final_assignment

February 9, 2023

1 Affordability of a healthy diet and BMI

[1]:
```
# general imports
import yaml
import pandas as pd
import numpy as np
import re

# plotting imports
from bokeh.plotting import output_notebook, figure
from bokeh.io import show
from bokeh.models import ColumnDataSource, FactorRange, Whisker
from panel.widgets import MultiSelect
from bokeh.transform import factor_cmap
import matplotlib.pyplot as plt

# geo-map imports
import folium
from branca.colormap import linear

# statistic imports
from scipy.stats import shapiro, norm
import statsmodels.api as sm

# panel imports
import panel as pn
from Panel import panel_text as txt
from Panel.dashboard_module import Dashboard  # self-made module to create␣
↪dashboards

output_notebook()
pn.extension()
```

1.1 1. The project

In Europe and Northern America 17.3 billion people can not afford a healthy diet. It is known that prices rise with the increase in the level of diet quality. In general, a healthy diet is 60% more expensive than an energy sufficient diet [1][2]. Therfore it can be speculated that the affordability of a healthy diet may lead to poor dietery decisions. These poor dietery decisions may be reflected on body mass index (BMI). In this project the relationship between the affordability of a healthy diet and BMI is studied. The hypothesis is that the mean BMI increases when a healthy diet becomes less affordable.

[1] https://www.fao.org/3/cb4474en/cb4474en.pdf
[2] https://bmcpublichealth.biomedcentral.com/articles/10.1186/s12889-016-2996-y

1.2 2. Data acquisition

The data used in this project is obtained from two different data sources. The BMI data is obtained from https://www.ncdrisc.org/data-downloads-adiposity.html, where the 'country-specific data for all countries' should be clicked. It can then be downloaded in the csv format. This data contains out of all of the mean BMI's (with 95% confidence intervals) per country per sex, ranging from the year 1975 to 2016. In addition, the dataset contains the prevalence of certain BMI ranges (with 95% confidence intervals). For example, a person is classified as obese if the BMI is higher or equal to 30.

The affordability of a healthy diet is obtained from https://databank.worldbank.org/source/food-prices-for-nutrition (select all under country, select 'Affordability of a healthy diet: ratio of cost to food expenditures' under series, select 2017 under year) and downloaded in the csv format. Extract the zip and use the _data file (not the meta-data file). In this dataset the affordability of a healthy diet is defined as the ratio of the cost of a healthy diet to food expenditures. This ratio is given by country and by sex in the year 2017.

Keep in mind that the BMI dataset is from the year 2017 and the healthy diet affordability dataset is from 2016. At this point in time, the global BMI data is only available up to 2016 and the healthy diet data from 2017 until 2020. In this project the 2017 and 2016 will be compared. The mean BMI and heatlhy diet affordability are unlikely to change a lot from year to year and therefor are compared. However, when 2017 BMI data is available it is better to use that instead of the 2016 data.

1.3 3. Loading and cleaning the data

The dataset are loaded into a pandas dataframe using a config file

[2]:
```python
# load the datafiles using a config files
with open("./config.yaml", 'r') as stream:
    config = yaml.safe_load(stream)

bmi = config['bmi']
healthy_diet_affordability = config['food_affordability']
```

1.3.1 3.1 BMI

[3]:
```python
df_bmi = pd.read_csv(bmi, encoding='latin-1') # file uses latin-1 encoding
df_bmi = df_bmi[df_bmi.Year == 2016]  # we only want the 2016 data
df_bmi = df_bmi.drop(columns='Year')  # remove the year column, we only have␣
↪2016 left so redundant
df_bmi.head()
```

[3]:
```
    Country/Region/World  ISO  Sex   Mean BMI  \
41           Afghanistan  AFG  Men  22.682456
83               Albania  ALB  Men  27.174471
125              Algeria  DZA  Men  24.865386
167       American Samoa  ASM  Men  33.066721
209              Andorra  AND  Men  27.478395

     Mean BMI lower 95% uncertainty interval  \
41                                 20.157475
83                                 25.975170
125                                23.487321
167                                31.338678
209                                24.988831

     Mean BMI upper 95% uncertainty interval  \
41                                 25.241857
83                                 28.338256
125                                26.220294
167                                34.662447
209                                30.001977

     Prevalence of BMI>=30 kg/mï¿½ (obesity)  \
41                                  0.033603
83                                  0.223735
125                                 0.206662
167                                 0.587546
209                                 0.267498

     Prevalence of BMI>=30 kg/mï¿½ lower 95% uncertainty interval  \
41                                            0.013884
83                                            0.153334
125                                           0.141854
167                                           0.502606
209                                           0.186223

     Prevalence of BMI>=30 kg/mï¿½ upper 95% uncertainty interval  \
41                                            0.066334
83                                            0.300834
125                                           0.279979
167                                           0.666355
209                                           0.354723

     Prevalence of BMI>=35 kg/mï¿½ (severe obesity)  …  \
41                                         0.003314  …
83                                         0.045036  …
125                                        0.042840  …
167                                        0.322678  …
```

```
209                                        0.068565  …

     Prevalence of BMI 25 kg/mï¿½ to <30 kg/mï¿½ upper 95% uncertainty interval
\
41                                                  0.242503
83                                                  0.515957
125                                                 0.463570
167                                                 0.370043
209                                                 0.532478

     Prevalence of BMI 30 kg/mï¿½ to <35 kg/mï¿½  \
41                                            0.030290
83                                            0.178699
125                                           0.163822
167                                           0.264868
209                                           0.198934

     Prevalence of BMI 30 kg/mï¿½ to <35 kg/mï¿½ lower 95% uncertainty interval
\
41                                                  0.011207
83                                                  0.113401
125                                                 0.102610
167                                                 0.191211
209                                                 0.125035

     Prevalence of BMI 30 kg/mï¿½ to <35 kg/mï¿½ upper 95% uncertainty interval
\
41                                                  0.062681
83                                                  0.252200
125                                                 0.234048
167                                                 0.340031
209                                                 0.280761

     Prevalence of BMI 35 kg/mï¿½ to <40 kg/mï¿½  \
41                                            0.002271
83                                            0.037684
125                                           0.031750
167                                           0.183871
209                                           0.052701

     Prevalence of BMI 35 kg/mï¿½ to <40 kg/mï¿½ lower 95% uncertainty interval
\
41                                                  0.000310
83                                                  0.013616
125                                                 0.011066
167                                                 0.109862
209                                                 0.017627
```

```
     Prevalence of BMI 35 kg/mï¿½ to <40 kg/mï¿½ upper 95% uncertainty interval
\
41                                                0.007487
83                                                0.076984
125                                               0.065164
167                                               0.263723
209                                               0.109639

     Prevalence of BMI >=40 kg/mï¿½(morbid obesity)  \
41                                             0.001043
83                                             0.007352
125                                            0.011090
167                                            0.138807
209                                            0.015864

     Prevalence of BMI >=40 kg/mï¿½ lower 95% uncertainty interval  \
41                                                0.000074
83                                                0.001180
125                                               0.002289
167                                               0.067651
209                                               0.002652

     Prevalence of BMI >=40 kg/mï¿½ upper 95% uncertainty interval
41                                                0.004265
83                                                0.021953
125                                               0.029802
167                                               0.223666
209                                               0.046309

[5 rows x 33 columns]
```

Some column names are pretty long and make it inconvenient to read and work with. The next code chunk shortens these column names to make it more readible. This is done by regex and substituting some words into shorter words or removing parts of the column name.

[4]:
```
# dict for patterns and replacements for the column names
replace_dict = {'Prevalence': 'Prev', r'kg/m.*': '', 'lower 95% uncertainty␣
↪interval': 'lower', 'upper 95% uncertainty interval': 'upper'}
for pattern, replacement in replace_dict.items():
    df_bmi = df_bmi.rename(columns=lambda column: re.sub(pattern, replacement,␣
↪column).rstrip())  # loops trhough all of the columns and use re.sub for the␣
↪replacement
```

[5]:
```
df_bmi.head()
```

[5]:
```
     Country/Region/World  ISO  Sex   Mean BMI  Mean BMI lower  Mean BMI upper  \
41            Afghanistan  AFG  Men  22.682456       20.157475       25.241857
83                Albania  ALB  Men  27.174471       25.975170       28.338256
125               Algeria  DZA  Men  24.865386       23.487321       26.220294
167        American Samoa  ASM  Men  33.066721       31.338678       34.662447
209               Andorra  AND  Men  27.478395       24.988831       30.001977

     Prev of BMI>=30  Prev of BMI>=30  Prev of BMI>=30  Prev of BMI>=35  …  \
41          0.033603         0.013884         0.066334         0.003314  …
83          0.223735         0.153334         0.300834         0.045036  …
125         0.206662         0.141854         0.279979         0.042840  …
167         0.587546         0.502606         0.666355         0.322678  …
209         0.267498         0.186223         0.354723         0.068565  …

     Prev of BMI 25  Prev of BMI 30  Prev of BMI 30  Prev of BMI 30  \
41         0.242503        0.030290        0.011207        0.062681
83         0.515957        0.178699        0.113401        0.252200
125        0.463570        0.163822        0.102610        0.234048
167        0.370043        0.264868        0.191211        0.340031
209        0.532478        0.198934        0.125035        0.280761

     Prev of BMI 35  Prev of BMI 35  Prev of BMI 35  Prev of BMI >=40  \
41         0.002271        0.000310        0.007487          0.001043
83         0.037684        0.013616        0.076984          0.007352
125        0.031750        0.011066        0.065164          0.011090
167        0.183871        0.109862        0.263723          0.138807
209        0.052701        0.017627        0.109639          0.015864

     Prev of BMI >=40  Prev of BMI >=40
41           0.000074          0.004265
83           0.001180          0.021953
125          0.002289          0.029802
167          0.067651          0.223666
209          0.002652          0.046309

[5 rows x 33 columns]
```

1.3.2 3.2 Affordability of a healthy diet

The affordability of a healthy diet dataset, contains some irrelevant columns which can be dropped. Furthermore, the NaN values are displayed as `..` and therefore must be replaced by real NaN values.

[6]:
```python
df_healthy_diet = pd.read_csv(healthy_diet_affordability, encoding='latin-1',␣
 ↪skipfooter=5, engine='python')  # skip last lines, does not contain data
df_healthy_diet.drop(columns=['Classification Name', 'Classification Code',␣
 ↪'Time', 'Time Code'], inplace=True)  # irrelevant columns
```

```
df_healthy_diet.rename(columns={'Affordability of a healthy diet: ratio of cost␣
↪to food expenditures [CoHD_fexp]': 'Affordability of a healthy diet'},␣
↪inplace=True)  # shortens column name
df_healthy_diet['Affordability of a healthy diet'] =␣
↪df_healthy_diet['Affordability of a healthy diet'].replace('..', np.nan)  #␣
↪replaces .. to nan values
df_healthy_diet = df_healthy_diet.astype({'Affordability of a healthy diet':␣
↪float})
```

[7]:
```
df_healthy_diet.head()
df_healthy_diet.dtypes
```

[7]:
```
Country Name                          object
Country Code                          object
Affordability of a healthy diet      float64
dtype: object
```

1.4 4. Data exploration

1.4.1 4.1 Data merging

Before the data exploration is done, a decision must be made whether or not the data should be merged already. This can be decided by looking at the differences in countries used in the datasets. The datasets are merged on the country variable. If some countries are not shared between the datasets, the data will be omitted.

[8]:
```
countries_bmi = df_bmi['ISO'].unique()
countries_healthy_diet = df_healthy_diet['Country Code'].unique()
countries_not_in_common = set(countries_bmi) ^ set(countries_healthy_diet)  #␣
↪check which values are not shared
print(f'countries not in common:\n {countries_not_in_common}\n')
print(f'amount: {len(countries_not_in_common)}')
```

```
countries not in common:
 {'SOM', 'FSM', 'WSM', 'ABW', 'CUB', 'UKR', 'SSF', 'CYM', 'TCA', 'HIC', 'PRI',
'GTM', 'MHL', 'GEO', 'WLD', 'VGB', 'SXM', 'AFG', 'LIC', 'KIR', 'LMC', 'UZB',
'LBY', 'TKM', 'GRL', 'TKL', 'ERI', 'PNG', 'VEN', 'BON', 'ASM', 'LBN', 'NRU',
'PLW', 'VUT', 'NIU', 'MSR', 'LCN', 'UMC', 'YEM', 'EAS', 'ECS', 'SYR', 'AND',
'TLS', 'TON', 'NAC', 'SAS', 'COK', 'SLB', 'TUV', 'MEA', 'CUW', 'PYF', 'PRK',
'AIA'}

amount: 56
```

As can be seen above there are a lot of countries (56) which are not in common between both datasets. When merging these datasets (using inner merge) this data gets lost. Therefore it makes sense to merge the datasets before data exploration, because otherwise data exploration is done on data which is later omitted.

[9]:
```
df_merged = df_healthy_diet.merge(right=df_bmi,
                                  left_on='Country Code',
                                  right_on='ISO',
                                  how='inner')
df_merged.drop(columns=['Country/Region/World', 'ISO'], inplace=True)  #␣
 ↪duplicate columns that can be deleted
df_merged.head()
```

[9]:
```
  Country Name Country Code  Affordability of a healthy diet    Sex  \
0      Albania          ALB                            0.425    Men
1      Albania          ALB                            0.425  Women
2      Algeria          DZA                            0.605    Men
3      Algeria          DZA                            0.605  Women
4       Angola          AGO                            0.972    Men

    Mean BMI  Mean BMI lower  Mean BMI upper  Prev of BMI>=30  \
0  27.174471       25.975170       28.338256         0.223735
1  26.507512       25.196840       27.859854         0.227215
2  24.865386       23.487321       26.220294         0.206662
3  26.561166       25.080506       28.031641         0.362187
4  22.436538       19.732903       25.172488         0.042276

   Prev of BMI>=30  Prev of BMI>=30  …  Prev of BMI 25  Prev of BMI 30  \
0         0.153334         0.300834  …        0.515957        0.178699
1         0.160322         0.300595  …        0.365555        0.154878
2         0.141854         0.279979  …        0.463570        0.163822
3         0.287831         0.440170  …        0.383587        0.225826
4         0.016349         0.081935  …        0.237444        0.035127

   Prev of BMI 30  Prev of BMI 30  Prev of BMI 35  Prev of BMI 35  \
0        0.113401        0.252200        0.037684        0.013616
1        0.096033        0.225306        0.054146        0.022398
2        0.102610        0.234048        0.031750        0.011066
3        0.155871        0.303311        0.093082        0.047469
4        0.011140        0.073699        0.005868        0.000739

   Prev of BMI 35  Prev of BMI >=40  Prev of BMI >=40  Prev of BMI >=40
0        0.076984          0.007352          0.001180          0.021953
1        0.100402          0.018191          0.004656          0.044692
2        0.065164          0.011090          0.002289          0.029802
3        0.152826          0.043279          0.015501          0.089878
4        0.018757          0.001281          0.000033          0.006267

[5 rows x 34 columns]
```

[10]:
```
df_merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Int64Index: 330 entries, 0 to 329
Data columns (total 34 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   Country Name                   330 non-null    object
 1   Country Code                   330 non-null    object
 2   Affordability of a healthy diet 322 non-null    float64
 3   Sex                            330 non-null    object
 4   Mean BMI                       330 non-null    float64
 5   Mean BMI lower                 330 non-null    float64
 6   Mean BMI upper                 330 non-null    float64
 7   Prev of BMI>=30                330 non-null    float64
 8   Prev of BMI>=30                330 non-null    float64
 9   Prev of BMI>=30                330 non-null    float64
 10  Prev of BMI>=35                330 non-null    float64
 11  Prev of BMI>=35                330 non-null    float64
 12  Prev of BMI>=35                330 non-null    float64
 13  Prev of BMI<18.5               330 non-null    float64
 14  Prev of BMI<18.5               330 non-null    float64
 15  Prev of BMI<18.5               330 non-null    float64
 16  Prev of BMI 18.5               330 non-null    float64
 17  Prev of BMI 18.5               330 non-null    float64
 18  Prev of BMI 18.5               330 non-null    float64
 19  Prev of BMI 20                 330 non-null    float64
 20  Prev of BMI 20                 330 non-null    float64
 21  Prev of BMI 20                 330 non-null    float64
 22  Prev of BMI 25                 330 non-null    float64
 23  Prev of BMI 25                 330 non-null    float64
 24  Prev of BMI 25                 330 non-null    float64
 25  Prev of BMI 30                 330 non-null    float64
 26  Prev of BMI 30                 330 non-null    float64
 27  Prev of BMI 30                 330 non-null    float64
 28  Prev of BMI 35                 330 non-null    float64
 29  Prev of BMI 35                 330 non-null    float64
 30  Prev of BMI 35                 330 non-null    float64
 31  Prev of BMI >=40               330 non-null    float64
 32  Prev of BMI >=40               330 non-null    float64
 33  Prev of BMI >=40               330 non-null    float64
dtypes: float64(31), object(3)
memory usage: 90.2+ KB
```

We have 330 entries left after merging. If correct, each country should have BMI data about both sexes. So data of a total of $330/2 = 165$ countries should be available. This is checked in the following code chunk. Additionally, some missing values can be observed in the 'Affordability of a healthy diet' column. This will be explored later.

[11]:
```
df_merged['Country Name'].nunique()
```

[11]: 165

This confirms that the dataset indeed holds data of 165 countries.

1.4.2 4.2 Affordability of a healthy diet

When looking at the 'Affordability of a healthy diet' data, we should select only one gender. This is because the data of 'Affordability of a healthy diet' is available per country. This data is the same per sex per country. To confirm this, the descriptive statistics can be compared.

[12]:
```
df_merged[df_merged['Sex'] == 'Men']['Affordability of a healthy diet'].
↪describe() == df_merged[df_merged['Sex'] == 'Women']['Affordability of a␣
↪healthy diet'].describe()
```

[12]:
```
count    True
mean     True
std      True
min      True
25%      True
50%      True
75%      True
max      True
Name: Affordability of a healthy diet, dtype: bool
```

This confirms that they are the same. Since the data of both genders are the same, we can select one of the genders when looking at this data

[13]:
```
df_merged[df_merged['Sex'] == 'Men']['Affordability of a healthy diet'].
↪describe()
```

[13]:
```
count    161.000000
mean       0.818025
std        0.679886
min        0.247000
25%        0.391000
50%        0.640000
75%        0.972000
max        5.272000
Name: Affordability of a healthy diet, dtype: float64
```

The first thing that is noticebly is that there are 161 entries, while we have 165 countries. This is probably due to missing values. Additionaly, a high standard deviation can be observed. The max value is also high compared the 75% (Q3) value. This means that here are probably outliers.

[14]:
```
df_merged[df_merged['Sex'] == 'Men']['Affordability of a healthy diet'].isna().
↪sum()
```

[14]: 4

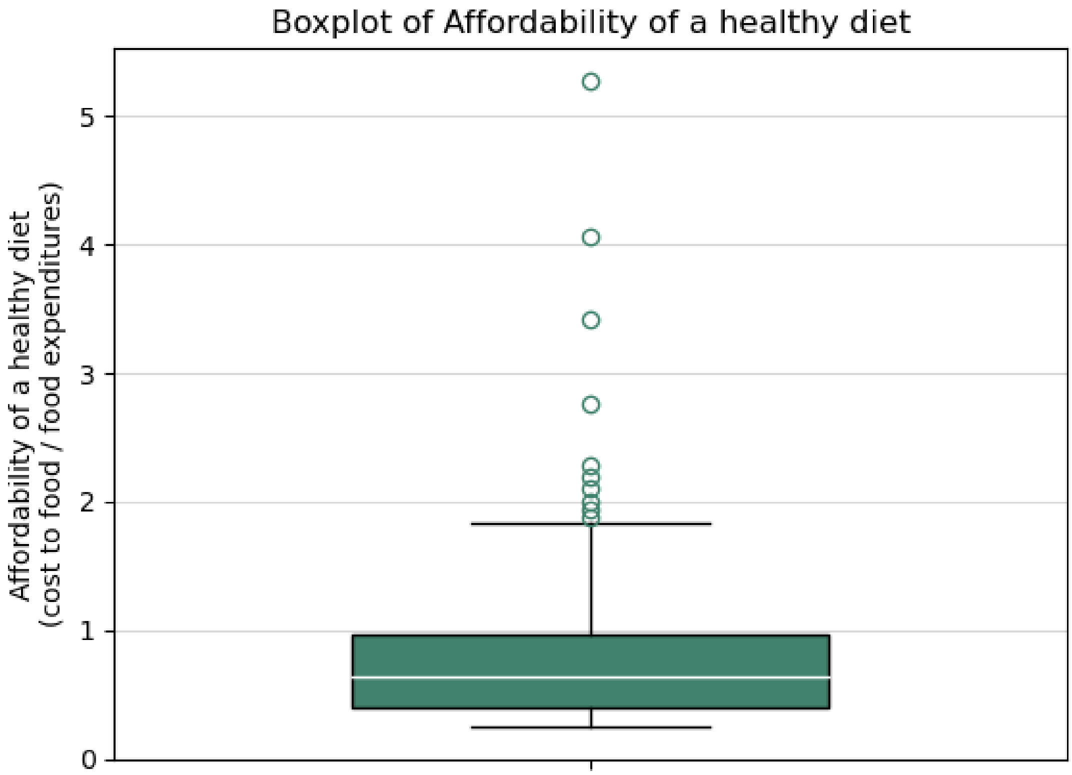
There are indeed missing data for four countries. Since this data cannot be used in our data analysis later on, we should remove these entries from the merged dataset.

[15]:
```
df_merged = df_merged[df_merged['Affordability of a healthy diet'].notna()]  #␣
↪keeping all values that or not na.
df_merged[df_merged['Sex'] == 'Men']['Affordability of a healthy diet'].isna().
↪sum()
```

[15]: 0

The rows with missing values in the affordability of a healthy diet have been succesfully removed. Since the descriptive statistics show a a large distribution it might be good to look at the distribution of the data visually using a boxplot.

[16]:
```
def create_afford_boxplot(df, xlabel, ylabel, title):
    fig = plt.figure()
    plt.boxplot(df, widths=0.5,
                flierprops=dict(markeredgecolor='#40826D', marker='o'),
                boxprops=dict(facecolor='#40826D'),
                medianprops= dict(color='white'),
                patch_artist=True)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.title(title)
    plt.xticks([1], [None])  # remove the redundant xtick number
    plt.grid(axis='y', alpha=.5)
    return fig   # so that the figure can be saved in a variable for the␣
↪dashboard later

boxplot_affordability = create_afford_boxplot(df_merged[df_merged['Sex'] ==␣
↪'Men']['Affordability of a healthy diet'],
                                              xlabel=None,␣
↪ylabel='Affordability of a healthy diet\n(cost to food / food expenditures)',
                                              title='Boxplot of Affordability␣
↪of a healthy diet')
```



As expected there are a lot of outliers in the upper range. This means that in certain countries the healthy diet is very unaffordable. To get an insight into the reason, we can look at these countries.

[17]:
```
stats = df_merged[df_merged['Sex'] == 'Men']['Affordability of a healthy diet'].
  ↪describe()
q1 = stats[4]
q3 = stats[6]
iqr = q3 - q1
upper_value = q3 + 1.5 * iqr

list(df_merged[df_merged['Affordability of a healthy diet'] >␣
  ↪upper_value]['Country Name'].unique())
```

[17]:
```
['Burkina Faso',
 'Burundi',
 'Central African Republic',
 'Congo, Dem. Rep.',
 'Ethiopia',
 'Ghana',
 'Liberia',
 'Malawi',
 'Mozambique',
```

```
 'Niger']
```

These are all poor countries in Africa. Since there is a lot of poverty in Africa, it makes sense that a healthy diet is also very unaffordable, just like all the other food. The same problem may also occur in other poor countries outside Africa. Because of this reason and the intention of this project the decision is made to only include countries located in Europe, where no real poor countries are located. To do this, a table is found in where it list all of the countries and there 3 letter codes and there corresponding continent. The table is obtained from: https://gist.githubusercontent.com/stevewithington/20a69c0b6d2ff846ea5d35e5fc47f26c/raw/13716ceb2f22b5643c and-continent-codes-list-csv.csv.

[18]:
```
df_countries = pd.read_csv(config['country_codes'])
df_countries.head()
```

[18]:
```
  Continent_Name Continent_Code                                Country_Name  \
0           Asia             AS           Afghanistan, Islamic Republic of
1         Europe             EU                        Albania, Republic of
2     Antarctica             AN  Antarctica (the territory South of 60 deg S)
3         Africa             AF      Algeria, People's Democratic Republic of
4        Oceania             OC                              American Samoa

  Two_Letter_Country_Code Three_Letter_Country_Code  Country_Number
0                      AF                       AFG             4.0
1                      AL                       ALB             8.0
2                      AQ                       ATA            10.0
3                      DZ                       DZA            12.0
4                      AS                       ASM            16.0
```

[19]:
```
contries_eu = df_countries[df_countries['Continent_Name'] ==␣
 ↪'Europe']['Three_Letter_Country_Code']
df_merged = df_merged[df_merged['Country Code'].isin(contries_eu)]
df_merged['Country Name'].nunique()
```

[19]:
```
43
```

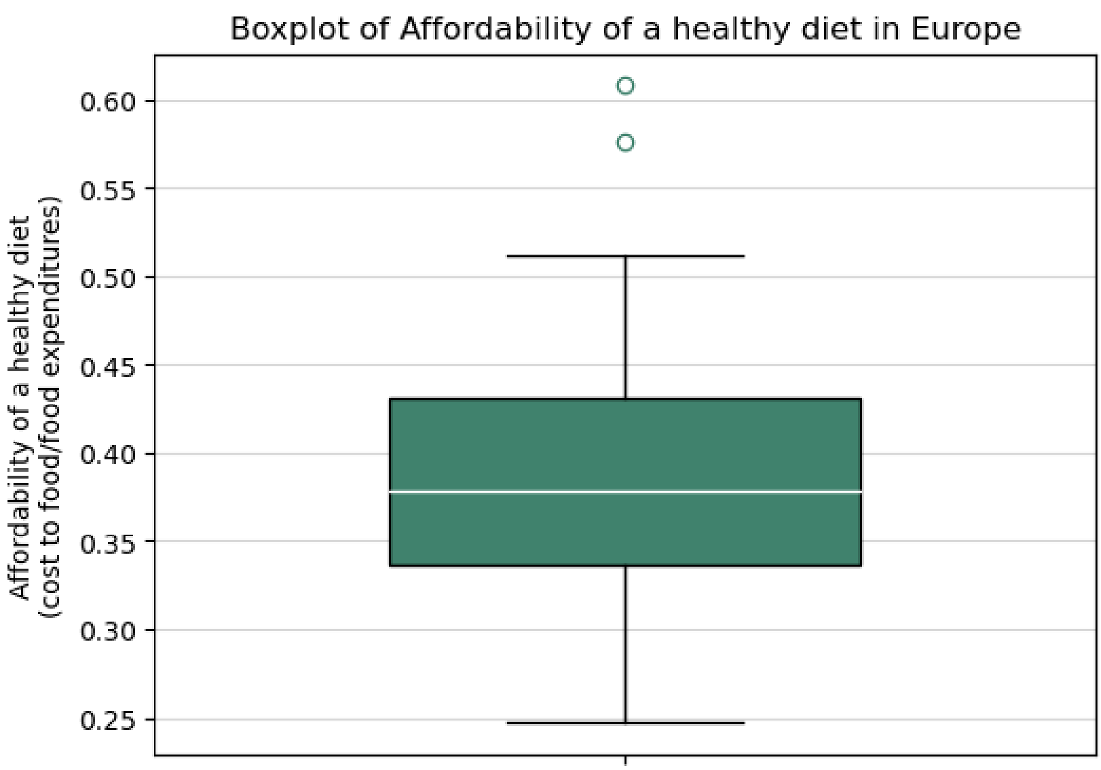
There are 43 countries left after selecting European countries. Because a lot of data is removed we can again have a look at the distribution of the data, descriptive and graphically.

[20]:
```
df_merged[df_merged['Sex'] == 'Men']['Affordability of a healthy diet'].
 ↪describe()
```

[20]:
```
count    43.000000
mean      0.384070
std       0.082415
min       0.247000
25%       0.337000
50%       0.378000
75%       0.431000
```

```
max       0.608000
Name: Affordability of a healthy diet, dtype: float64
```

[21]:
```
boxplot_affordability_eu = create_afford_boxplot(df_merged[df_merged['Sex'] ==␣
↪'Men']['Affordability of a healthy diet'],
                                              xlabel=None,␣
↪ylabel='Affordability of a healthy diet\n(cost to food/food expenditures)',
                                              title='Boxplot of␣
↪Affordability of a healthy diet in Europe')
```



[22]:
```
stats = df_merged[df_merged['Sex'] == 'Men']['Affordability of a healthy diet'].
↪describe()
q1 = stats[4]
q3 = stats[6]
iqr = q3 - q1
upper_value = q3 + 1.5 * iqr

list(df_merged[df_merged['Affordability of a healthy diet'] >␣
↪upper_value]['Country Name'].unique())
```

[22]:
```
['Bulgaria', 'Serbia']
```

The distribution of the data is now better. There are two outliers (Bulgaria and Serbia), because there is no reason to remove them they are not removed.

1.4.3 4.3 BMI

The BMI data can also be explored:

[23]:
```
df_merged['Mean BMI'].describe()
```

[23]:
```
count    86.000000
mean     26.614409
std       0.947460
min      23.818752
25%      26.062499
50%      26.669177
75%      27.201321
max      28.942234
Name: Mean BMI, dtype: float64
```

As can be seen the mean BMI in the world using the available data is approximately 26.6. At a BMI higher or equal to 25 a person is classified as 'overweight'. This means that the mean of the population is overweight. The standard deviation is not very large and quartiles look nicely distributed. We can also look at the distribution of the BMI data per Sex.

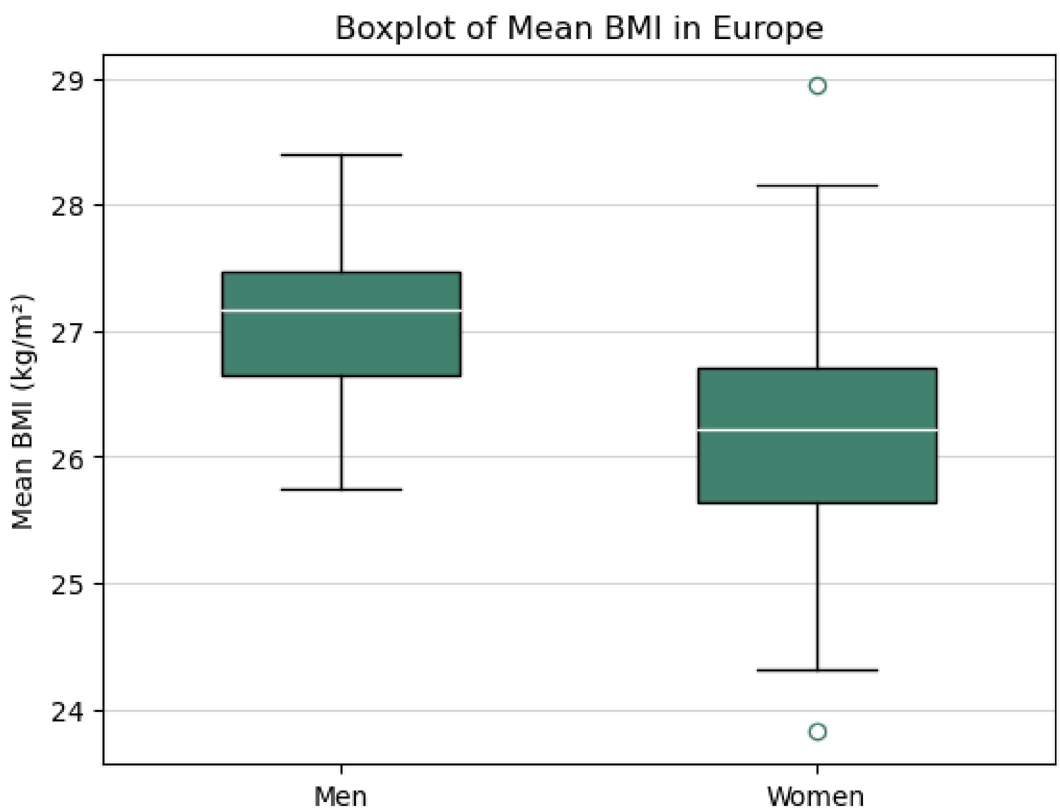
[24]:
```
df_merged.groupby('Sex')['Mean BMI'].describe()
```

[24]:
```
       count       mean       std        min        25%        50%        75%  \
Sex
Men     43.0  27.064342  0.635135  25.743889  26.639426  27.162678  27.476467
Women   43.0  26.164475  0.999408  23.818752  25.636193  26.226505  26.715332

             max
Sex
Men    28.395202
Women  28.942234
```

In general, the BMI data from men and women look similar. Most of the values are lower in women, when compared to men. However, the maximum value and standard deviation is a bit higher in the women dataset. This higher standard deviation is probably visible when ploting the data in a boxplot.

[25]:
```
def create_bmi_boxplot(df, column, groupby, xlabel, ylabel, title):
    fig = plt.figure()
    box_groups = df[groupby].unique()
    boxes = [df[df[groupby] == group][column] for group in box_groups]
    plt.boxplot(boxes,widths=0.5,
                # setting colors of boxplot
                flierprops=dict(markeredgecolor='#40826D', marker='o'),
                boxprops=dict(facecolor='#40826D'),
                medianprops= dict(color='white'),
```

```
                patch_artist=True)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.title(title)
    plt.xticks([i for i in range(1, len(box_groups)+1)], box_groups)  # change␣
 ↪x-ticks names
    plt.grid(axis='y', alpha=.5)
    return fig  # so that the figure can be saved in a variable for the␣
 ↪dashboard later

boxplot_bmi = create_bmi_boxplot(df_merged, column='Mean BMI', groupby='Sex',
                               xlabel=None, ylabel='Mean BMI (kg/m²)',
                               title='Boxplot of Mean BMI in Europe')
```



As expected, the mean BMI data shows a nice distribution for both genders.The data of men is a bit higher in all quartiles, and therefor also has a higher median. The women data has a higher spread in the dsitrbution compared to men, and also has an outlier on both sides. The followining code chunk looks at which countries are these outliers:

[26]:
```
print('Upper outlier:\n', df_merged['Country Name'][df_merged['Mean BMI'] ==␣
 ↪max(df_merged['Mean BMI'])])
```

```
print()
print('Lower outlier:\n', df_merged['Country Name'][df_merged['Mean BMI'] ==␣
↪min(df_merged['Mean BMI'])])
```

```
Upper outlier:
 311    Türkiye
Name: Country Name, dtype: object

Lower outlier:
 295    Switzerland
Name: Country Name, dtype: object
```

The outliers are the Mean BMI of Turkey and Switserland, since there again is no reason to remove these outliers so they are not removed from the dataset.

Next we can have a look at each mean BMI per country per sex. A barplot is chosen where also the 95% confidence interval is shown using whiskers. Since there are a lot of country, and each country has a mean BMI for each gender it is almost impossible to show this all at once. Therefore, panel widgets are used in which the user can select which data is shown. (multiple values can be selected by holding ctrl or shift and clicking mutlitple countries or genders or by clicking and dragging the mouse)

[27]:
```
def create_barplot(df, countries, genders):

    # select only the to be used data
    df = df[df['Country Name'].isin(countries)]
    df = df[df['Sex'].isin(genders)]

    source = ColumnDataSource(df)

    # create grouped data x-axis names
    country_gender = [(country, gender) for country in countries for gender in␣
↪genders]
    source= ColumnDataSource(data=dict(x=country_gender, bars=list(df['Mean␣
↪BMI']),
                                        lower=list(df['Mean BMI lower']),␣
↪upper=list(df['Mean BMI upper'])))

    # create colors (if-else statements for if only one gender is selected)
    if len(genders) == 2:
        colors = factor_cmap('x', palette=['#89cff0', 'pink'], factors=genders,␣
↪start=1, end=2)
    else:
        if genders == ['Men']:
            colors= '#89cff0'
        else:
            colors='pink'
```

```
    # create barplot
    p = figure(x_range=FactorRange(*country_gender), plot_width=800,␣
 ↪y_range=[0,32],
               y_axis_label='Mean BMI (kg/m²)', title='Barplot of the mean BMI␣
 ↪per country per gender')
    p.vbar(x='x', top='bars', source=source, fill_color=colors,␣
 ↪line_color=None, width=.9)

    # create whiskers and add to plot
    whiskers = Whisker(source=source, base="x", upper="upper", lower="lower",
                 line_color='black', level="overlay")
    p.add_layout(whiskers)

    # vertical xaxis labels
    p.xaxis.major_label_orientation = 'vertical'
    p.xaxis.group_label_orientation = 'vertical'
    return p

# create lists of all countries and genders
countries = df_merged['Country Name'].unique()
genders = df_merged['Sex'].unique()

# create multiselect widgets
country_options = MultiSelect(options=list(countries),
                              value=list(countries)[:10], # select only first␣
 ↪10
                              size=20)  # window size of widget is set to 20
gender_options = MultiSelect(options=list(genders),
                             value=list(genders))

# bind widgets to the barplot function
bmi_barplot = pn.bind(create_barplot, df=df_merged, countries=country_options,␣
 ↪genders=gender_options)

# show barplots
interactive_barplot_bmi = pn.Column(pn.Row(country_options, gender_options),␣
 ↪bmi_barplot)
interactive_barplot_bmi
```

[27]:
```
Column
    [0] Row
        [0] MultiSelect(options=['Albania', 'Armenia', …], size=20,
value=['Albania', 'Armenia', …])
        [1] MultiSelect(options=['Men', 'Women'], value=['Men', 'Women'])
    [1] ParamFunction(function)
```

As can be explored above, most of the countries have similar BMI's. When taking the error or 95%

confidince intervals into account not much variation can be seen between most of the countries.

1.4.4 4.4 Geomap

Lastly, for the data exploration we can try to visualise both datasets geographically and see if the relationship can be seen visually. This was done with the help of the tutorial from: https://towardsdatascience.com/folium-and-choropleth-map-from-zero-to-pro-6127f9e68564.

[28]:
```
def create_choropleth(df, column, colors):

    choropleth = folium.Choropleth(geo_data=config['europe_geojson'],
                                   name="choropleth",
                                   highlight=True,
                                   data=df,
                                   columns=["Country Code", column],
                                   key_on="properties.ISO3",
                                   fill_color=colors,
                                   nan_fill_opacity=0,
                                   fill_opacity=0.8,
                                   line_opacity=0.2,
                                   legend_name=column)

    legend = linear.OrRd_09.scale(
                    df[column].min(),
                    df[column].max()).to_step(10)

    return choropleth, legend

map = folium.Map(location=[60, 10], zoom_start=3.4, overlay=False, tiles=None)
bmi_fig = folium.FeatureGroup(name='Mean BMI', overlay=False).add_to(map)
affordability_fig = folium.FeatureGroup(name='Affordability of a healthy diet',␣
 ↪overlay=False).add_to(map)

choropleth_bmi, legend_bmi = create_choropleth(df_merged, 'Mean BMI', 'OrRd')
legend_bmi.caption = 'Mean BMI (kg/m²)'

choropleth_afford, legend_afford = create_choropleth(df_merged, 'Affordability␣
 ↪of a healthy diet', 'OrRd')
legend_afford.caption = 'Affordability of a healthy diet (cost to food / food␣
 ↪expenditures)'

choropleth_bmi.geojson.add_to(bmi_fig)
choropleth_afford.geojson.add_to(affordability_fig)

legend_bmi.add_to(map)
legend_afford.add_to(map)

folium.TileLayer('CartoDB positron', overlay=True, control=False).add_to(map)
```

```
folium.LayerControl(collapsed=False).add_to(map)

map
```

[28]: <folium.folium.Map at 0x7f2fc6378bb0>

When switching between the data with the menu in the top right, no visual relation is observed. The hypothesis is that the mean BMI increases if the healthy diet is less affordable (higher value in the affordability of a healthy diet). If this assumption is true it was expected that dark coloured areas would be dark coloured in both maps. This does not seem to be the case.

1.5 5. Statistical analysis

Lastly, the relationship between BMI and affordability of a healthy diet can be statistaclly analysed. This is done by linear regression and correlation.

1.5.1 5.1 Linear regression

[29]:
```
X = sm.add_constant(df_merged['Affordability of a healthy diet'])  # design␣
↪matrix
model = sm.OLS(df_merged['Mean BMI'], X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               Mean BMI   R-squared:                       0.021
Model:                            OLS   Adj. R-squared:                  0.009
Method:                 Least Squares   F-statistic:                     1.761
Date:                Thu, 09 Feb 2023   Prob (F-statistic):              0.188
Time:                        17:39:04   Log-Likelihood:                -115.99
No. Observations:                  86   AIC:                             236.0
Df Residuals:                      84   BIC:                             240.9
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
===================
                                      coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
-------------------
const                              25.9780      0.490     52.986      0.000
25.003      26.953
Affordability of a healthy diet     1.6570      1.249      1.327      0.188
-0.826       4.140
==============================================================================
Omnibus:                        3.252   Durbin-Watson:                   2.327
```

```
Prob(Omnibus):                  0.197   Jarque-Bera (JB):                2.603
Skew:                          -0.405   Prob(JB):                        0.272
Kurtosis:                       3.266   Cond. No.                         14.1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

As can be seen in the summary, the fit has a very low R-squared (and adjusted R-squared). This means that the amount of variance explained by this model is very small. Therefore the quality of the fit is not good. We can visualise the fitted model, where it is expected that there is not a good fit. The p-value of the intercept is 0.000 and thus a significant parameter to the model. However, the p-value of the slope is 0.188 and not significant to the model. If the slope is not significant to the model it can be removed. This results in a straight horizontal line, and therefore there will not be a significant relationship.

[30]:
```
intercept = results.params[0]
slope = results.params[1]

regr_plot = figure(title='Affordability of a healthy diet on BMI',
                   x_axis_label='Affordability of a healthy diet\n(cost to food␣
 ↪/ food expenditures)',
                   y_axis_label='Mean BMI (kg/m²)')
regr_plot.circle(df_merged['Affordability of a healthy diet'], df_merged['Mean␣
 ↪BMI'], legend_label='data', size=8)
regr_plot.line(df_merged['Affordability of a healthy diet'], intercept +␣
 ↪slope*df_merged['Affordability of a healthy diet'],
               line_color='red', legend_label='fitted line', width=2)
show(regr_plot)
```

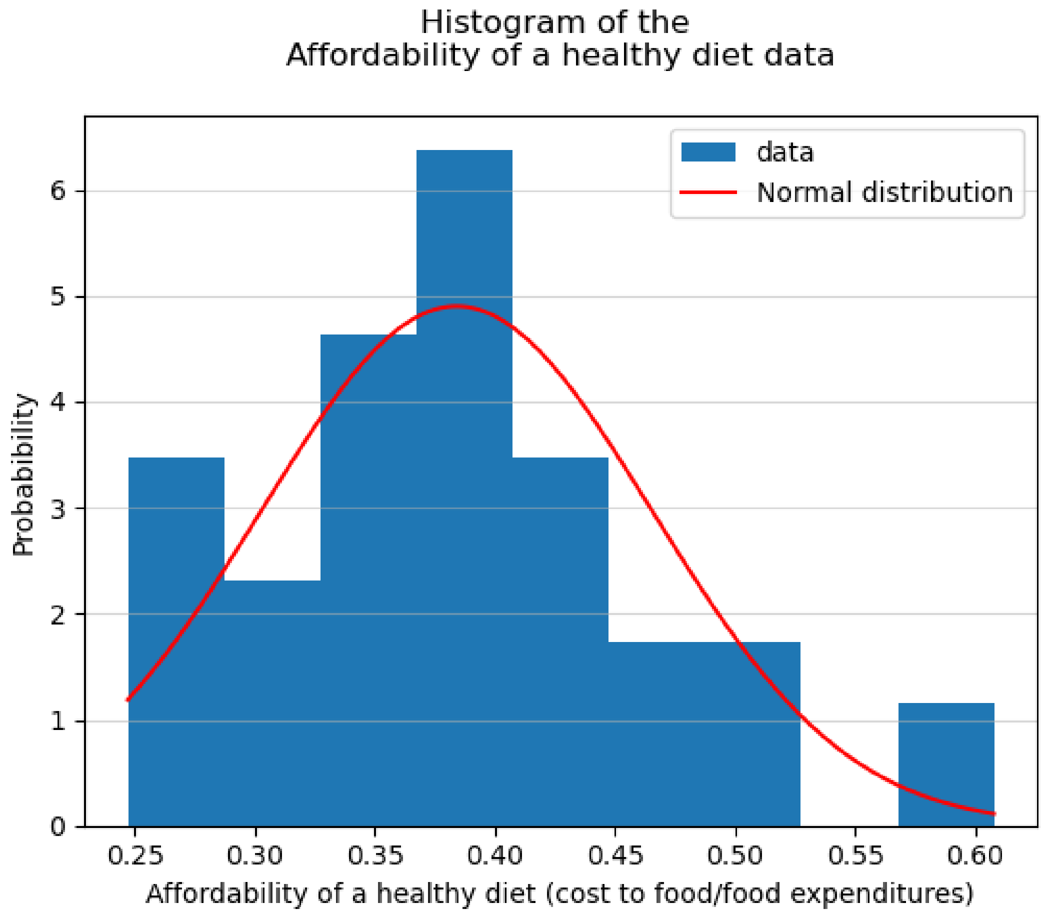
As expected, the fited line does not fit the data very well. The data it self has a lot of variance. This is also the reason the R-squared is so low, it is almost impossible to create a fit that explains this variance. In conclusion, the BMI does not seem to be affected by the affordability of a healthy diet.

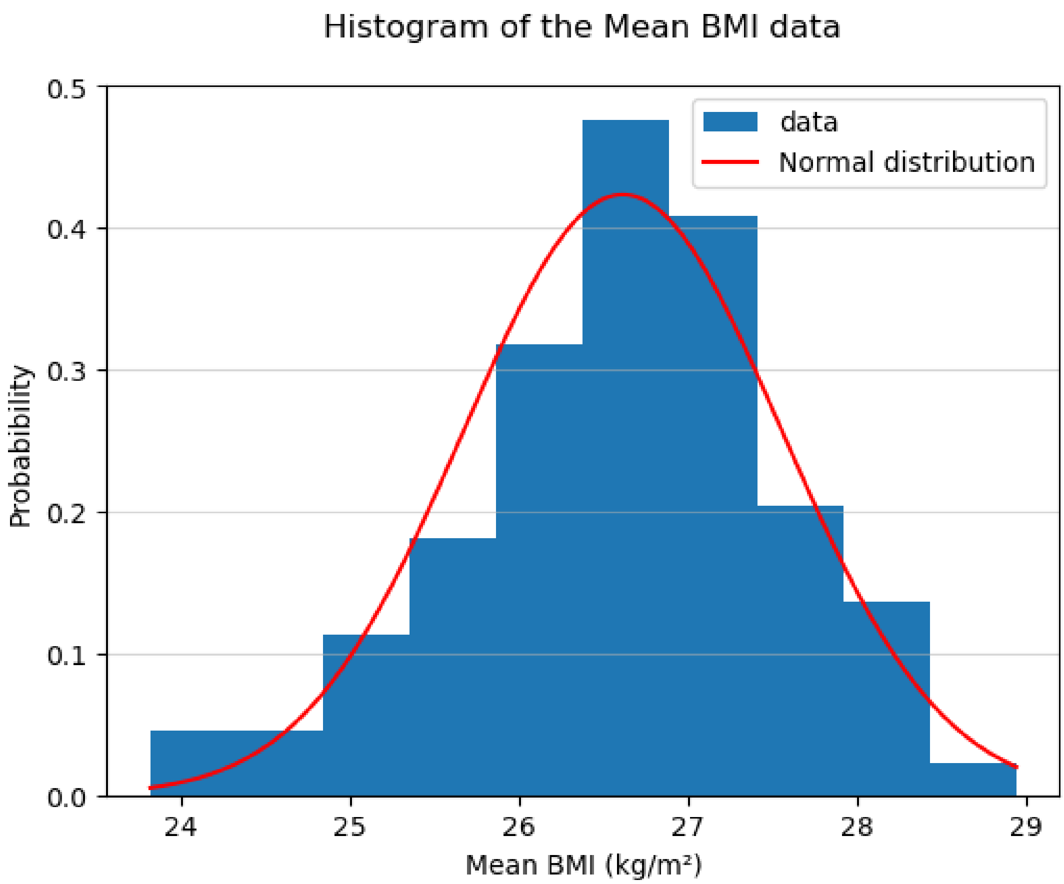
1.5.2 5.3 Correlation

To know which correlation tests to use, we need to know if the data is normally distrubted. First we look at both datasets using a histogram to see if we can visually see if it follows a normal distribution or not. After that we can use a Q-Q plot (Quantile-Quantile plot) to confirm or deny the normallity of the datasets.

[31]:
```
def create_histogram(data, title, xlabel):
    fig = plt.figure()

    # create data for normalised line approximation
    mean = np.mean(data)
    stdev = np.std(data)
```

```
    xs = np.linspace(min(data), max(data), 201)
    ys = np.array([norm.pdf(x, loc = mean, scale = stdev) for x in xs])

    # create the plot
    plt.hist(data, density=True, bins='auto', label='data')
    plt.plot(xs, ys, color='red', label='Normal distribution')
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel('Probabibility')
    plt.legend()
    plt.grid(axis='y', alpha=.5)
    return fig   # so that the figure can be saved in a variable for the␣
 ↪dashboard later

affordability_hist = create_histogram(df_merged['Affordability of a healthy␣
 ↪diet'], 'Histogram of the \nAffordability of a healthy diet data\n',
                 'Affordability of a healthy diet (cost to food/food␣
 ↪expenditures)')
bmi_hist = create_histogram(df_merged['Mean BMI'], 'Histogram of the Mean BMI␣
 ↪data\n',
                 'Mean BMI (kg/m²)')
```


Histogram of the
Affordability of a healthy diet data
data
Normal distribution
Probability
0
1
2
3
4
5
6
0.25
0.30
0.35
0.40
0.45
0.50
0.55
0.60
Affordability of a healthy diet (cost to food/food expenditures)



The first histogram does somewhat follow a normal distrubtion, but it contains a more data than expected in the tails and will probably be not normally distributed when checked with a Q-Q plot. However, the mean BMI data does seem to be normally distributed since it nicely follows the red line, which is the normal distribution approximation. We can check if they are normally distributed using a Q-Q plot.

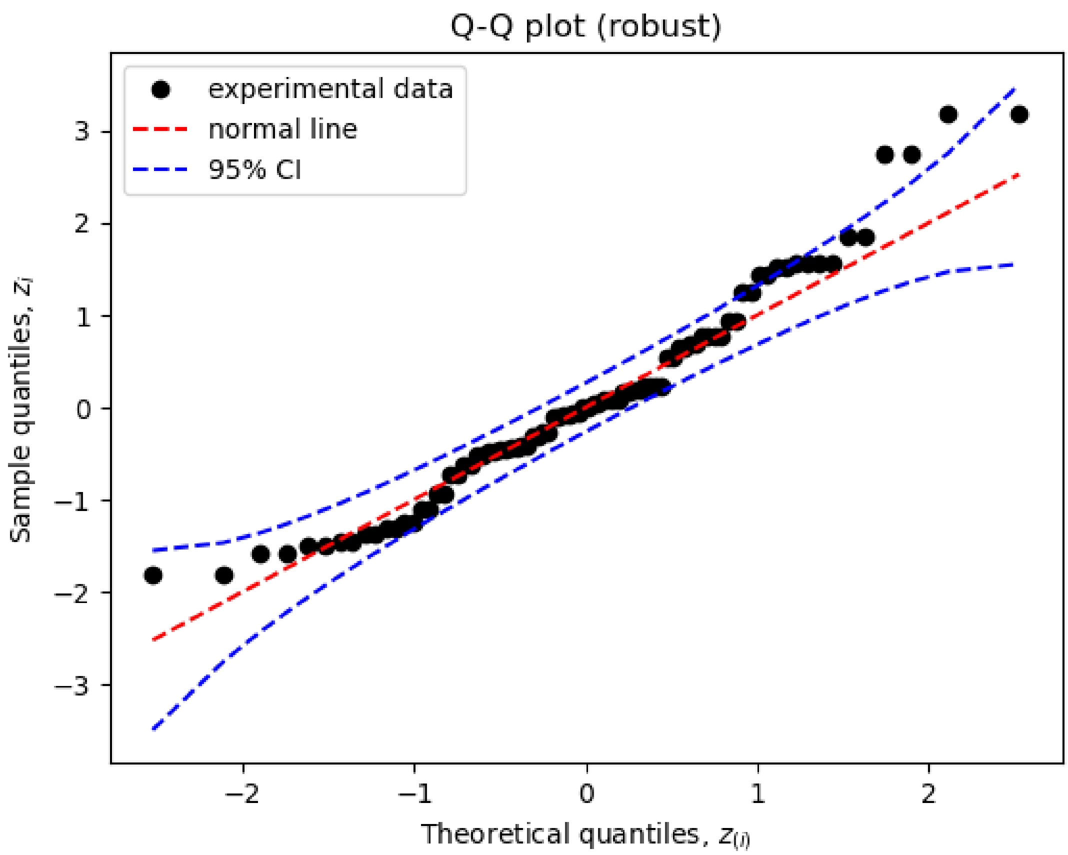
[32]:
```
def DS_Q_Q_Plot(y, est = 'robust', **kwargs):
    """
    *
    Function DS_Q_Q_Plot(y, est = 'robust', **kwargs)

       This function makes a normal quantile-quantile plot (Q-Q-plot), also␣
↪known
       as a probability plot, to visually check whether data follow a normal␣
↪distribution.

    Requires:            -

    Arguments:
```

```
    y                   data array
    est                 Estimation method for normal parameters mu and sigma:
                        either 'robust' (default), or 'ML' (Maximum␣
↪Likelihood),
                        or 'preset' (given values)
    N.B. If est='preset' than the *optional* parameters mu, sigma must be␣
↪provided:
    mu                  preset value of mu
    sigma               preset value of sigma

  Returns:
    Estimated mu, sigma, n, and expected number of datapoints outside CI in␣
↪Q-Q-plot.
    Q-Q-plot

  Author:             M.E.F. Apol
  Date:               2020-01-06, revision 2022-08-30
  """

  import numpy as np
  from scipy.stats import iqr # iqr is the Interquartile Range function
  import matplotlib.pyplot as plt

  # First, get the optional arguments mu and sigma:
  mu_0 = kwargs.get('mu', None)
  sigma_0 = kwargs.get('sigma', None)

  n = len(y)

  # Calculate order statistic:
  y_os = np.sort(y)

  # Estimates of mu and sigma:
  # ML estimates:
  mu_ML = np.mean(y)
  sigma2_ML = np.var(y)
  sigma_ML = np.std(y) # biased estimate
  s2 = np.var(y, ddof=1)
  s = np.std(y, ddof=1) # unbiased estimate
  # Robust estimates:
  mu_R = np.median(y)
  sigma_R = iqr(y)/1.349

  # Assign values of mu and sigma for z-transform:
  if est == 'ML':
      mu, sigma = mu_ML, s
  elif est == 'robust':
```

```python
        mu, sigma = mu_R, sigma_R
    elif est == 'preset':
        mu, sigma = mu_0, sigma_0
    else:
        print('Wrong estimation method chosen!')
        return()

    print('Estimation method: ' + est)
    print('n = {:d}, mu = {:.4g}, sigma = {:.4g}'.format(n, mu,sigma))

    # Expected number of deviations (95% confidence level):
    n_dev = np.round(0.05*n)

    print('Expected number of data outside CI: {:.0f}'.format(n_dev))

    # Perform z-transform: sample quantiles z.i
    z_i = (y_os - mu)/sigma

    # Calculate cumulative probabilities p.i:
    i = np.array(range(n)) + 1
    p_i = (i - 0.5)/n

    # Calculate theoretical quantiles z.(i):
    from scipy.stats import norm
    z_th = norm.ppf(p_i, 0, 1)

    # Calculate SE or theoretical quantiles:
    SE_z_th = (1/norm.pdf(z_th, 0, 1)) * np.sqrt((p_i * (1 - p_i)) / n)

    # Calculate 95% CI of diagonal line:
    CI_upper = z_th + 1.96 * SE_z_th
    CI_lower = z_th - 1.96 * SE_z_th

    # Make Q-Q plot:
    fig = plt.figure()
    plt.plot(z_th, z_i, 'o', color='k', label='experimental data')
    plt.plot(z_th, z_th, '--', color='r', label='normal line')
    plt.plot(z_th, CI_upper, '--', color='b', label='95% CI')
    plt.plot(z_th, CI_lower, '--', color='b')
    plt.xlabel('Theoretical quantiles, $z_{(i)}$')
    plt.ylabel('Sample quantiles, $z_i$')
    plt.title('Q-Q plot (' + est + ')')
    plt.legend(loc='best')
    return fig    # so that the figure can be saved in a variable for the␣
↪dashboard later
```

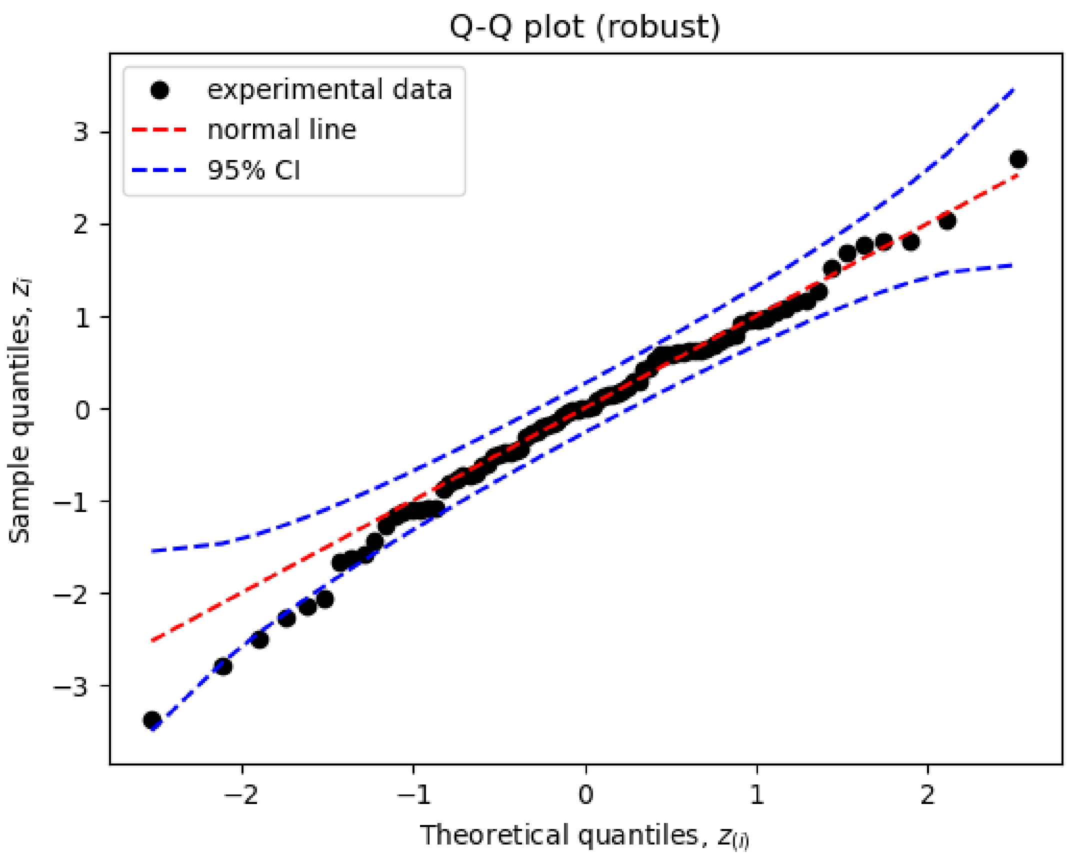
[33]:
```python
affordability_qqplot = DS_Q_Q_Plot(df_merged['Affordability of a healthy diet'])
```

```
Estimation method: robust
n = 86, mu = 0.378, sigma = 0.07228
Expected number of data outside CI: 4
```



[34]:
```
bmi_qqplot= DS_Q_Q_Plot(df_merged['Mean BMI'])
```

```
Estimation method: robust
n = 86, mu = 26.67, sigma = 0.8442
Expected number of data outside CI: 4
```



The frist Q-Q plot does show some points out of the 95% confidence interval. However, it is expected that four points are outside this confidence interval and there are about four points outside, but this is hard to see in the plot. It also does not really follow the diagonal line perfectly. In the second Q-Q plot the data does seem to follow the diagonal line nicely in the middle, but in the lower quantiles it has 4 points outside the confidence interval, but this is the same amount as expected.

Since it is still not very clear if the data is normally distributed or not, a normality test is used. The Shapiro-Wilk test is chosen, with the following hypothesis:

- H0: the data is normally distributed
- H1: the data is not normally distributed

It will be tested using an alpha of 0.05.

[35]:
```
stat, pval = shapiro(df_merged['Affordability of a healthy diet'])
print(f'p-value Affordability of a healthy diet: {pval:.6f}')
stat, pval = shapiro(df_merged['Mean BMI'])
print(f'p-value Mean BMI: {pval:.6f}')
```

```
p-value Affordability of a healthy diet: 0.009157
p-value Mean BMI: 0.495638
```

The p-value of the Affordability of a healthy diet is below the set alpha. Therefore H0 is rejected and H1 is accepted, the data is not normally distributed. On the other hand the p-value of the mean BMI is above the alpha, and is thus normally distributed.

Because one of the datasets is not normally distributed a the non-parametric correlation test is going to be used.

In order to see if there is a significant correlation between the two variables a Spearman rank-order correlation test is used. This tests calculated the correlation between two variables. If there is a strong positive correlation, the correlation will increase as well (up to 1). If there is a negative correlation the correlation will go down (up to -1). With this statistical test, you test if the correlation obtained is significantly different from 0. Because it is hypothesised that there might be a positive correlation the following hypothesis for this can be madecan be made:

- H0: The correlation is not different from 0 (correlation = 0)
- H1: The correlation is bigger than 0 (correlation > 0)

[36]:
```python
from scipy.stats import spearmanr

correlation, pval_correlation = spearmanr(df_merged['Affordability of a healthy␣
↪diet'], (df_merged['Mean BMI']), alternative='greater')
print(f'Correlation = {correlation:.4f}')
print(f'p-value     = {pval_correlation:.4f}')
```

```
Correlation = 0.1471
p-value     = 0.0883
```

The correlation obtained is approximately 0.1471, which is a small positive correlation. This means that when the affordability of a healthy diet increases the mean BMI also increases. However, the obtained p-value is approximatly 0.0883 and above an alpha of 0.05. Therfore, we can not reject H0 and this means that there is no significant correlation between the mean BMI and affordability of a healthy diet.

1.6 6. Conclusion

Using regression and correlation there does not seem to be a significant correlation between the mean BMI and affordability of a healthy diet in Europe.

1.7 7. Panel

[37]:
```python
# CSS styling
css = '''
.sidebar_button .bk-btn-group button {
    color: white;
    background-color: #40826D;
}

.bk-root {
     font-size: 108%;
}
'''
```

[38]:
```
# initialise dashboard
db = Dashboard(title='Affordability of a healthy diet and BMI',␣
↪header_color='#40826D',css=css)

# add pages to the dashboard
db.add_page('Homepage', True, txt.homepage)

db.add_page('Data exploration', False, txt.exploration1)
db.add_tabs_to_page('Data exploration', {'Affordability of a healthy diet':␣
↪[txt.exploration2 ,boxplot_affordability,
                                                                                    ␣
↪txt.exploration3, boxplot_affordability_eu,
                                                                                    ␣
↪txt.exploration4]},
                                        {'BMI': [txt.exploration5, boxplot_bmi,
                                                 txt.exploration6,␣
↪interactive_barplot_bmi,
                                                 txt.exploration7]})

db.add_page('Geo map', False, txt.geomap, pn.panel(map, height=600, width=1000))

db.add_page('Statistics', False, txt.statistics)
db.add_tabs_to_page('Statistics', {'Regression':  [results.summary(), txt.
↪regression1,
                                                   regr_plot, txt.regression2]},
                                  {'Correlation': [txt.correlation1,
                                                   pn.Row(affordability_hist,␣
↪bmi_hist),
                                                   txt.correlation2,
                                                   pn.Row(affordability_qqplot,␣
↪bmi_qqplot),
                                                   txt.correlation3]})

# show the dashboard
db.show()
```

Launching server at http://localhost:35459