Portfolio 1 - assignment

Christian Laursen (bfd962)

Mickey Johansson (mvx394)

Mushtaba Osmani (wgq323)

5123 characters - 2,13 normal pages

17/12 - 19

1 - 2) Open and inspect the content of the file and import the data to a dataframe.

By opening the given file in a text-editor (we used the program intended for Python called *Spyder*), it was possible to obtain an overview of the data and data types present and/or missing from the file. To describe the dataset present in the file, we imported it to a dataframe. A dataframe is data structure with labeled axes (columns and rows) (Pydata, n.d). Next, the content of the file was described by printing built-in function according to “df_titan”. For example, using “df_titan.dtypes” one can find out that “integer”, “float” and “object” are used in the dataset. “Integer”, a numerical value without decimals, is used in survivors or not, price range and the number of family members etc. on board. “Float”, a numerical value with decimals, used in age and fares. Finally, “Object” occurred when the column contained “strings” - text, used in names and genders. By using “df_titan.columns”, it is possible to find out the names of the different columns and thus find out what information one has on their hands when working with the dataset. These are the following: ‘Survived’, 'Pclass’, ‘Name’, ‘Sex’, ‘Age’, ‘Siblings / Spouses Aboard’, ‘Parents / Children Aboard’, ‘Fare’. Regarding missing data, one can take a look at the ‘Fare’ column. ‘This includes multiple instances with the value 0. This may indicate a lack of data, but by researching some of the names, it is revealed that many of them were employees of the Titanic, which most likely means they did not have to pay for a ticket. All the different built-in functions used, separated by a line break, can be seen in the screenshot below:

```
# Antallet af rækker i datasættet
print("Antal rækker i datasættet: ",len(df_titan), "\n")

# Antallet af rækker og kolonner i datasættet
print("Antal rækker og kolonner i datasættet: ",df_titan.shape, "\n")

# Antallet af celler i tabellen
print("Antal celler i datasættet: ",df_titan.size, "\n")

# Kolonnernes navne
print("Kolonnernes navne:\n",df_titan.columns, "\n")

# Kolonnernes datatyper
print("Kolonnernes datatyper:\n",df_titan.dtypes, "\n")
```

3) Calculating the data and making descriptive statistics.

Now we are tasked with extracting data and producing descriptive statistics. We firstly wanted to figure out the amount of survivors on the Titanic ship, which we did by taking the sum of the 'Survived' column as this will count the amount of people that survived the accident. The final number we got was then 342 survivors.

```
59 # Antal der overlevede
60 print("Antal der overlevede:","\n",df_titan['Survived'].sum(),"\n")
61
```

```
Antal der overlevede:
 342
```

Afterward, we ran functions such as mean and median in python which we ran through the 'Age' column. The mean age was 29.4 and the median was 28 years old of the passengers on the Titanic.

```
62 # Gennemsnitsalder
63 print("Gennemsnitsalderen:","\n",df_titan['Age'].mean(),"\n")
64
65 # Alders medianen
66 print("Alders medianen:","\n",df_titan['Age'].median(),"\n")
```

Lastly, we did a count of the different sexes on the ship. We did this by using 'groupby' function of and then using the count function in python. This gave us an interesting result of 314 female passengers and 573 male passengers.

```
68 # Antal mænd og kvinder
69 print("Antal mænd og kvinder:","\n",df_titan.groupby('Sex')['Sex'].count(),"\n")
70
```

4) Examining people with the same last name

An interesting aspect of the Titanic dataset is to examine whether or not there were passengers with the same last name, to figure out whether there traveling families or married couples. To do this we first had to save the 'Name' column to a variable, which we call 'namelist'. We then create an empty list, 'lastnames' that will be used to append all last names.

```
80 namelist = df_titan['Name'] # Gemmer datafilens navne i en liste
81 lastnames = [] # Tom liste til at indeholde efternavne
```

To extract all last names we make a for loop that will run through all last names in our 'namelist' variable. We split all the names and extract only the last name that we then append into our 'lastnames' list.

```
for lastname in namelist:
    name = lastname.split(" ")[-1:] # Deler alle navnene ved mellemrum, gemmer kun efternavnet
    lastnames.append(name[0]) # Indsætter efternavnet i min liste til efternavne
```

Now that we have a list with all the last names, we create a new dataframe that we call 'df_lastnames'. The reasoning for this is to then utilize value_copunts of that dataframe, which will then produce the most frequent last names on the Titanic. lastly, we use 'head(10)' to limit our results to the 10 most frequent last names. The most frequent last name on the Titanic was Andersson with 9 passengers sharing the same last name.

```
# Jeg gemmer listen med navne i en dataframe, for at kunne bruge en pandas funktion
df_lastnames = pd.DataFrame(lastnames)

# Antal forekomster af hvert enkelt efternavn
print("De 10 mest hyppige efternavne:\n",df_lastnames[0].value_counts().head(10),"\n")
```

5) Pivot-table of price classes

By using the *value_counts()* method on the 'Pclass' column in our dataframe, we get a pivot-table showing the amount of people in each of the three different price classes. The value_counts() method counts the occurrences of the different values in the specified columns. In this instance the values in the 'Pclass' column are 1, 2 or 3, each representing a price class.

```
# Antal personer i hver prisklasse
antal = df_titan['Pclass'].value_counts()
print("Antal personer i hver prisklasse:\n",antal,"\n")
```

The table shows that price class 3 has the most people, and price class 2 the fewest number of people.

```
Antal personer i hver prisklasse:
 3    487
1    216
2    184
Name: Pclass, dtype: int64
```

Which price class had the most casualties?

To examine how many people of each price class that died, we use the *groupby()* method to sort the 'Pclass' column into groups with people of each price class, and then again use the value_counts() method to count the occurrences of different values in the 'Survived' column of each of these groups. People who survived have 1 in the 'Survived' column and people who did not survive have a 0.

```
# Antal overlevende og døde i hver prisklasse
print("Antal overlevende og døde i hver prisklasse:\n (0 = døde, 1 = overlevende) \n",df_titan.groupby('Pclass')['Survived'].value_counts())
```

This gives us a pivot-table showing the amount of people who survived and died in each price class. The table shows that price class 3 had the most casualties, while class 1 had fewest casualties.

```
Antal overlevende og døde i hver prisklasse:
 (0 = døde, 1 = overlevende)
 Pclass  Survived
1       1           136
        0            80
2       0            97
        1            87
3       0           368
        1           119
Name: Survived, dtype: int64
```

We have also calculated the percentages of people who survived and died in each price class. This was done by dividing the amount of survivors and casualties in each class by the total amount of people in that class and multiplying by 100.

```
procent = df_titan.groupby('Pclass')['Survived'].value_counts()
# Procentdelen for døde og overlevende i hver prisklasse:
print('Procentdel døde og overlevende i prisklasse 1: ',procent[1]/antal[1]*100)
print('Procentdel døde og overlevende i prisklasse 2: ',procent[2]/antal[2]*100)
print('Procentdel døde og overlevende i prisklasse 3: ',procent[3]/antal[3]*100)
```

The results show that more than 75% of people in price class 3 did not survive. It was close to 50/50 in price class 2, while only 37% of people in price class 1 did not survive.

```
Procentdel døde og overlevende i prisklasse 1:
 Survived
1    62.962963
0    37.037037
Name: Survived, dtype: float64
Procentdel døde og overlevende i prisklasse 2:
 Survived
0    52.717391
1    47.282609
Name: Survived, dtype: float64
Procentdel døde og overlevende i prisklasse 3:
 Survived
0    75.564682
1    24.435318
Name: Survived, dtype: float64
```

Bibliography:

Pydata (n.d). pandas.DataFrame. Retrieved the 9th of December 2019 from: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html