `str(dataset)`)

5. **Look at the top and the bottom of your data**
   Wenn man einen großen Datensatz hat und trotzdem sichergehen will, dass beim Einlesen alles funktioniert hat und der Datensatz somit in Ordnung ist, kann man sich die ersten fünf und die letzten fünf Zeilen des Datensatzes ansehen. Das ist natürlich keine Garantie für irgendetwas. Die Methode kann jedoch einen groben Überblick über die aktuelle Beschaffenheit geben, da zum Beispiel oft am Ende des Datensatzes Probleme beim Einlesen auftreten können.

6. **Check your „n“s**
   Dieser Schritt bezieht sich nicht auf leere Zellen (die in `R` mit `n` gefüllt werden), sondern auf die tatsächliche Anzahl von Beobachtungen und ob deren Variablen korrekt aufgezeichnet wurden. Trifft man hier auf Ungereimtheiten kann man natürlich weitere Abfragen mit den Datensatz durchführen, um sichergehen zu können, dass die entsprechenden Zeilen alle korrekte Daten enthalten und somit auch weiterverwendet werden können.

7. **Validate with at least one external data source**
   Dieser Schritt ist natürlich sinnvoll, wenn man Datensätze hat, die öfter vorkommen. Das bedeutet, dass es hilfreich ist, wenn man seinen eigenen Datensatz mit einem externen Datensatz nochmal überprüft. Das ist jedoch meiner Erfahrung nach nicht immer möglich bzw.

gibt es auch immer wieder selbst erstellte Datensätze, die man selbst oder ein Auftraggeber erstellt hat, die dann auch einzigartig sind.

8. **Try the easy solution first**
   In Bezug auf seine in Punkt 1 formulierte Frage, kann man nun versuchen die einfachsten Methoden zu verwenden, um die gewünschte Frage zu beanworten.

9. **Challenge your solution**
   Nachdem man eine Lösung gefunden hat, sollte man diese nochmal gründlich analysieren und auf etwaige Vearbeitungsfehler prüfen. Das machte es wesentlich einfacher schneller zu einem Ergebnis zu kommen, da man schon eine grundlegende Lösung hat, mit der man arbeiten kann.

10. **Follow up**
    Natürlich kann man sich nach der Analyse oder währenddessen auch immer wieder folgende drei wichtige Fragen stellen:

    „

    - Do you have the right data?
    - Do you need other data?
    - Do you have the right question?

    “ [11]

2.2 Bereinigen von Datensätzen

Wie vorhin schon angedeutet, werden wir uns in diesem Unterkapitel kurz mit dem Bereinigen von Datensatz beschäftigen.

Grundsätzlich sollte beim Bereinigen eines Datensatzes auf folgende Dinge geachtet werden:

- Entfernen von leeren Spalten
- Entfernen von leeren Zeilen
- Richtiges Zuweisen der Datentypen
- Entfernen von falsch eingetragenen Werten (anderer Datentyp oÄ)

In R verwendet man dazu zB die Packages `tidyr` und `tidyverse`.

2.3 Python

2.3.1 Voraussetzungen

Folgende Python-Pakete werden allgemein für EDA verwendet:

- `pandas`
- `numpy`
- `seaborn`
- `matplotlib.pyplot`

2.4 R

2.4.1 Voraussetzungen

n. Unter anderem folgende:

- `ggplot2`
- `dplyr`
- `tidyr`
- `broom`

In R gibt es die Möglichkeit, dass man die Packages direkt über `CRAN` oder manuell über `GitHub` installiert:

```r
# Install from CRAN
install.packages("tidyverse")

# Or the development version from GitHub
# install.packages("devtools")
devtools::install_github("tidyverse/tidyverse")
```

Auflistung 1: Laden von `tidyverse`

3 Praktische Beispiele

3.1 Datensatz

Zur Darstellung der Methoden von EDA wird der Datensatz `titanic` verwendet. Dieser beinhaltet Informationen über die Sterblichkeit von Passagieren auf der Titanic, wobei folgende Variablen beziehungsweise Merkmale der Passagiere vorhanden sind:

- economic status (class)
- sex
- age
- survival

Dieser Datensatz wird oft dafür verwendet, um Machine Learning - Algorithmen zu trainieren, da zusätzlich zu dem Datensatz der in R verfügbar ist auch ein Trainings- und ein Testing-Datensatz zur Verfügung steht.

Nachfolgend sind die Output-Dateien der beiden Beispiele eingefügt. Da diese durch die jupyter-spezifische Formatierung nur schwer inkludiert werden konnte, kann man sich auch die PDF oder Markdown-Exports im Ordner `Output-Example` [14] ansehen.

1 Beispiel mit Python

1.1 Imports

[1]:
```
import time
start = time.time()
## imports
# Importing required libraries for EDA
import pandas as pd
import numpy as np
import seaborn as sns # visualisation
import matplotlib.pyplot as plt # visualisation
#matplotlib inline
sns.set(color_codes=True)

# additonal libraries
from pathlib import Path
```

1.2 Einfaches Beispiel

1.2.1 Daten einlesen und anzeigen

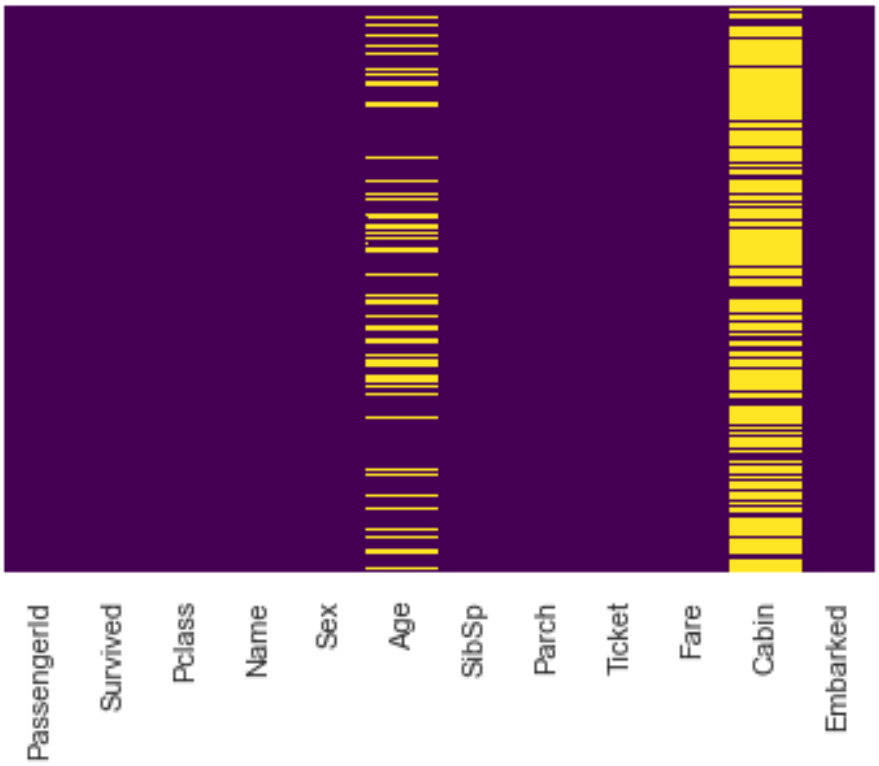
[2]:
```
path = "train.csv"
train=pd.read_csv(path)
print(train.head())
```

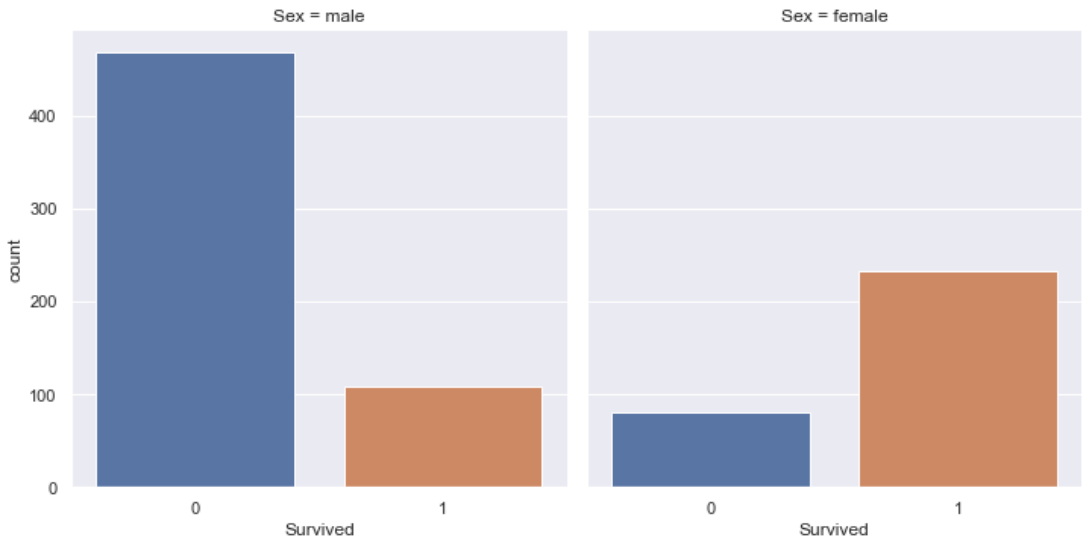
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0
```

	Parch	Ticket	Fare	Cabin	Embarked
0	0	A/5 21171	7.2500	NaN	S
1	0	PC 17599	71.2833	C85	C
2	0	STON/O2. 3101282	7.9250	NaN	S
3	0	113803	53.1000	C123	S
4	0	373450	8.0500	NaN	S

1.2.2 einfache Plots

[3]:
```
sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
sns.catplot(x='Survived',col='Sex',kind='count',data=train) # factorplot ->␣
 ↪catplot
plt.show()
```





1.2.3 Datensatz analysieren

[4]:
```
# Datentypen analysieren
train.dtypes

# Auf Duplikate prüfen
duplicate_rows_df = train[train.duplicated()]

# Folgende Ausgabe sollte die Anzahl der Duplikate in unserem Datensatz␣
 ↪anzeigen.
# In unserem Fall 0
print("number of duplicate rows: ", duplicate_rows_df.shape, "\n")

# Auf null-Values prüfen
print(train.count())
print("\n")
print(train.isnull().sum())
```

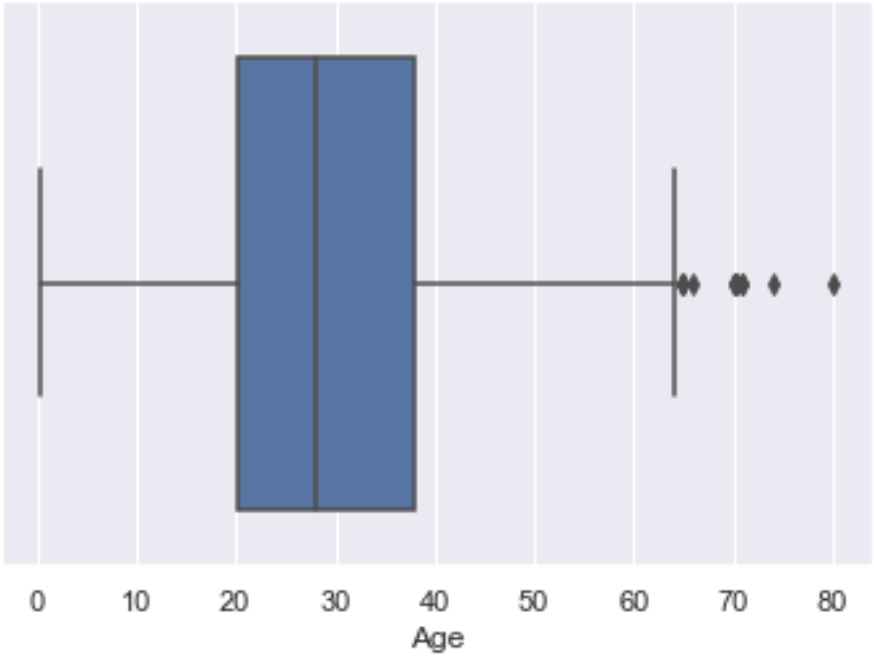
```
number of duplicate rows:  (0, 12)

PassengerId    891
Survived       891
Pclass         891
Name           891
Sex            891
Age            714
SibSp          891
Parch          891
Ticket         891
```

```
Fare           891
Cabin          204
Embarked       889
dtype: int64


PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

1.2.4 Boxplots erstellen

[5]:
```
sns.boxplot(x=train['Age'])
```

[5]:
```
<AxesSubplot:xlabel='Age'>
```



1.2.5 Kennwerte berechnen

[6]:
```
# Interquartilsdistanz
Q1 = train.quantile(0.25)
Q3 = train.quantile(0.75)
IQR = Q3 - Q1
print(IQR)

# Mittelwerte
train.mean()
```

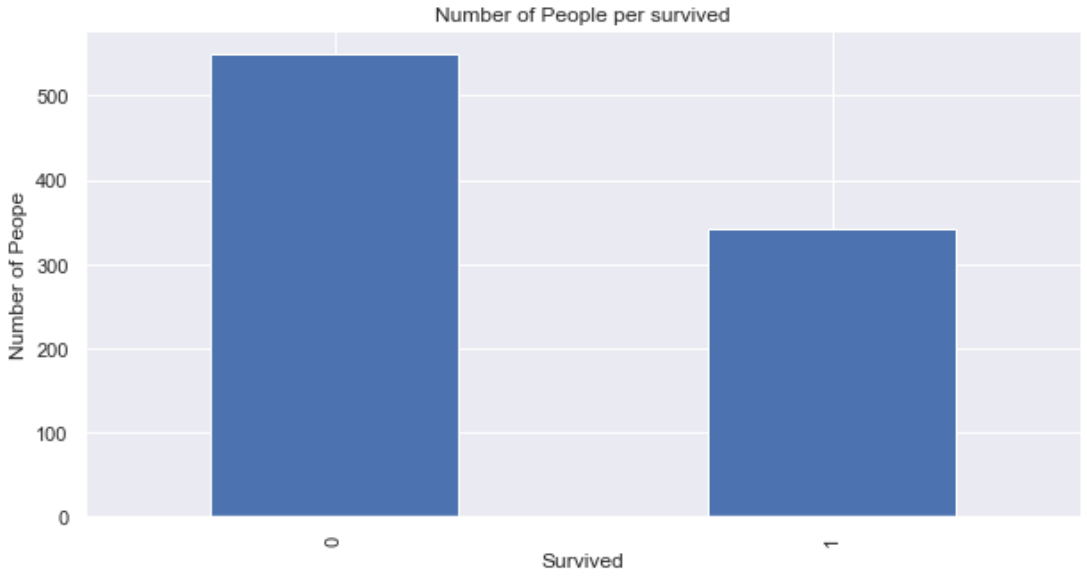
```
PassengerId    445.0000
Survived         1.0000
Pclass           1.0000
Age             17.8750
SibSp            1.0000
Parch            0.0000
Fare            23.0896
dtype: float64
```

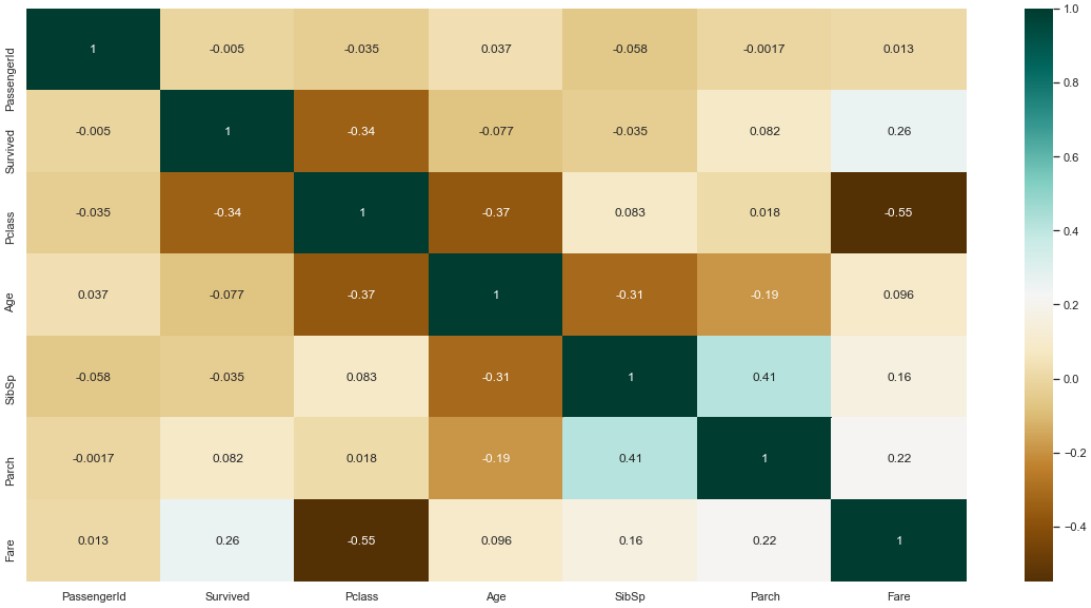
[6]:
```
PassengerId    446.000000
Survived         0.383838
Pclass           2.308642
Age             29.699118
SibSp            0.523008
Parch            0.381594
Fare            32.204208
dtype: float64
```

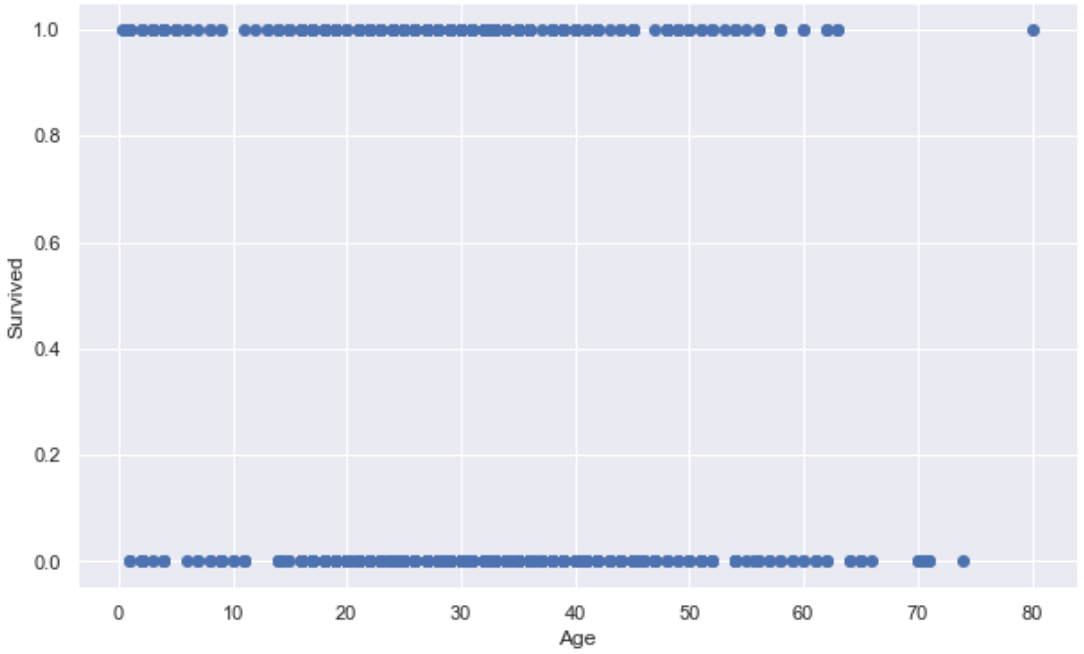
1.2.6 verschiedene Plots

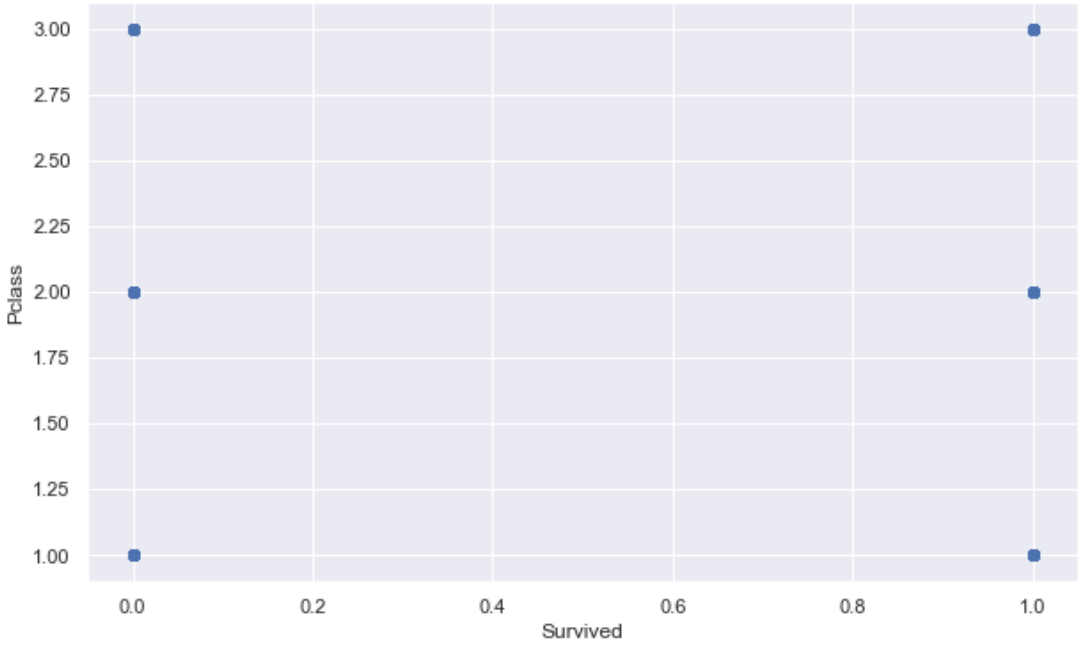
[7]:
```
# Plotting histogram
# Plotting a Histogram
train.Survived.value_counts().nlargest(40).plot(kind='bar', figsize=(10,5))
plt.title("Number of People per survived")
plt.ylabel('Number of Peope')
plt.xlabel('Survived');

# Plotting heat maps
# Finding the relations between the variables.
plt.figure(figsize=(20,10))
c= train.corr()
sns.heatmap(c,cmap="BrBG",annot=True)
c

# Plotting a scatter plot
fig, ax = plt.subplots(figsize=(10,6))
ax.scatter(train['Age'], train['Survived'])
```

```
ax.set_xlabel('Age')
ax.set_ylabel('Survived')
plt.show()

# Plotting a scatter plot
fig, ax = plt.subplots(figsize=(10,6))
ax.scatter(train['Survived'], train['Pclass'])
ax.set_xlabel('Survived')
ax.set_ylabel('Pclass')
plt.show()
```




PassengerId
Survived
Pclass
Age
SibSp
Parch
Fare
1 -0.005 -0.035 0.037 -0.058 -0.0017 0.013
-0.005 1 -0.34 -0.077 -0.035 0.082 0.26
-0.035 -0.34 1 -0.37 0.083 0.018 -0.55
0.037 -0.077 -0.37 1 -0.31 -0.19 0.096
-0.058 -0.035 0.083 -0.31 1 0.41 0.16
-0.0017 0.082 0.018 -0.19 0.41 1 0.22
0.013 0.26 -0.55 0.096 0.16 0.22 1
PassengerId Survived Pclass Age SibSp Parch Fare
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4


Survived
1.0
0.8
0.6
0.4
0.2
0.0
0 10 20 30 40 50 60 70 80
Age



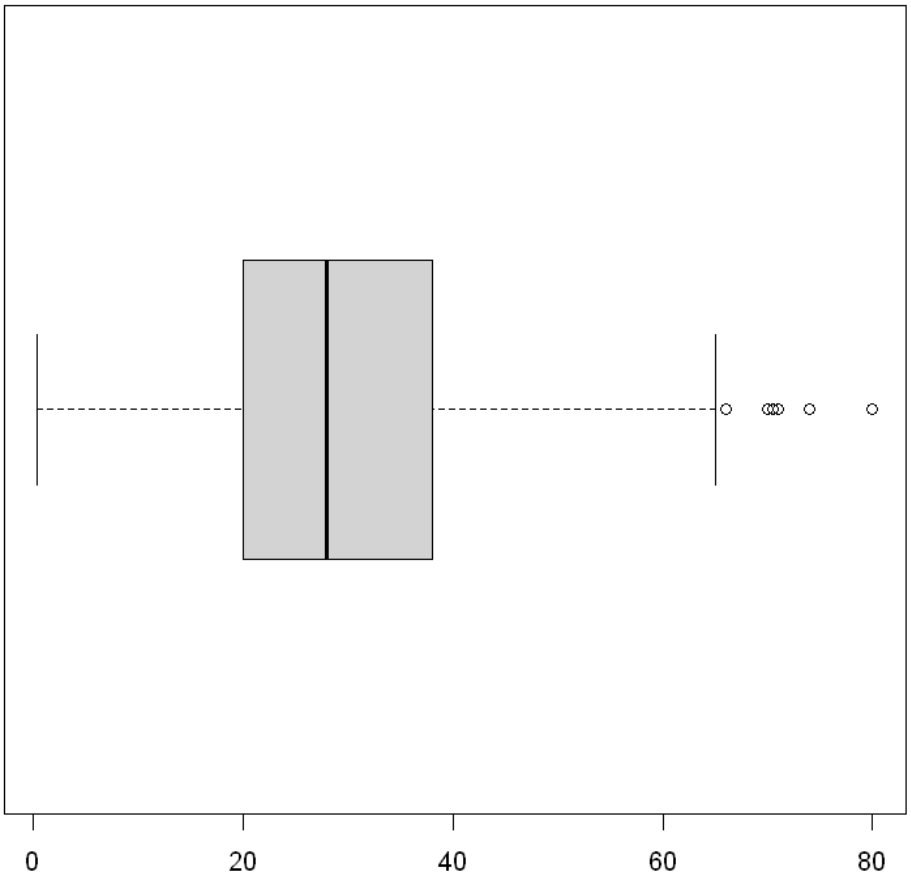
[8]:
```
ende = time.time()
print('{:5.3f}s'.format(ende-start))
```

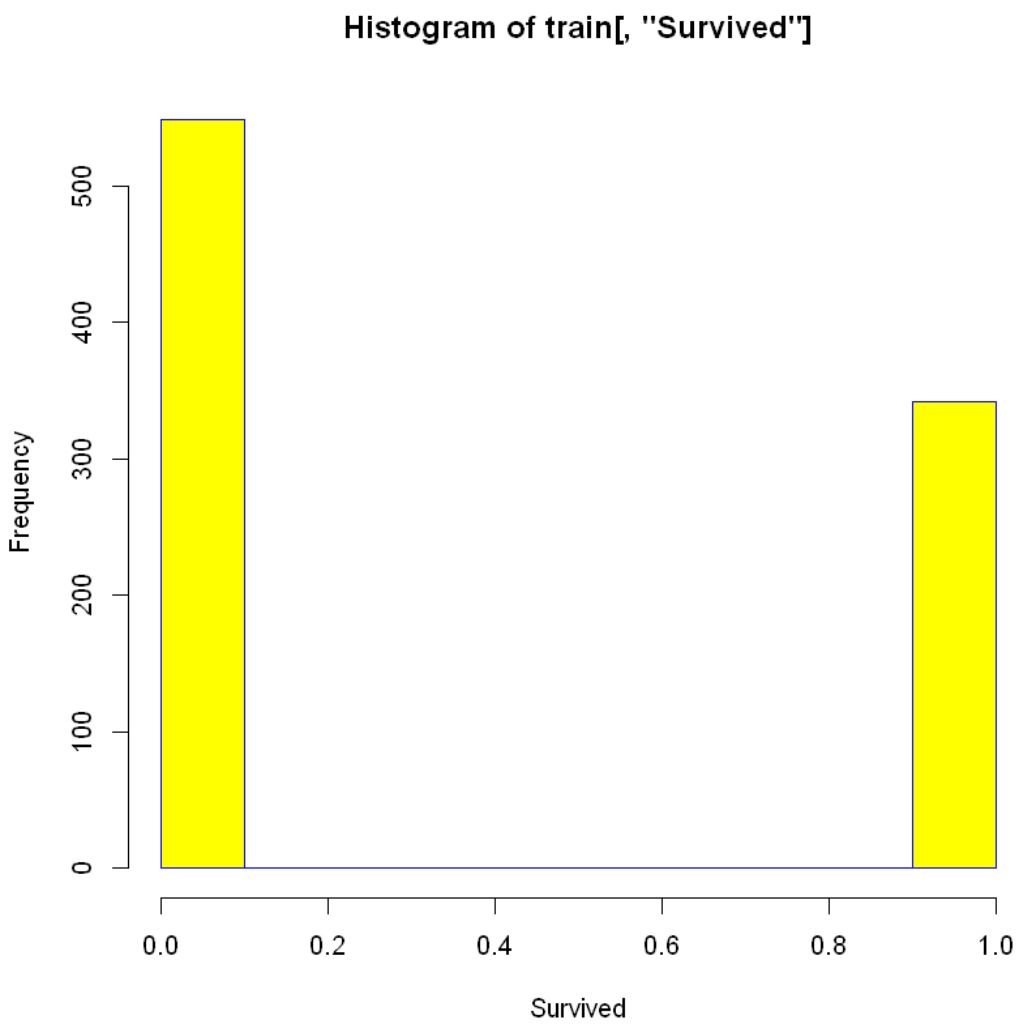
```
15.984s
```

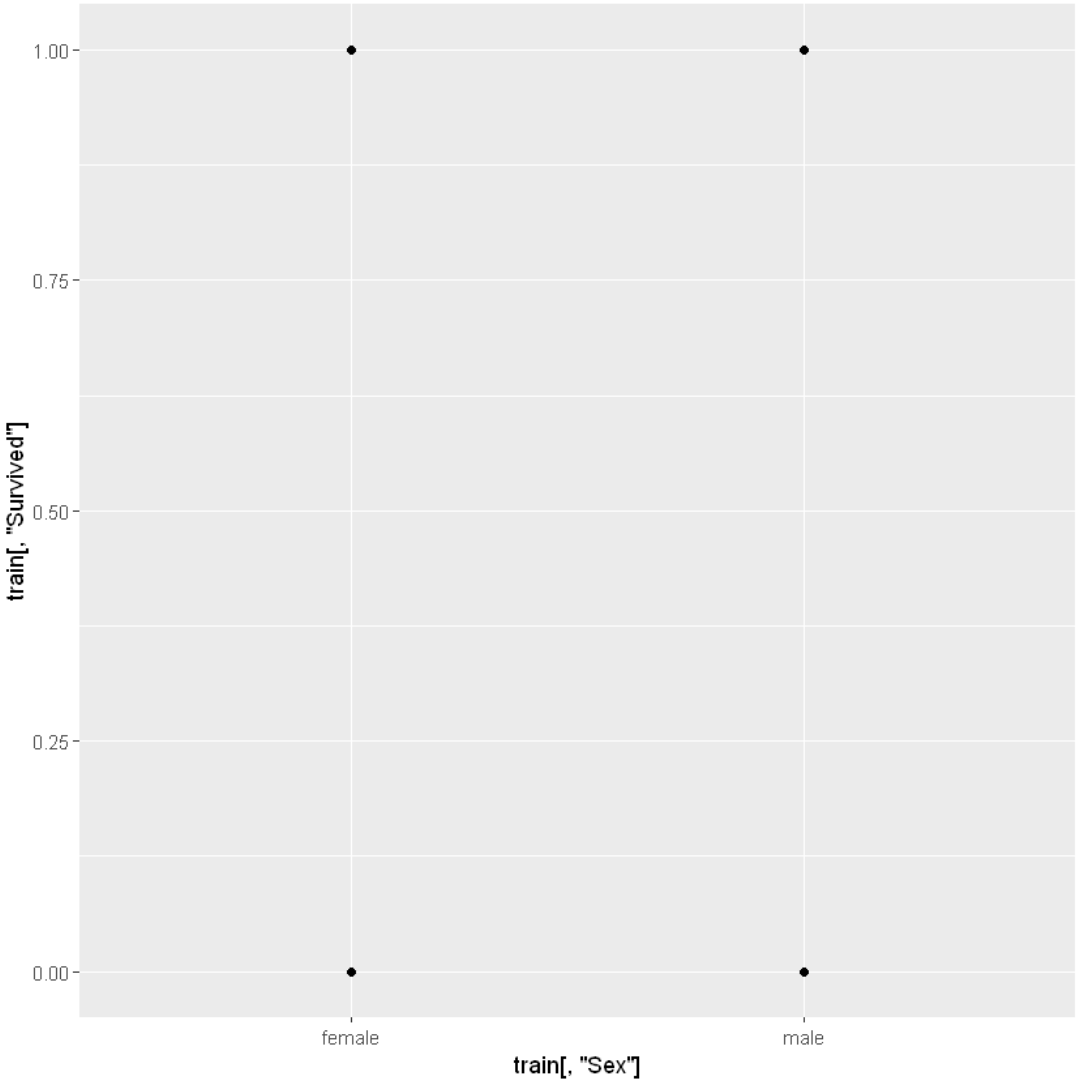
[1]:
```r
start_time <- Sys.time()

library(ggplot2)
library(reshape2)

path = "train.csv"
train=read.csv(path)
head(train)
#train = as.matrix(train)

# heatmap(as.matrix(train))
boxplot(train['Age'], horizontal=TRUE)

hist(train['Survived'],xlab = "Survived",col = "yellow",border = "blue")

ggplot(train, aes(x=train['Sex'], y=train['Survived'])) + geom_point()

cormat <- round(cor(train[sapply(train, is.numeric)]),2)
melted_cormat <- melt(cormat)
ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile()
```

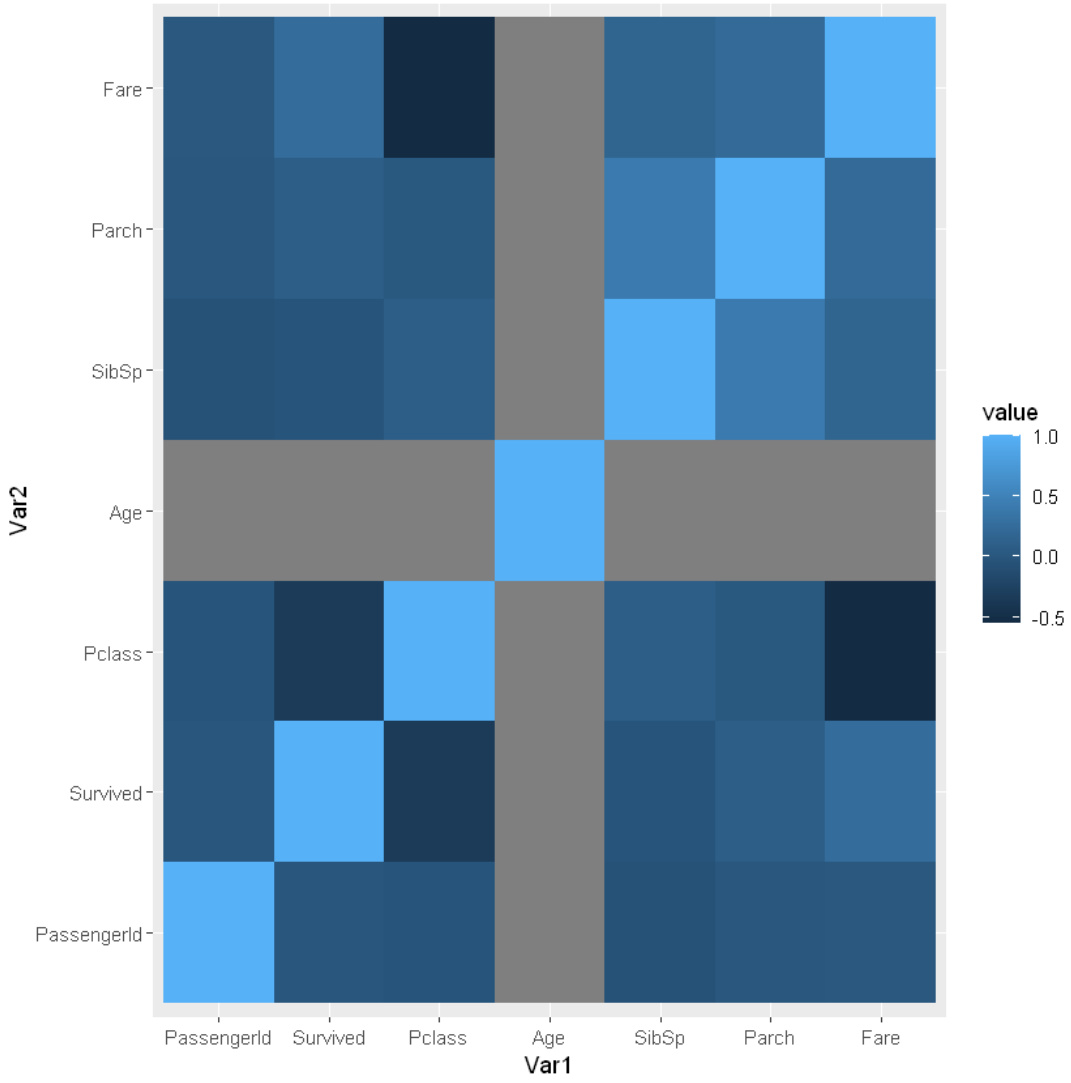
A data.frame: 6 × 12

	PassengerId <int>	Survived <int>	Pclass <int>	Name <chr>
1	1	0	3	Braund, Mr. Owen Harris
2	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thaye
3	3	1	3	Heikkinen, Miss. Laina
4	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)
5	5	0	3	Allen, Mr. William Henry
6	6	0	3	Moran, Mr. James


0
20
40
60
80


Histogram of train[, "Survived"]
Frequency
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0
Survived


1.00
0.75
0.50
0.25
0.00
train[, "Survived"]
female
male
train[, "Sex"]



[2]:
```
end_time <- Sys.time()
end_time - start_time
```

```
Time difference of 3.018706 secs
```

Glossar

Explorative Datenanalyse „Die explorative Datenanalyse (EDA) oder explorative Statistik ist ein Teilgebiet der Statistik. Sie untersucht und begutachtet Daten, von denen nur ein geringes Wissen über deren Zusammenhänge vorliegt." [6]. 24

Akronyme

EDA Explorative Datenanalyse. 5, 9, 10, 24

Literaturverzeichnis

[1] 7 Exploratory Data Analysis | R for Data Science. URL: https://r4ds.had.co.nz/exploratory-data-analysis.html (besucht am 11. 01. 2021).

[2] 7 Exploratory Data Analysis | R for Data Science. URL: https://r4ds.had.co.nz/exploratory-data-analysis.html (besucht am 01. 12. 2020).

[3] Wikipedia Autoren. Systemtechnik. 07.03.2018. Wikipedia. URL: https://de.wikipedia.org/wiki/Systemtechnik.

[4] Chapter 4 Exploratory Data Analysis | Data Analysis in R. URL: https://bookdown.org/steve_midway/DAR/exploratory-data-analysis.html (besucht am 11. 01. 2021).

[5] Easily Install and Load the Tidyverse • tidyverse. URL: https://tidyverse.tidyverse.org/ (besucht am 01. 12. 2020).

[6] Explorative Datenanalyse – Wikipedia. URL: https://de.wikipedia.org/wiki/Explorative_Datenanalyse (besucht am 01. 12. 2020).

[7] Exploratory data analysis in Python. | by Tanu N Prabhu | Towards Data Science. URL: https://towardsdatascience.com/exploratory-data-analysis-in-python-c9a77dfa39ce (besucht am 17. 11. 2020).

[8] How to display a Seaborn plot in Python. URL: https://www.kite.com/python/answers/how-to-display-a-seaborn-plot-in-python (besucht am 01. 12. 2020).

[9] How to Export DataFrame to CSV in R. URL: https://datatofish.com/export-dataframe-to-csv-in-r/.

[10] How To Insert Jupyter and IPython Notebooks in Authorea articles. URL: https://www.authorea.com/users/9932/articles/11070 (besucht am 12. 01. 2021).

[11] D. Peng Roger. Exploratory Data Analysis with R. 2020. URL: https://leanpub.com/exdata.

[12] Ronald K. Pearson. Exploratory Data Analysis Using R. eng. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press. ISBN: 9781138480605. DOI: 10.1201/9781315382111.

[13] Step By Step Exploratory Data Analysis Of Titanic DataSet | by Aditya Mohanty | Data Driven Investor | Medium. URL: https://medium.com/datadriveninvestor/step-by-step-exploratory-data-analysis-of-titanic-dataset-2d0fb09b0e86 (besucht am 01. 12. 2020).

[14] TGM-HIT/syt-vertiefung-jborensky-tgm at master. URL: https://github.com/TGM-HIT/syt-vertiefung-jborensky-tgm/tree/master (besucht am 12. 01. 2021).

[15] Titanic: Machine Learning from Disaster | Kaggle. URL: https://www.kaggle.com/c/titanic (besucht am 01. 12. 2020).

[16] titanic package | R Documentation. URL: https://www.rdocumentation.org/packages/titanic/versions/0.1.0 (besucht am 01. 12. 2020).

[17] Hadley Wickham und Garrett Grolemund. R for data science : import, tidy, transform, visualize, and model data. eng. First edit. Beijing Boston Farnham Sebastopol Tokyo: O'Reilly, 2017. ISBN: 1491910399.

Auflistungsverzeichnis