Untitled

December 10, 2025

[1]:
```
#Importing the necessary libraries
```

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

[3]:
```
df1 = pd.read_csv('dailyActivity_merged.csv')
print(df1)
```

```
             Id ActivityDate  TotalSteps  TotalDistance  TrackerDistance  \
0    1503960366    4/12/2016       13162       8.500000         8.500000
1    1503960366    4/13/2016       10735       6.970000         6.970000
2    1503960366    4/14/2016       10460       6.740000         6.740000
3    1503960366    4/15/2016        9762       6.280000         6.280000
4    1503960366    4/16/2016       12669       8.160000         8.160000
..          …            …          …             …               …
935  8877689391     5/8/2016       10686       8.110000         8.110000
936  8877689391     5/9/2016       20226      18.250000        18.250000
937  8877689391    5/10/2016       10733       8.150000         8.150000
938  8877689391    5/11/2016       21420      19.559999        19.559999
939  8877689391    5/12/2016        8064       6.120000         6.120000

     LoggedActivitiesDistance  VeryActiveDistance  ModeratelyActiveDistance  \
0                         0.0                1.88                      0.55
1                         0.0                1.57                      0.69
2                         0.0                2.44                      0.40
3                         0.0                2.14                      1.26
4                         0.0                2.71                      0.41
..                        …                 …                        …
935                       0.0                1.08                      0.20
936                       0.0               11.10                      0.80
937                       0.0                1.35                      0.46
938                       0.0               13.22                      0.41
939                       0.0                1.82                      0.04
```

```
     LightActiveDistance  SedentaryActiveDistance  VeryActiveMinutes  \
0                   6.06                     0.00                 25
1                   4.71                     0.00                 21
2                   3.91                     0.00                 30
3                   2.83                     0.00                 29
4                   5.04                     0.00                 36
..                    …                      …                …
935                 6.80                     0.00                 17
936                 6.24                     0.05                 73
937                 6.28                     0.00                 18
938                 5.89                     0.00                 88
939                 4.25                     0.00                 23

     FairlyActiveMinutes  LightlyActiveMinutes  SedentaryMinutes  Calories
0                     13                   328               728      1985
1                     19                   217               776      1797
2                     11                   181              1218      1776
3                     34                   209               726      1745
4                     10                   221               773      1863
..                    …                    …               …       …
935                    4                   245              1174      2847
936                   19                   217              1131      3710
937                   11                   224              1187      2832
938                   12                   213              1127      3832
939                    1                   137               770      1849

[940 rows x 15 columns]
```

[4]:

```
df1["Date"] = pd.to_datetime(df1["ActivityDate"])
df1 = df1.drop(columns=["ActivityDate"])
```

[5]:

```
df1 = df1[["Id","Date", "TotalSteps","Calories", "TotalDistance",␣
 ↪"VeryActiveMinutes","FairlyActiveMinutes",␣
 ↪"LightlyActiveMinutes","SedentaryMinutes"]]

df1 = df1.drop_duplicates()
df1.duplicated().sum()
df1.isnull().sum()
print(df1.head())
print(df1.shape)
print(df1.columns)
#dropping the dublicates from the data (the same way it was taught at the␣
 ↪course assignments) and checking that there are only lines with information␣
 ↪included.
#Selecting the columns that I want to use in the project
```

```
           Id      Date  TotalSteps  Calories  TotalDistance  \
0  1503960366 2016-04-12       13162      1985           8.50
1  1503960366 2016-04-13       10735      1797           6.97
2  1503960366 2016-04-14       10460      1776           6.74
3  1503960366 2016-04-15        9762      1745           6.28
4  1503960366 2016-04-16       12669      1863           8.16

   VeryActiveMinutes  FairlyActiveMinutes  LightlyActiveMinutes  \
0                 25                   13                   328
1                 21                   19                   217
2                 30                   11                   181
3                 29                   34                   209
4                 36                   10                   221

   SedentaryMinutes
0               728
1               776
2              1218
3               726
4               773
(940, 9)
Index(['Id', 'Date', 'TotalSteps', 'Calories', 'TotalDistance',
       'VeryActiveMinutes', 'FairlyActiveMinutes', 'LightlyActiveMinutes',
       'SedentaryMinutes'],
      dtype='object')
```

[6]:

```
#this checks that we are using the data from the 31 days between 12.4-12.5.2016
```

[7]:

```
df1 = df1[
    (df1["Date"] >= "2016-04-12") &
    (df1["Date"] <= "2016-05-12")
]
print(df1.shape)
print(df1["Date"].nunique())
print(df1["Date"].min(), df1["Date"].max())
```

```
(940, 9)
31
2016-04-12 00:00:00 2016-05-12 00:00:00
```

[8]:

```
print(df1) #here we have the data that is cleaned and ready for visualizations
```

```
             Id       Date  TotalSteps  Calories  TotalDistance  \
0    1503960366 2016-04-12       13162      1985       8.500000
1    1503960366 2016-04-13       10735      1797       6.970000
2    1503960366 2016-04-14       10460      1776       6.740000
3    1503960366 2016-04-15        9762      1745       6.280000
4    1503960366 2016-04-16       12669      1863       8.160000
..          …        …          …       …            …
```

```
935  8877689391 2016-05-08       10686       2847        8.110000
936  8877689391 2016-05-09       20226       3710       18.250000
937  8877689391 2016-05-10       10733       2832        8.150000
938  8877689391 2016-05-11       21420       3832       19.559999
939  8877689391 2016-05-12        8064       1849        6.120000

     VeryActiveMinutes  FairlyActiveMinutes  LightlyActiveMinutes  \
0                   25                   13                   328
1                   21                   19                   217
2                   30                   11                   181
3                   29                   34                   209
4                   36                   10                   221
..                 …                  …                   …
935                 17                    4                   245
936                 73                   19                   217
937                 18                   11                   224
938                 88                   12                   213
939                 23                    1                   137

     SedentaryMinutes
0                 728
1                 776
2                1218
3                 726
4                 773
..                …
935              1174
936              1131
937              1187
938              1127
939               770

[940 rows x 9 columns]
```

[9]:

```
df5 = pd.read_csv('sleepDay_merged.csv')
df5["Date"] = pd.to_datetime(df5["SleepDay"])
print(df5)
#this reads the sleepdata
```

```
             Id               SleepDay  TotalSleepRecords  TotalMinutesAsleep  \
0    1503960366  4/12/2016 12:00:00 AM                  1                 327
1    1503960366  4/13/2016 12:00:00 AM                  2                 384
2    1503960366  4/15/2016 12:00:00 AM                  1                 412
3    1503960366  4/16/2016 12:00:00 AM                  2                 340
4    1503960366  4/17/2016 12:00:00 AM                  1                 700
..          …                    …                …                 …
408  8792009665  4/30/2016 12:00:00 AM                  1                 343
409  8792009665   5/1/2016 12:00:00 AM                  1                 503
```

```
410  8792009665   5/2/2016 12:00:00 AM                   1                 415
411  8792009665   5/3/2016 12:00:00 AM                   1                 516
412  8792009665   5/4/2016 12:00:00 AM                   1                 439

     TotalTimeInBed       Date
0               346 2016-04-12
1               407 2016-04-13
2               442 2016-04-15
3               367 2016-04-16
4               712 2016-04-17
..              …        …
408             360 2016-04-30
409             527 2016-05-01
410             423 2016-05-02
411             545 2016-05-03
412             463 2016-05-04

[413 rows x 6 columns]
```

```
/tmp/ipykernel_654/3438589101.py:2: UserWarning: Could not infer format, so each
element will be parsed individually, falling back to `dateutil`. To ensure
parsing is consistent and as-expected, please specify a format.
  df5["Date"] = pd.to_datetime(df5["SleepDay"])
```

[10]:

```python
df5 = df5[
    (df5["Date"] >= "2016-04-12") &(df5["Date"] <= "2016-05-12")]
print(df5.shape)
print(df5["Date"].min(), df5["Date"].max())
#this checks that the sleepdata is also from 12.4-12.5. period
```

```
(413, 6)
2016-04-12 00:00:00 2016-05-12 00:00:00
```

[11]:

```python
df5 = df5[[ "Id","Date","TotalMinutesAsleep","TotalTimeInBed"]]
df5.duplicated().sum()
df5 = df5.drop_duplicates()
#dropping dublicates from the sleepdata
```

[12]:

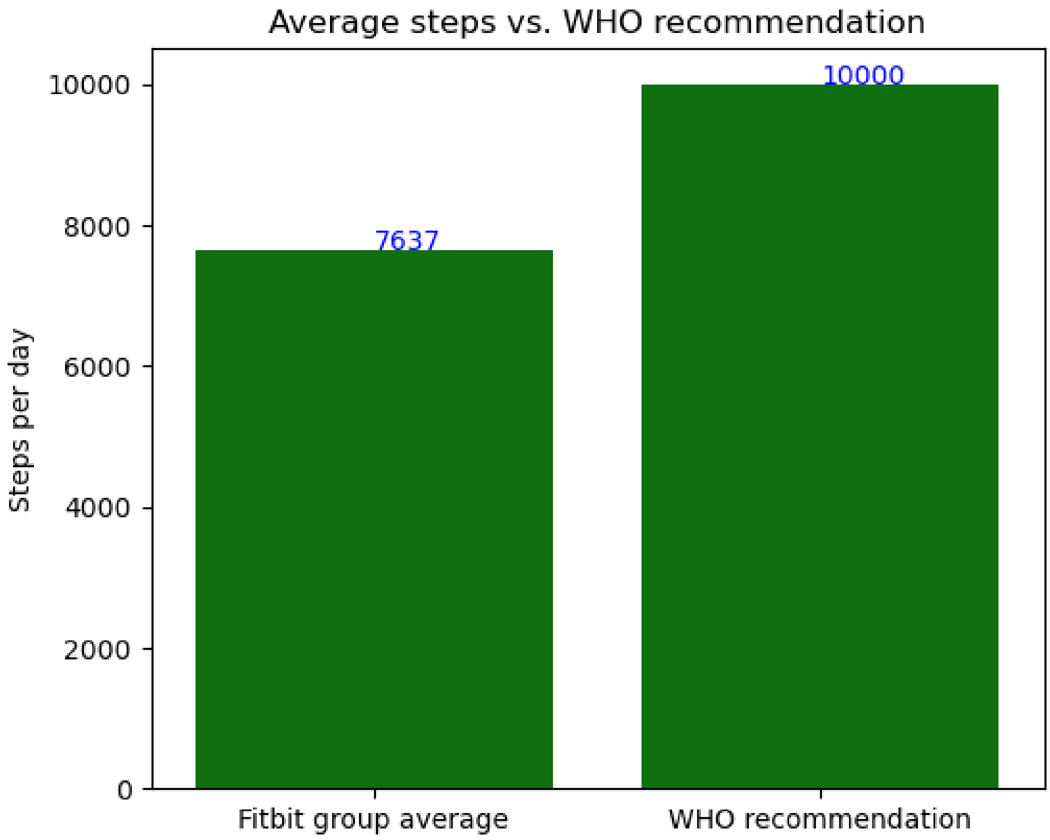
```python
df = pd.merge(df1, df5, on=["Id", "Date"],how="left")
print(df) # print the compined data
```

```
             Id       Date  TotalSteps  Calories  TotalDistance  \
0    1503960366 2016-04-12       13162      1985       8.500000
1    1503960366 2016-04-13       10735      1797       6.970000
2    1503960366 2016-04-14       10460      1776       6.740000
3    1503960366 2016-04-15        9762      1745       6.280000
4    1503960366 2016-04-16       12669      1863       8.160000
..          …        …         …       …             …
```

```
935  8877689391 2016-05-08       10686      2847       8.110000
936  8877689391 2016-05-09       20226      3710      18.250000
937  8877689391 2016-05-10       10733      2832       8.150000
938  8877689391 2016-05-11       21420      3832      19.559999
939  8877689391 2016-05-12        8064      1849       6.120000

     VeryActiveMinutes  FairlyActiveMinutes  LightlyActiveMinutes  \
0                   25                   13                   328
1                   21                   19                   217
2                   30                   11                   181
3                   29                   34                   209
4                   36                   10                   221
..                 …                  …                   …
935                 17                    4                   245
936                 73                   19                   217
937                 18                   11                   224
938                 88                   12                   213
939                 23                    1                   137

     SedentaryMinutes  TotalMinutesAsleep  TotalTimeInBed
0                 728               327.0           346.0
1                 776               384.0           407.0
2                1218                 NaN             NaN
3                 726               412.0           442.0
4                 773               340.0           367.0
..                …                 …             …
935              1174                 NaN             NaN
936              1131                 NaN             NaN
937              1187                 NaN             NaN
938              1127                 NaN             NaN
939               770                 NaN             NaN

[940 rows x 11 columns]
```

[13]:

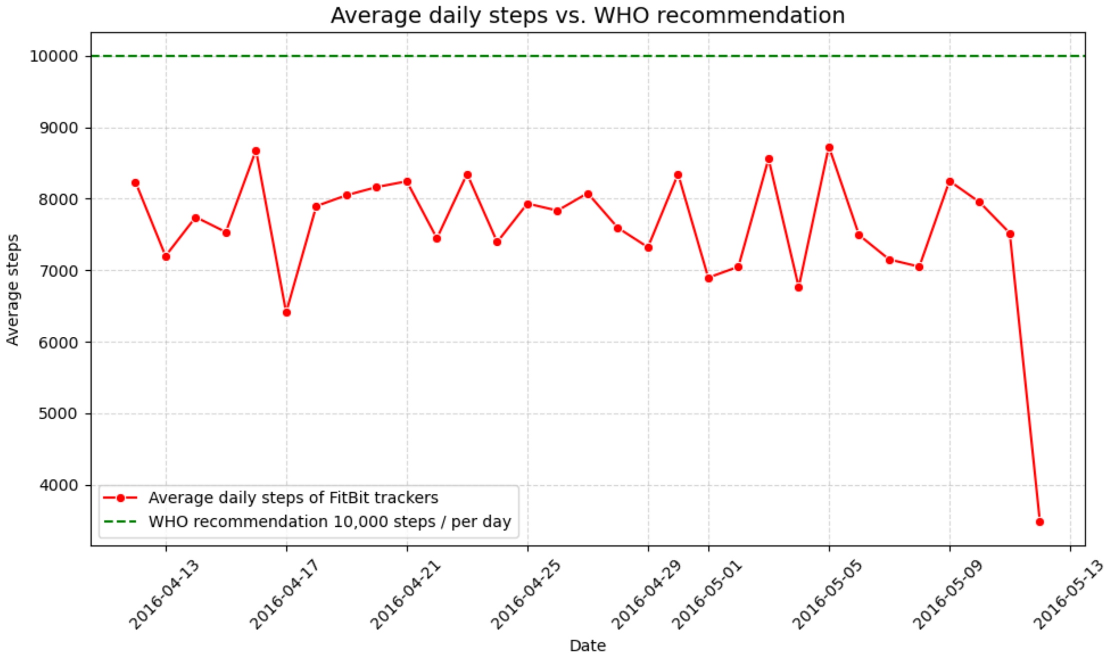
```
# first visualization: boxplot of the saverage steps compared to WHO␣
 ↪recommendations
average_steps = df1['TotalSteps'].mean()
comparison_data = {
    "Category": ["Fitbit group average", "WHO recommendation"],
    "Steps": [average_steps, 10000]
}
comparison_df = pd.DataFrame(comparison_data)
plt.figure(figsize=(6,5))
ax = sns.barplot(data=comparison_df, x="Category", y="Steps", color = "green")

for p in ax.patches: #adds the numbers of steps on top of the bow plots
    ax.annotate(
        format(int(p.get_height())),
        (p.get_x() + p.get_width() / 2., p.get_height()),color= "blue")

plt.title("Average steps vs. WHO recommendation")
plt.ylabel("Steps per day")
plt.xlabel("")
plt.show()
```



[14]:

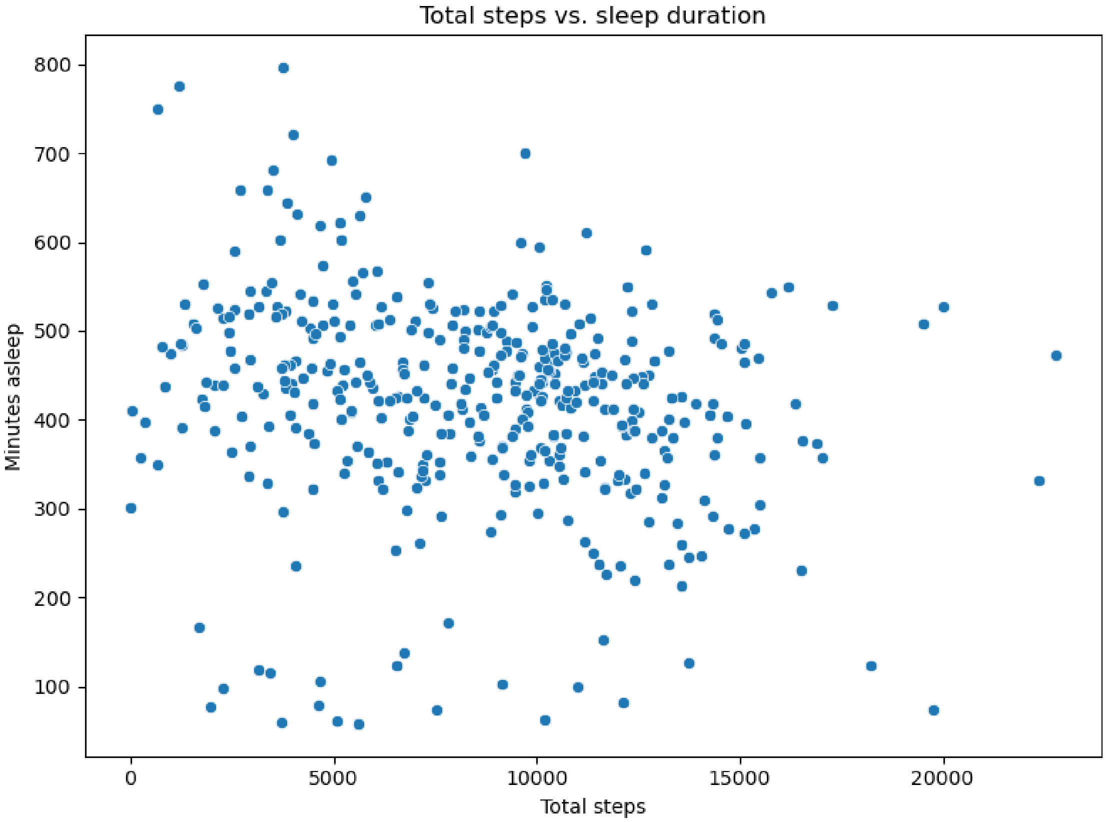
```
avg_daily_steps = df1.groupby("Date")["TotalSteps"].mean().reset_index()
plt.figure(figsize=(10,6))

# creating a lineplot of average daily steps compared to the 10 000 steps
sns.lineplot(
    data=avg_daily_steps,
    x="Date",
    y="TotalSteps",
    marker="o",
    color = "red",
    label = "Average daily steps of FitBit trackers",
)
```

```
#this draws the green line for the WHO recommendation
plt.axhline(y=10000, linestyle="--", color="green", label="WHO recommendation␣
  ↪10,000 steps / per day")

plt.title("Average daily steps vs. WHO recommendation", fontsize=14)
plt.xlabel("Date")
plt.ylabel("Average steps")
plt.xticks(rotation=45) #this makse the text roate on the x-axel to be easier␣
  ↪to read
plt.legend()
plt.grid(True, linestyle="--", alpha=0.5)

plt.tight_layout()
plt.show()
```



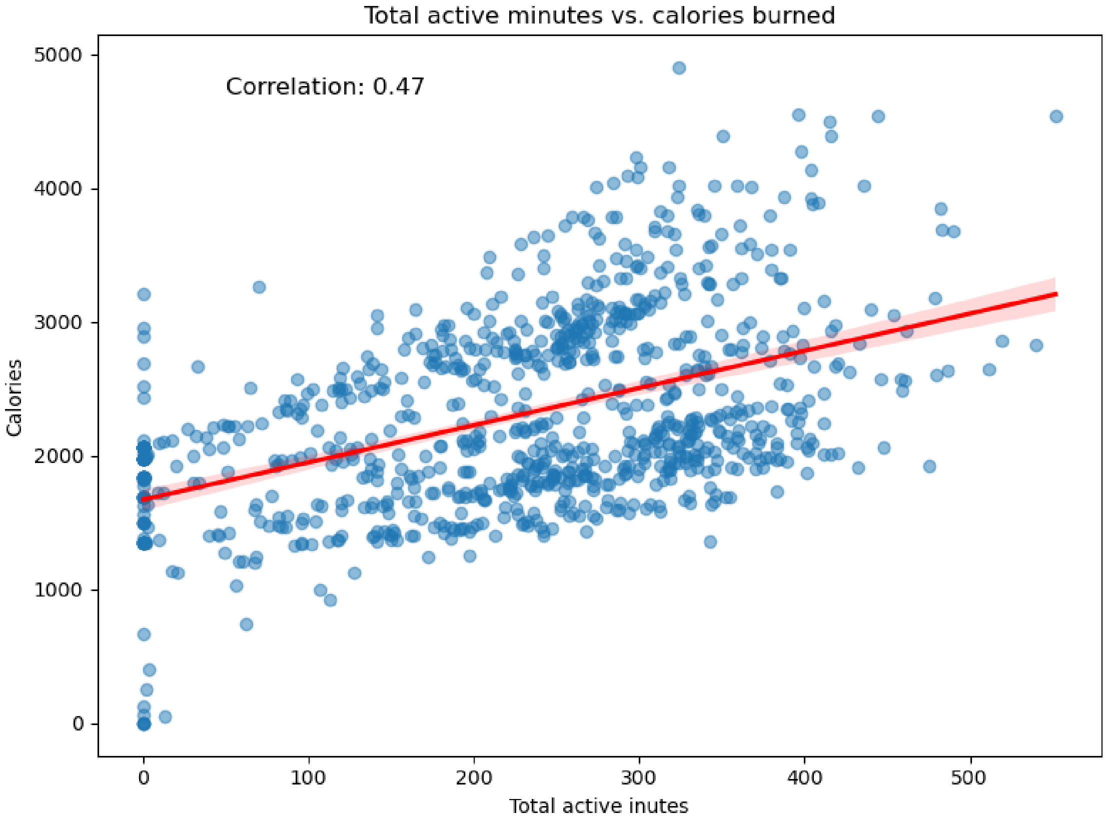
[15]:
```
df_sleep = df.dropna(subset=["TotalMinutesAsleep"])
```

[16]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(data=df_sleep,
                x="TotalSteps",
                y="TotalMinutesAsleep")

plt.title("Total steps vs. sleep duration")
plt.xlabel("Total steps")
plt.ylabel("Minutes asleep")
plt.tight_layout()
plt.show()
#extra visualisation not used in the report
```



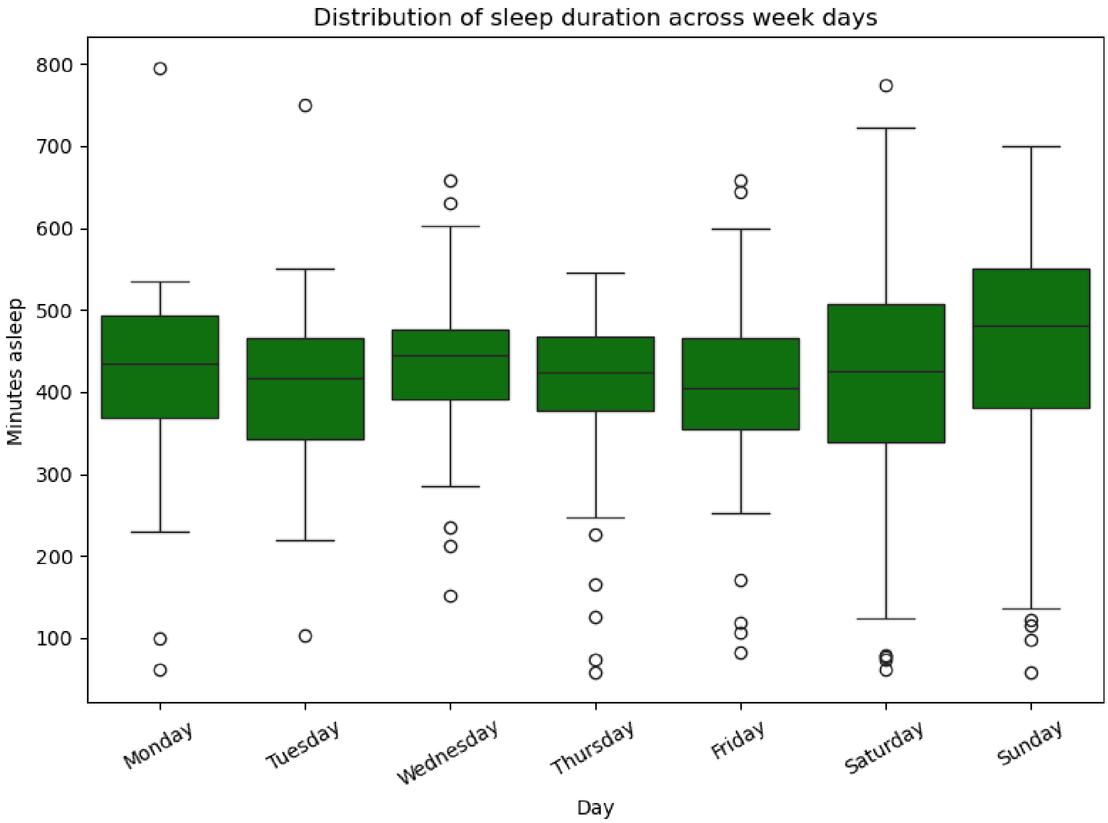
[17]:
```
df['TotalActiveMinutes'] = (df['VeryActiveMinutes'] + df['FairlyActiveMinutes']␣
↪+ df['LightlyActiveMinutes'])
corr = df["TotalActiveMinutes"].corr(df["Calories"])
#combining all the active minutes to create one variable TotalActiveMinutes and␣
↪then calcutate the correlation between the two columns

plt.figure(figsize=(8,6))
sns.regplot(
    data=df,
    x="TotalActiveMinutes",
    y="Calories",
    scatter_kws={"alpha":0.5}, #this makes the blue dots to be a little␣
↪seethrought inorder to see dots better when they are overlapping
    line_kws={"color": "red"}
)
```

```
plt.text(
    x=50,
    y=df["Calories"].max() - 200,
    s=f"Correlation: {corr:.2f}", #corr:.2f shows correlation with 2 decimals
    fontsize=12,
    color="black"
)

plt.title("Total active minutes vs. calories burned")
plt.xlabel("Total active inutes")
plt.ylabel("Calories")
plt.tight_layout()
plt.show()
```

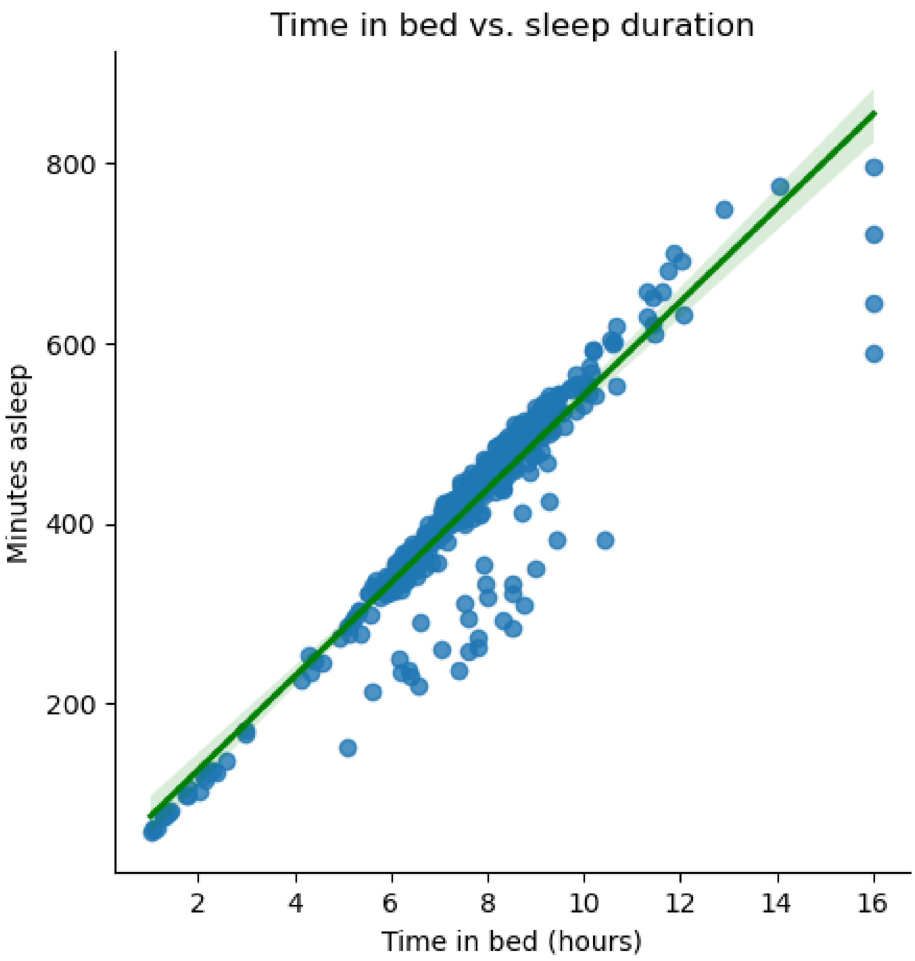


[18]:
```
df_sleep["Weekday"] = df_sleep["Date"].dt.day_name()

plt.figure(figsize=(8,6))
#creating a distribution boxplot of sleep duration across week days, boxplot␣
 ↪was introduced at the course material
sns.boxplot(data=df_sleep,
            x="Weekday",
            y="TotalMinutesAsleep",
           ␣
 ↪order=["Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"],␣
 ↪color = "green")

plt.title("Distribution of sleep duration across week days")
plt.xlabel("Day")
plt.ylabel("Minutes asleep")
plt.xticks(rotation=30)
plt.tight_layout()
plt.show()
```

```
/tmp/ipykernel_654/1340094653.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_sleep["Weekday"] = df_sleep["Date"].dt.day_name()
/opt/software/lib/python3.10/site-packages/seaborn/categorical.py:632:
FutureWarning: SeriesGroupBy.grouper is deprecated and will be removed in a
future version of pandas.
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
```



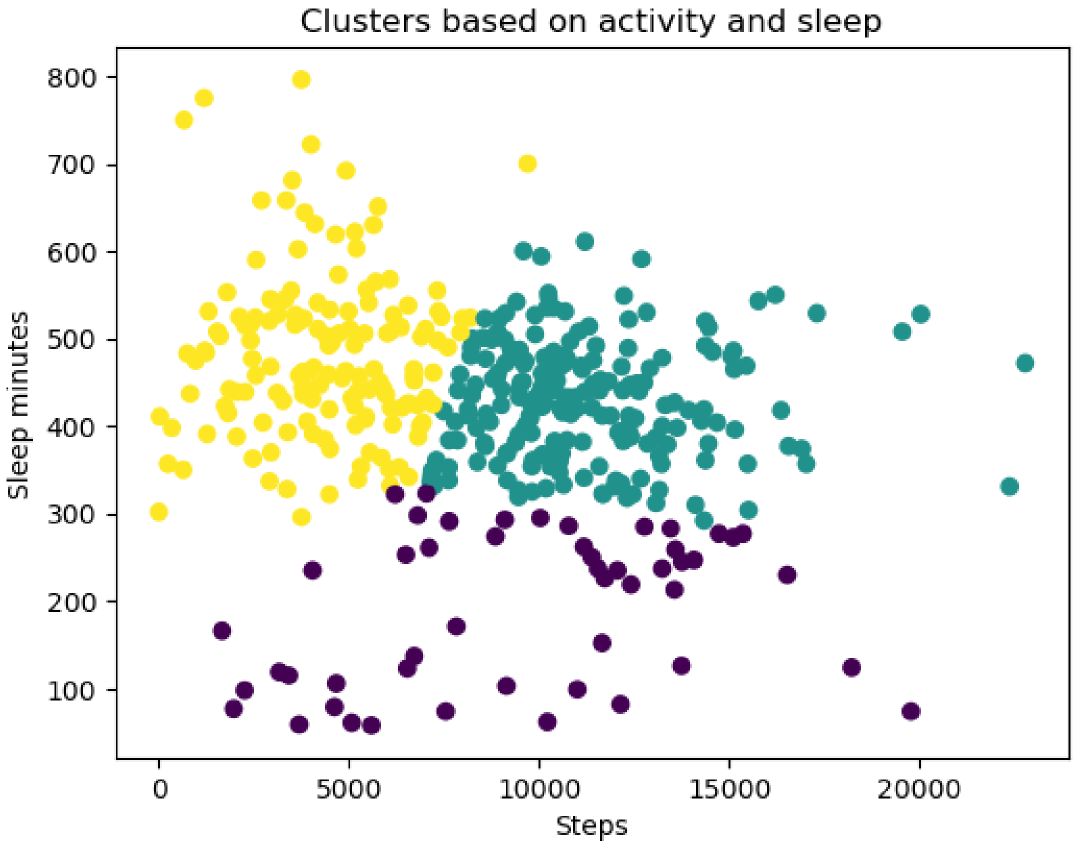
```
[19]: df_sleep = df_sleep.copy()
      df_sleep.loc[:, "TimeInBedHours"] = df_sleep["TotalTimeInBed"] / 60 #change␣
       ↪time to hours

      sns.lmplot(data=df_sleep,␣
       ↪x="TimeInBedHours",y="TotalMinutesAsleep",line_kws={"color": "green"})

      plt.xlabel("Time in bed (hours)")
      plt.ylabel("Minutes asleep")
      plt.title("Time in bed vs. sleep duration")
      plt.show()
```



[20]:
```
from sklearn.preprocessing import StandardScaler
#advanced analysis method the k-means clutering, it was intorduced at the␣
 ↪course assigment material
clustering_data = df[["TotalSteps", "TotalMinutesAsleep"]].dropna() #first we␣
 ↪drop the null values from the columns we are using

scaler = StandardScaler() #using standard scaler to scale the different␣
 ↪measurement types
scaled = scaler.fit_transform(clustering_data)

kmeans = KMeans(n_clusters=3, random_state=42) #3 defines that we want 3␣
 ↪different clusters
clusters = kmeans.fit_predict(scaled)

clustering_data["Cluster"] = clusters
```

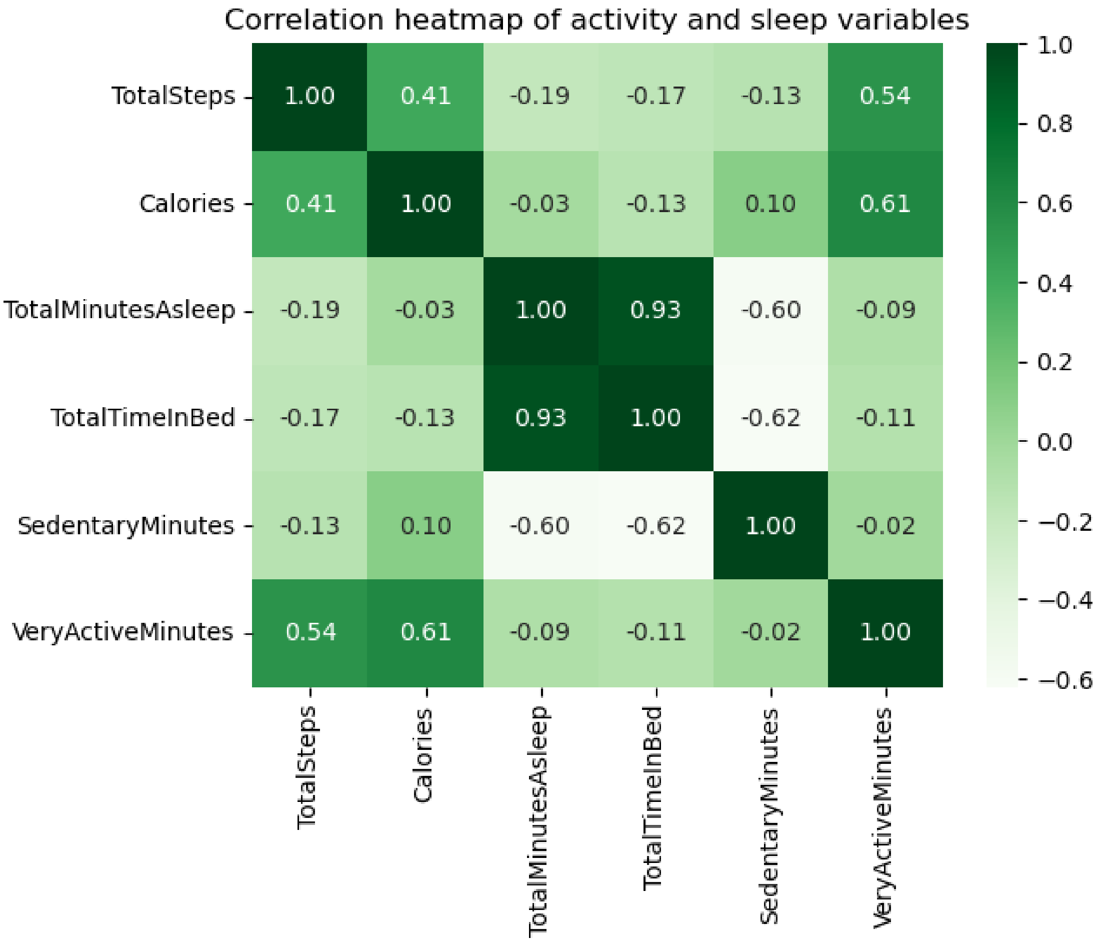
```
plt.scatter(clustering_data["TotalSteps"],
            clustering_data["TotalMinutesAsleep"],
            c=clustering_data["Cluster"])
plt.xlabel("Steps")
plt.ylabel("Sleep minutes")
plt.title("Clusters based on activity and sleep")
plt.show()
```

```
/opt/software/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:1416:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



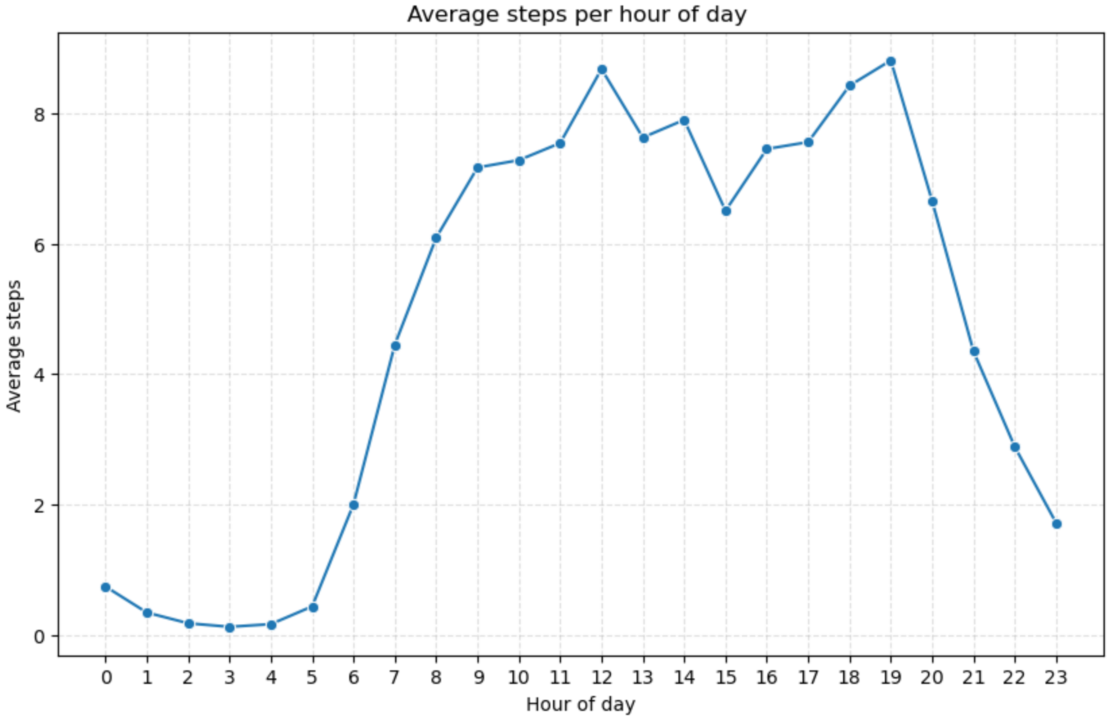
[21]:
```
corr_cols = ["TotalSteps", "Calories", "TotalMinutesAsleep", "TotalTimeInBed",␣
 ↪"SedentaryMinutes","VeryActiveMinutes"]
#corr_cols defines all the columns I want to use in the matrix
corr_df = df[corr_cols].dropna()
#creating the correlation heatmap, corr was introduced in the assigment␣
 ↪materials
```

```
sns.heatmap(corr_df.corr(), annot=True, fmt=".2f", cmap="Greens") #fmt=".2f"␣
↪defines that the matrix values are only presented with two decimals
plt.title("Correlation heatmap of activity and sleep variables")
plt.show()
```

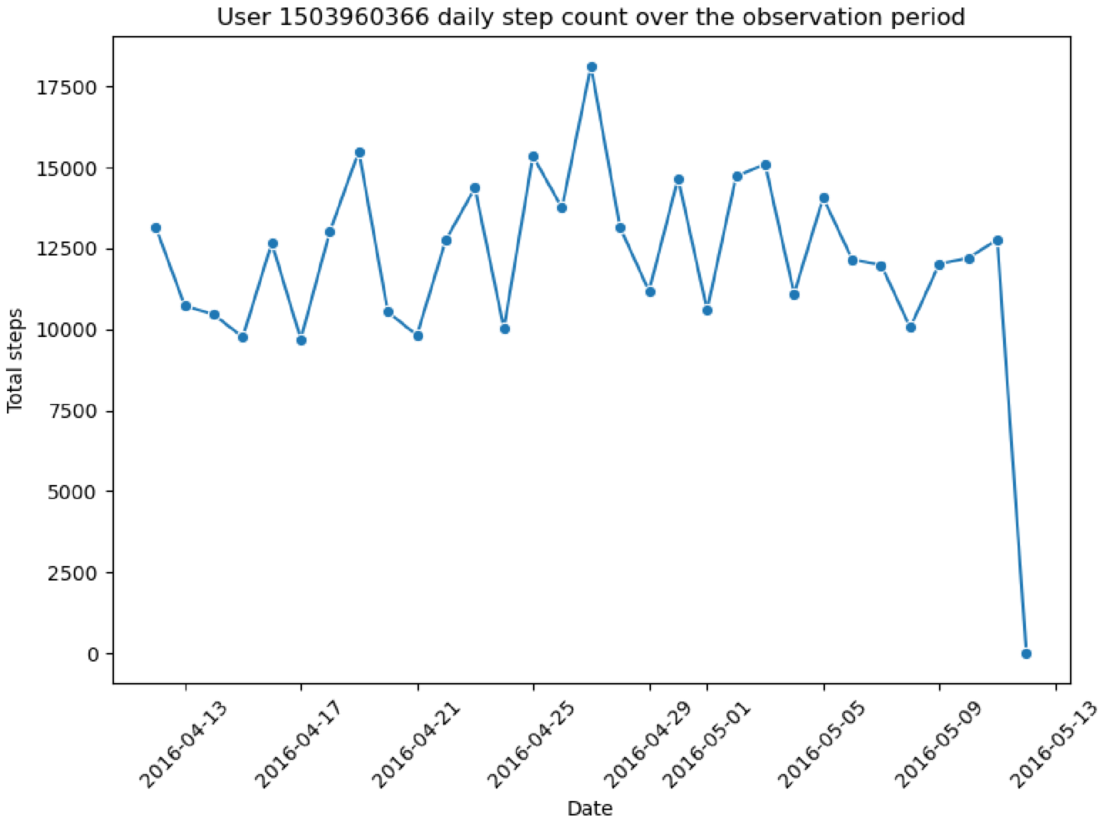


[22]:
```
df_min = pd.read_csv("minuteStepsNarrow_merged.csv")

df_min["ActivityMinute"] = pd.to_datetime(
    df_min["ActivityMinute"],
    format="%m/%d/%Y %I:%M:%S %p" #tranforming the time to correct format
)

df_min["Hour"] = df_min["ActivityMinute"].dt.hour

# calculating the average for hour
hourly_steps = df_min.groupby("Hour")["Steps"].mean().reset_index()
```

```
plt.figure(figsize=(10,6))
sns.lineplot(data=hourly_steps, x="Hour", y="Steps", marker="o")

plt.title("Average steps per hour of day ")
plt.xlabel("Hour of day")
plt.ylabel("Average steps")
plt.xticks(range(0,24))
plt.grid(True, linestyle="--", alpha=0.4)
plt.show()
```



[23]:
```
user_id = 1503960366 #just random user id from the data
df_subject = df[df['Id'] == user_id]
plt.figure(figsize=(8,6))

#individual level visualisation
sns.lineplot(data=df_subject, x = "Date", y = "TotalSteps", marker = "o")

plt.title(f"User 1503960366 daily step count over the observation period")
plt.xlabel("Date")
plt.ylabel("Total steps")
plt.xticks(rotation=45) #this roates the text in x-axel so its is easier to read
plt.tight_layout()
plt.show()
```


User 1503960366 daily step count over the observation period
Total steps
17500
15000
12500
10000
7500
5000
2500
0
2016-04-13
2016-04-17
2016-04-21
2016-04-25
2016-04-29
2016-05-01
2016-05-05
2016-05-09
2016-05-13
Date

[ ]:

[ ]:

[ ]: