Introduzione

Il codice del progetto è open source con licenza MIT ed è possibile trovarlo al seguente link: https://github.com/OB-UNISA/Statistica.

E' possibile visualizzare una versione interattiva online al seguente link:
https://datalore.jetbrains.com/view/notebook/Hwyisb4WAl4X8LTia3qYnV.

I dati unidimensionali esaminati sono quelli del peso corporeo nel periodo di tre anni di un uomo di età 24 attuali.
I dati bidimensionali riguardano il peso corporeo della medesima persona nello stesso periodo e le calorie assunte giornalmente.

Il peso corporeo è stato registrato con due bilance differenti, una per il primo anno e un'altra nei restanti e quindi vi potrebbe essere una differenza di misura.
I dati dei primi mesi sulle calorie assunte non sono molto precisi in quanto la quantità del cibo non veniva pesata, ma approssimata.

Allo stato dell'arte, importando i dati da Samsung Health, è possibile visualizzare le proprie statistiche. Questo potrebbe non funzionare in futuro se la forma dei dati venisse cambiata, ossia il parsing dei dati non funzionerebbe più. I commenti sui dati, tuttavia, risulteranno inefficienti in quanto i più significativi non sono dinamici, bensì basati su un campione di esso.

Poiché i dati sono molti riguardanti il peso corporeo, circa 500, e sulle calorie assunte, quasi 13000, ne verrà preso un campione calcolato nel seguente modo:

- Per il peso corporeo, il campione sarà il peso medio delle pesate in ogni settimana.
  Quindi x_i = peso medio nella settimana i.
- Per le calorie assunte, il campione sarà la media delle calorie assunte in ogni settimana.
  Quindi y_i = media delle calorie assunte nella settimana i.

Nota: Se nella settimana i non vi sono dati sul peso, essa verrà saltata e quindi anche le rispettive calorie assunte.

Si è deciso di usare la media campionaria e non la mediana perché i picchi di valore sono importanti da considerare.

Installazione dipendenze

In [275…

```
!pip install matplotlib
!pip install pandas
```

```
Requirement already satisfied: matplotlib in c:\python397\lib\site-packages (3.5.2)
Requirement already satisfied: fonttools>=4.22.0 in c:\python397\lib\site-packages (from matplotlib) (4.33.3)
Requirement already satisfied: numpy>=1.17 in c:\python397\lib\site-packages (from matplotlib) (1.22.4)
Requirement already satisfied: python-dateutil>=2.7 in c:\python397\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: cycler>=0.10 in c:\python397\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\python397\lib\site-packages (from matplotlib) (1.4.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:\python397\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: packaging>=20.0 in c:\python397\lib\site-packages (from matplotlib) (21.3)
Requirement already satisfied: pillow>=6.2.0 in c:\python397\lib\site-packages (from matplotlib) (9.1.1)
Requirement already satisfied: six>=1.5 in c:\python397\lib\site-packages (from python-dateutil>=2.7->matplotli
b) (1.16.0)
Requirement already satisfied: pandas in c:\python397\lib\site-packages (1.4.2)
Requirement already satisfied: numpy>=1.18.5 in c:\python397\lib\site-packages (from pandas) (1.22.4)
Requirement already satisfied: pytz>=2020.1 in c:\python397\lib\site-packages (from pandas) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\python397\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\python397\lib\site-packages (from python-dateutil>=2.8.1->pandas)
(1.16.0)
```

In [276…

```
import matplotlib.pyplot as plt
import pandas as pd
import csv
from math import sqrt

plt.rcParams['figure.figsize'] = [8, 6]
plt.rcParams['figure.dpi'] = 100
```

Parsing dei dati del peso corporeo

È stato necessario arrotondare i valori del peso corporeo perché altrimenti si avrebbe avuto il numero di modalità del carattere quasi uguale all'ampiezza del dato.

In [277…

```
with open('peso.csv', 'r') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    peso = []
    reader.__next__()
    reader.__next__()
    for row in reader:
        peso.append([row[1], float(row[4])])

df_peso = pd.DataFrame(peso, columns=['data', 'peso'])
df_peso['data'] = pd.to_datetime(df_peso['data'])
df_peso = df_peso.groupby(pd.Grouper(key='data', freq='W')).mean().round(0)
df_peso = df_peso.dropna()
df_peso = df_peso.reset_index()
df_peso
```

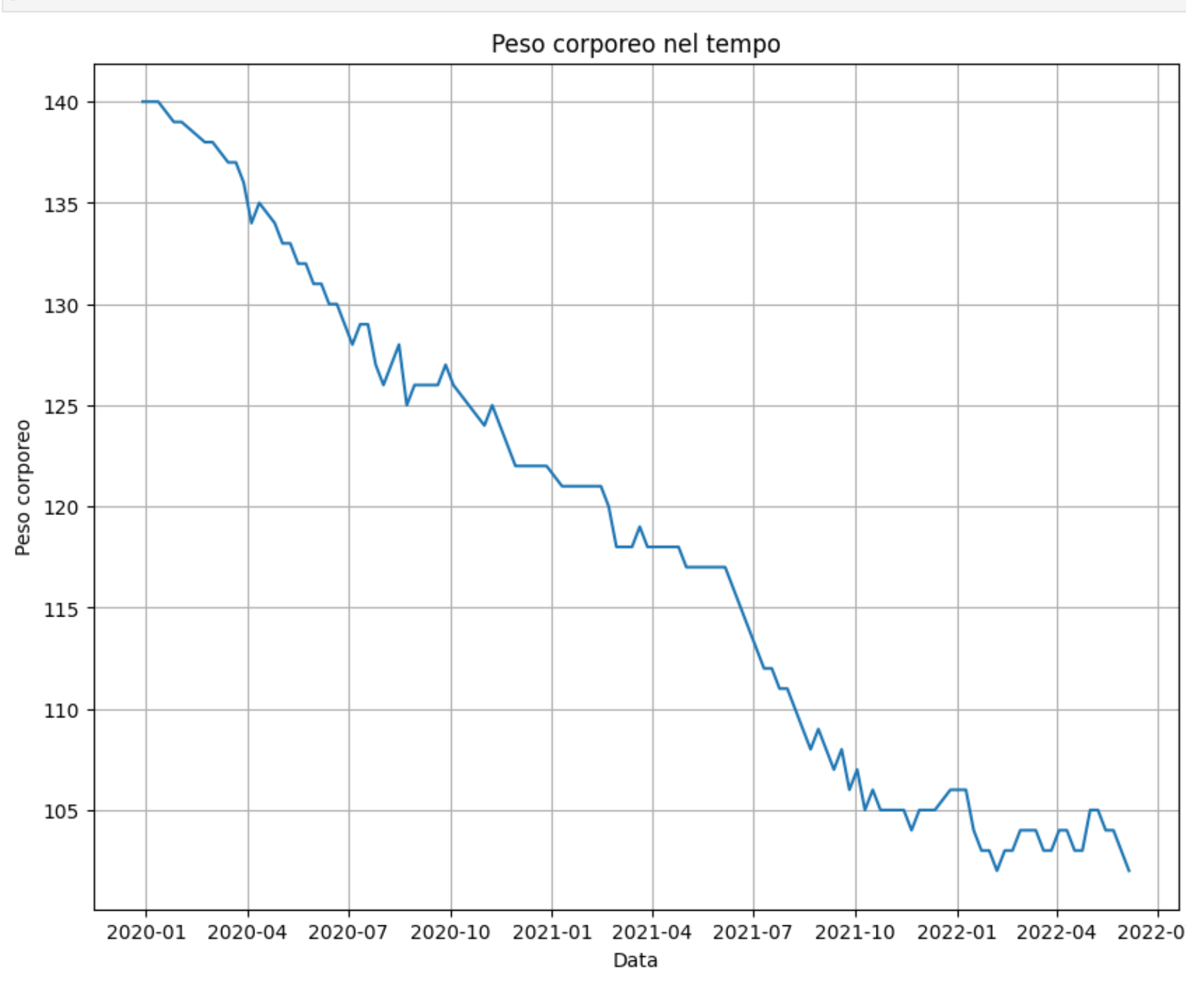
Out[277]:

	data	peso
0	2019-12-29	140.0
1	2020-01-12	140.0
2	2020-01-26	139.0
3	2020-02-02	139.0
4	2020-02-23	138.0
...	...	...
111	2022-05-08	105.0
112	2022-05-15	104.0
113	2022-05-22	104.0
114	2022-05-29	103.0
115	2022-06-05	102.0

116 rows × 2 columns

In [278…

```
fig, ax = plt.subplots(figsize=(10, 8))
ax.plot(df_peso['data'], df_peso['peso'])
ax.set_title('Peso corporeo nel tempo')
ax.set_xlabel('Data')
ax.set_ylabel('Peso corporeo')
ax.grid()
plt.show()
```



Sono state eliminate le settimane in cui non vi erano dati.

In [279…

```
x = sorted(int(p) for p in df_peso['peso'])
n = len(x)
print(f'Ampiezza del dato: {n}')
for i, x_i in enumerate(x):
    print(f'x_({i + 1})={x_i}', end=' ')
```

```
Ampiezza del dato: 116
x_(1)=102 x_(2)=102 x_(3)=103 x_(4)=103 x_(5)=103 x_(6)=103 x_(7)=103 x_(8)=103 x_(9)=103 x_(10)=103 x_(11)=103
x_(12)=104 x_(13)=104 x_(14)=104 x_(15)=104 x_(16)=104 x_(17)=104 x_(18)=104 x_(19)=104 x_(20)=104 x_(21)=105 x
_(22)=105 x_(23)=105 x_(24)=105 x_(25)=105 x_(26)=105 x_(27)=105 x_(28)=105 x_(29)=105 x_(30)=105 x_(31)=106 x_
(32)=106 x_(33)=106 x_(34)=106 x_(35)=106 x_(36)=107 x_(37)=107 x_(38)=108 x_(39)=108 x_(40)=108 x_(41)=109 x_
(42)=109 x_(43)=110 x_(44)=111 x_(45)=111 x_(46)=112 x_(47)=112 x_(48)=113 x_(49)=114 x_(50)=115 x_(51)=116 x_
(52)=117 x_(53)=117 x_(54)=117 x_(55)=117 x_(56)=117 x_(57)=117 x_(58)=118 x_(59)=118 x_(60)=118 x_(61)=118 x_
(62)=118 x_(63)=118 x_(64)=118 x_(65)=118 x_(66)=119 x_(67)=120 x_(68)=121 x_(69)=121 x_(70)=121 x_(71)=121 x_
(72)=122 x_(73)=122 x_(74)=122 x_(75)=122 x_(76)=122 x_(77)=123 x_(78)=124 x_(79)=124 x_(80)=125 x_(81)=125 x_
(82)=125 x_(83)=126 x_(84)=126 x_(85)=126 x_(86)=126 x_(87)=126 x_(88)=126 x_(89)=127 x_(90)=127 x_(91)=127 x_
(92)=128 x_(93)=128 x_(94)=129 x_(95)=129 x_(96)=129 x_(97)=130 x_(98)=130 x_(99)=131 x_(100)=131 x_(101)=132 x
_(102)=132 x_(103)=133 x_(104)=133 x_(105)=134 x_(106)=134 x_(107)=135 x_(108)=136 x_(109)=137 x_(110)=137 x_(1
11)=138 x_(112)=138 x_(113)=139 x_(114)=139 x_(115)=140 x_(116)=140
```

Costruzione della tabella delle frequenze

Calcolo delle modalità

In [280…

```
v_x = [x[0]]
for i in range(1, len(x)):
    if v_x[-1] != x[i]:
        v_x.append(x[i])

k_x = len(v_x)
print(f'Numero di modalità: {k_x}')

for i, v_i in enumerate(v_x):
    print(f'v_{i + 1}={v_i}', end=' ')
```

```
Numero di modalità: 39
v_1=102 v_2=103 v_3=104 v_4=105 v_5=106 v_6=107 v_7=108 v_8=109 v_9=110 v_10=111 v_11=112 v_12=113 v_13=114 v_1
4=115 v_15=116 v_16=117 v_17=118 v_18=119 v_19=120 v_20=121 v_21=122 v_22=123 v_23=124 v_24=125 v_25=126 v_26=1
27 v_27=128 v_28=129 v_29=130 v_30=131 v_31=132 v_32=133 v_33=134 v_34=135 v_35=136 v_36=137 v_37=138 v_38=139
v_39=140
```

Calcolo della frequenza assoluta delle modalità

Dalle frequenze assolute si può notare quando si è avuto più difficoltà nella perdita del peso o quando si era stabilizzato per un periodo di tempo.

In [281…

```
f_x = [1]
for i in range(1, n):
    if x[i - 1] == x[i]:
        f_x[-1] += 1
    else:
        f_x.append(1)

for i, f_i in enumerate(f_x):
    print(f'f_{i + 1}={f_i}', end=' ')
```

```
f_1=2 f_2=9 f_3=9 f_4=10 f_5=5 f_6=2 f_7=3 f_8=2 f_9=1 f_10=2 f_11=2 f_12=1 f_13=1 f_14=1 f_15=1 f_16=6 f_17=8
f_18=1 f_19=1 f_20=4 f_21=5 f_22=1 f_23=2 f_24=3 f_25=6 f_26=3 f_27=2 f_28=3 f_29=2 f_30=2 f_31=2 f_32=2 f_33=2
f_34=1 f_35=1 f_36=2 f_37=2 f_38=2 f_39=2
```

Calcolo della frequenza cumulativa assoluta delle modalità

In [282…

```
F_x = [f_x[0]]
for i in range(1, k_x):
    # Uso della relazione di ricorrenza
    F_x.append(F_x[-1] + f_x[i])

for i, F_i in enumerate(F_x):
    print(f'F_{i + 1}={F_i}', end=' ')
```

```
F_1=2 F_2=11 F_3=20 F_4=30 F_5=35 F_6=37 F_7=40 F_8=42 F_9=43 F_10=45 F_11=47 F_12=48 F_13=49 F_14=50 F_15=51 F
_16=57 F_17=65 F_18=66 F_19=67 F_20=71 F_21=76 F_22=77 F_23=79 F_24=82 F_25=88 F_26=91 F_27=93 F_28=96 F_29=98
F_30=100 F_31=102 F_32=104 F_33=106 F_34=107 F_35=108 F_36=110 F_37=112 F_38=114 F_39=116
```

Calcolo della frequenza relativa delle modalità

In [283…

```
p_x = []
for f_i in f_x:
    p_x.append(f_i / n)

for i, p_i in enumerate(p_x):
    print(f'p_{i + 1}={p_i:.3f}', end=' ')
```

```
p_1=0.017 p_2=0.078 p_3=0.078 p_4=0.086 p_5=0.043 p_6=0.017 p_7=0.026 p_8=0.017 p_9=0.009 p_10=0.017 p_11=0.017
p_12=0.009 p_13=0.009 p_14=0.009 p_15=0.009 p_16=0.052 p_17=0.069 p_18=0.009 p_19=0.009 p_20=0.034 p_21=0.043 p
_22=0.009 p_23=0.017 p_24=0.026 p_25=0.052 p_26=0.026 p_27=0.017 p_28=0.026 p_29=0.017 p_30=0.017 p_31=0.017 p_
32=0.017 p_33=0.017 p_34=0.009 p_35=0.009 p_36=0.017 p_37=0.017 p_38=0.017 p_39=0.017
```

Calcolo della frequenza cumulativa relativa delle modalità

In [284…

```
P_x = []
for F_i in F_x:
    P_x.append(F_i / n)

for i, P_i in enumerate(P_x):
    print(f'P_{i + 1}={P_i:.3f}', end=' ')
```

```
P_1=0.017 P_2=0.095 P_3=0.172 P_4=0.259 P_5=0.302 P_6=0.319 P_7=0.345 P_8=0.362 P_9=0.371 P_10=0.388 P_11=0.405
P_12=0.414 P_13=0.422 P_14=0.431 P_15=0.440 P_16=0.491 P_17=0.560 P_18=0.569 P_19=0.578 P_20=0.612 P_21=0.655 P
_22=0.664 P_23=0.681 P_24=0.707 P_25=0.759 P_26=0.784 P_27=0.802 P_28=0.828 P_29=0.845 P_30=0.862 P_31=0.879 P_
32=0.897 P_33=0.914 P_34=0.922 P_35=0.931 P_36=0.948 P_37=0.966 P_38=0.983 P_39=1.000
```

Tabella delle frequenze

In [285…

```
data_table = []
for i in range(k_x):
    data_table.append([i + 1, v_x[i], f_x[i], p_x[i], F_x[i], P_x[i]])

df_data_table = pd.DataFrame(data_table, columns=['i', 'v_i', 'f_i', 'p_i', 'F_i', 'P_i'])
df_data_table.set_index('i', inplace=True)
df_data_table
```

Out[285]:

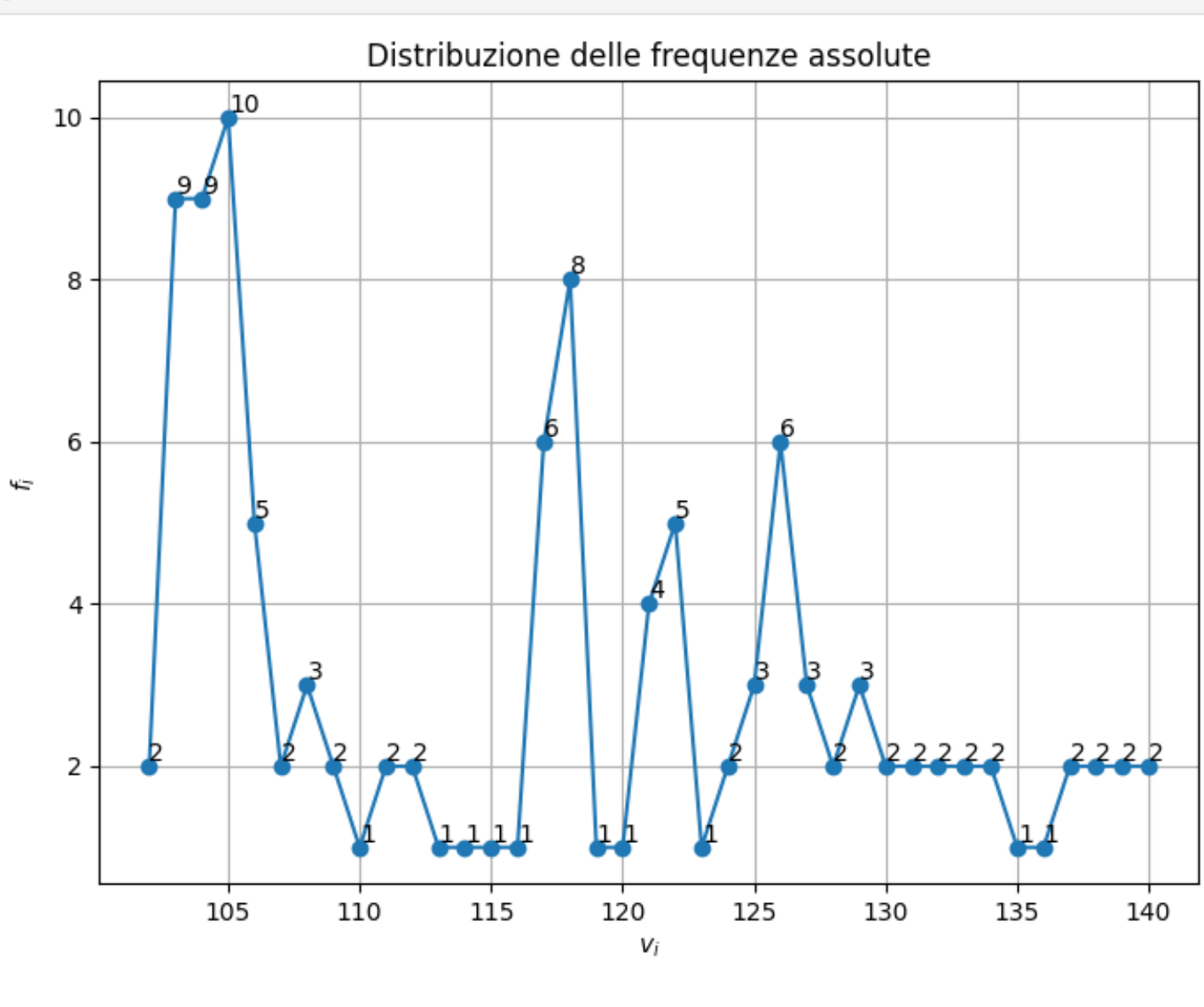
	v_i	f_i	p_i	F_i	P_i
i					
1	102	2	0.017241	2	0.017241
2	103	9	0.077586	11	0.094828
3	104	9	0.077586	20	0.172414
4	105	10	0.086207	30	0.258621
5	106	5	0.043103	35	0.301724
6	107	2	0.017241	37	0.318966
7	108	3	0.025862	40	0.344828
8	109	2	0.017241	42	0.362069
9	110	1	0.008621	43	0.370690
10	111	2	0.017241	45	0.387931
11	112	2	0.017241	47	0.405172
12	113	1	0.008621	48	0.413793
13	114	1	0.008621	49	0.422414
14	115	1	0.008621	50	0.431034
15	116	1	0.008621	51	0.439655
16	117	6	0.051724	57	0.491379
17	118	8	0.068966	65	0.560345
18	119	1	0.008621	66	0.568966
19	120	1	0.008621	67	0.577586
20	121	4	0.034483	71	0.612069
21	122	5	0.043103	76	0.655172
22	123	1	0.008621	77	0.663793
23	124	2	0.017241	79	0.681034
24	125	3	0.025862	82	0.706897
25	126	6	0.051724	88	0.758621
26	127	3	0.025862	91	0.784483
27	128	2	0.017241	93	0.801724
28	129	3	0.025862	96	0.827586
29	130	2	0.017241	98	0.844828
30	131	2	0.017241	100	0.862069
31	132	2	0.017241	102	0.879310
32	133	2	0.017241	104	0.896552
33	134	2	0.017241	106	0.913793
34	135	1	0.008621	107	0.922414
35	136	1	0.008621	108	0.931034
36	137	2	0.017241	110	0.948276
37	138	2	0.017241	112	0.965517
38	139	2	0.017241	114	0.982759
39	140	2	0.017241	116	1.000000

Grafici distribuzione delle frequenze assolute

Grafico a linee

In [286…

```
fig, ax = plt.subplots()
ax.plot(v_x, f_x, '-o')
ax.set_title('Distribuzione delle frequenze assolute')
ax.set_xlabel(r'$v_i$')
ax.set_ylabel(r'$f_i$')
ax.grid()

for i, v_i in enumerate(v_x):
    ax.text(v_i, f_x[i], f'{f_x[i]:.0f}', ha='left', va='bottom')

plt.show()
```



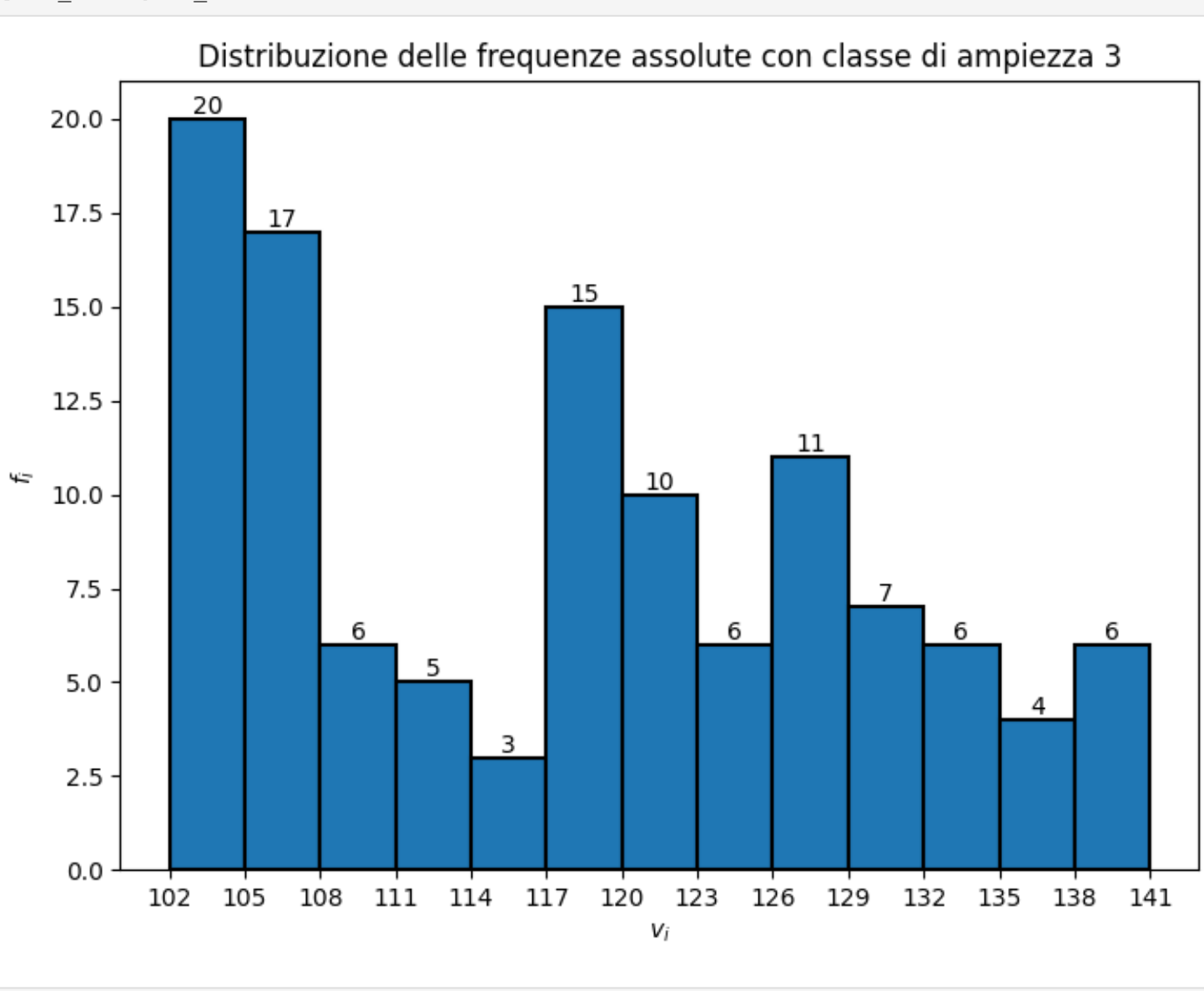
Istogramma

In [287…

```
def plot_histogram_classe(ampiezza_classe):
    min_value = x[0]
    max_value = x[n - 1]

    fig, ax = plt.subplots()
    values, bins, bars = ax.hist(x, bins=list(range(min_value, max_value + ampiezza_classe, ampiezza_classe)), 

    ax.set_xticks(bins)
    ax.bar_label(bars)
    ax.set_title(f'Distribuzione delle frequenze assolute con classe di ampiezza {ampiezza_classe}')
    ax.set_xlabel(r'$v_i$')
    ax.set_ylabel(r'$f_i$')

    plt.show()
```

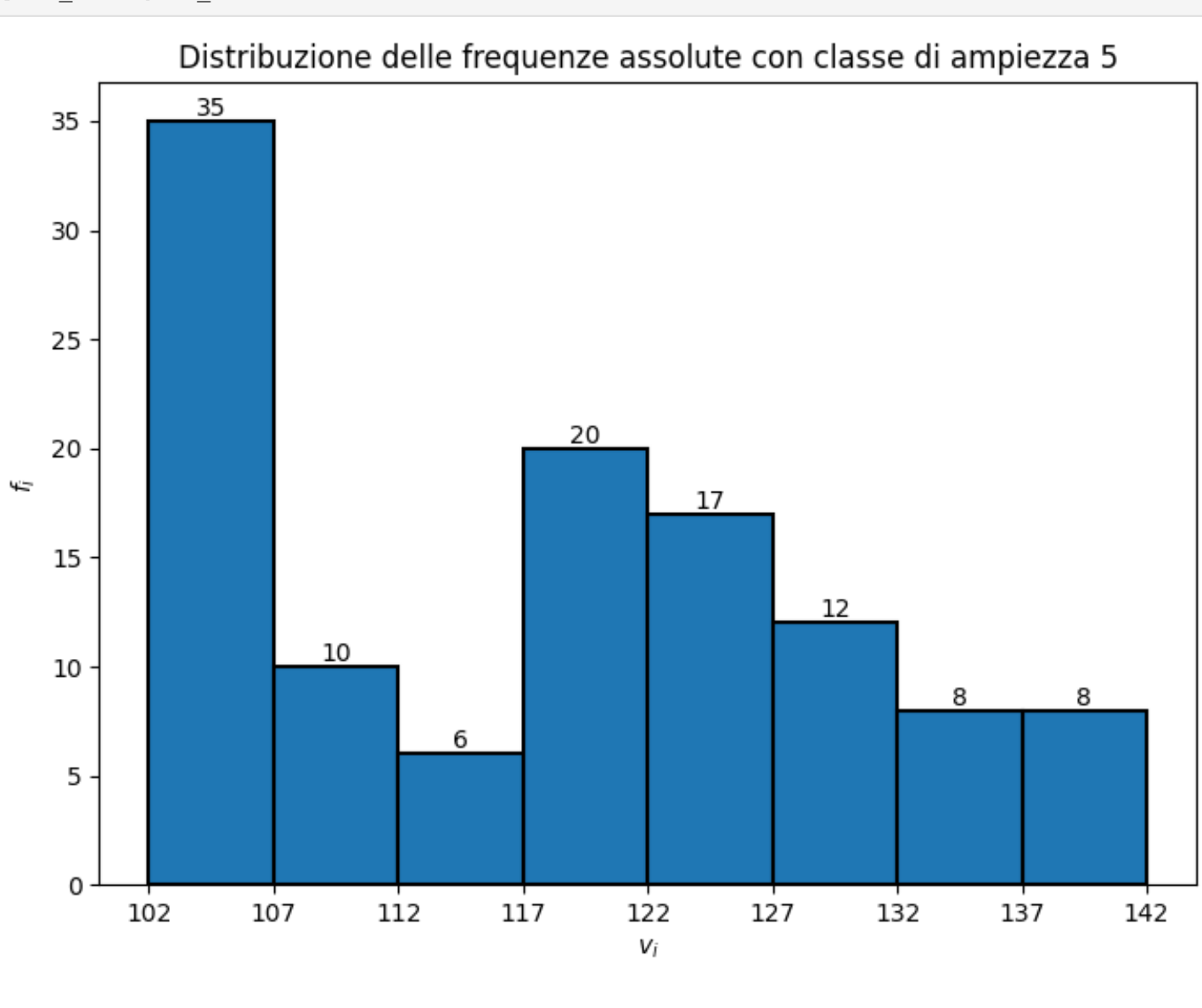
In [288…

```
plot_histogram_classe(3)
```



In [289…

```
plot_histogram_classe(5)
```



Indici di posizione

Calcolo della media campionaria

L'indice della media campionare su tutto il campione è poco significativo in questo campione perché, come si nota dal grafico dal peso corporeo nel tempo, il peso sempre sceso e quindi i valori sono cambiati spesso. La media campionaria invece indicherebbe che si è avuto una media di peso corporeo per ogni settimana di 117 kg, cosa non vera.

In [290…

```
x_mean = 0
for x_i in x:
    x_mean += x_i
x_mean = x_mean / n

print(f'Media campionaria: {x_mean}')
```

```
Media campionaria: 117.42241379310344
```

Calcolo della media pesata

In [291…

```
x_mean_pesata = 0
for i in range(k_x):
    x_mean_pesata += (f_x[i] / n) * v_x[i]

print(f'Media pesata: {x_mean_pesata}')
```

```
Media pesata: 117.42241379310344
```

Calcolo della mediana campionaria

Anche la mediana campionaria per la tipologia del dato, è poco signficativa.

In [292…

```
if n % 2 == 0:
    x_mediana = (x[n // 2] + x[n // 2 - 1]) / 2
else:
    x_mediana = x[n // 2]

print(f'Mediana campionaria: {x_mediana}')
```

```
Mediana campionaria: 118.0
```

Calcolo della moda campionaria

La moda campionaria è molto utile, in quanto come detto precedentemente, si riesce a determinare in quali fasi del peso corporeo si è avuto più difficoltà. Calcolando più mode campionarie, escludendo ogni volta quella precedente, si riesce a capire quali sono le fasi del peso corporeo in cui si è avuto più difficoltà.

In [293…

```
max_f_x = 0
index = 0
count = 0

for i, f_i in enumerate(f_x):
    if f_i > max_f_x:
        max_f_x = f_i
        count = 1
        index = i
    elif f_i == max_f_x:
        count += 1

x_moda = v_x[index]
if count == 1:
    print(f'Moda campionaria unimodale: v_{index + 1} = {x_moda}, f_{index + 1} = {max_f_x}')
elif count == 2:
    print(f'Moda campionaria bimodale: v_{index + 1} = {x_moda}, f_{index + 1} = {max_f_x}')
else:
    print(f'Moda campionaria multimodale: v_{index + 1} = {x_moda}, f_{index + 1} = {max_f_x}')
```

```
Moda campionaria unimodale: v_4 = 105, f_4 = 10
```

Indici di variabilità

Calcolo della varianza campionaria

La varianza dei dati rispetto alla media campionaria è un po' alta.

In [294…

```
s2_x = 0
for x_i in x:
    s2_x += (x_i - x_mean) ** 2

s2_x = s2_x / (n - 1)
print(f'Varianza campionaria: {s2_x}')
```

```
Varianza campionaria: 132.08958020989505
```

Calcolo della deviazione standard campionaria

In [295…

```
s_x = sqrt(s2_x)
print(f'Deviazione standard campionaria: {s_x}')
```

```
Deviazione standard campionaria: 11.493023110126206
```

Calcolo dello scarto medio assoluto

In [296…

```
sa_x = 0
for x_i in x:
    sa_x += abs(x_i - x_mean)

sa_x = sa_x / n
print(f'Scarto medio assoluto: {sa_x}')
```

```
Scarto medio assoluto: 9.897889417360267
```

Calcolo dell'ampiezza del campo di variazione

Questo dato indica anche il massimo di chili persi in tutto il periodo.

In [297…

```
w_x = x[n - 1] - x[0]
print(f'Ampiezza del campo di variazione: {w_x}')
```

```
Ampiezza del campo di variazione: 38
```

Calcolo del coefficiente di variazione

In [298…

```
cv_x = s_x / x_mean
print(f'Coefficiente di variazione: {cv_x}')
```

```
Coefficiente di variazione: 0.09787759201047205
```

Indici di forma

Calcolo dell'indice di simmetria

Dall'istogramma si nota una coda a destra, infatti l'indice è positivo.

In [299…

```
g_x = 0
for x_i in x:
    g_x += (x_i - x_mean) ** 3

g_x = g_x / n
g_x = g_x / (s_x ** 3)

if g_x < 0:
    print(f'Indice di simmetria: {g_x}, asimmetria negativa')
else:
    print(f'Indice di simmetria: {g_x}, asimmetria positiva')
```

```
Indice di simmetria: 0.25738968823736946, asimmetria positiva
```

In [300…

```
plot_histogram_classe(1)
```

Calcolo dell'indice di curtosi

In [301…

```
curtosi_x = 0
for x_i in x:
    curtosi_x += (x_i - x_mean) ** 4

curtosi_x = curtosi_x / n
curtosi_x = curtosi_x / (s_x ** 4)
curtosi_x = curtosi_x - 3

if curtosi_x > 0:
    print(f'Indice di curtosi: {curtosi_x}, è presente un eccesso di dati nelle classi centrali')
elif curtosi_x < 0:
    print(f'Indice di curtosi: {curtosi_x}, è presente una carenza di dati nelle classi centrali')
else:
    print(f'Indice di curtosi: {curtosi_x}, è presente una distribuzione di dati come quella di una distribuzio
```

```
Indice di curtosi: -1.2037434028051623, è presente una carenza di dati nelle classi centrali
```

Quartili

Calcolo dei quartili

In [302…

```
def get_quartile(k):
    np = n * (k / 100)
    if np % 1 == 0:
        return (x[int(np)] + x[int(np) - 1]) / 2
    else:
        return x[int(np)]
```

In [303…

```
Q1_x = get_quartile(25)
Q2_x = get_quartile(50)
Q3_x = get_quartile(75)

print(f'Principali quartili: Q1 = {Q1_x}, Q2 = {Q2_x}, Q3 = {Q3_x}')
```

```
Principali quartili: Q1 = 105.0, Q2 = 118.0, Q3 = 126.0
```

Calcolo dello scarto interquartile

Indica la lunghezza del rettangolo nel box plot

In [304…

```
si_x = Q3_x - Q1_x
print(f'Scarto interquartile: {si_x}')
```

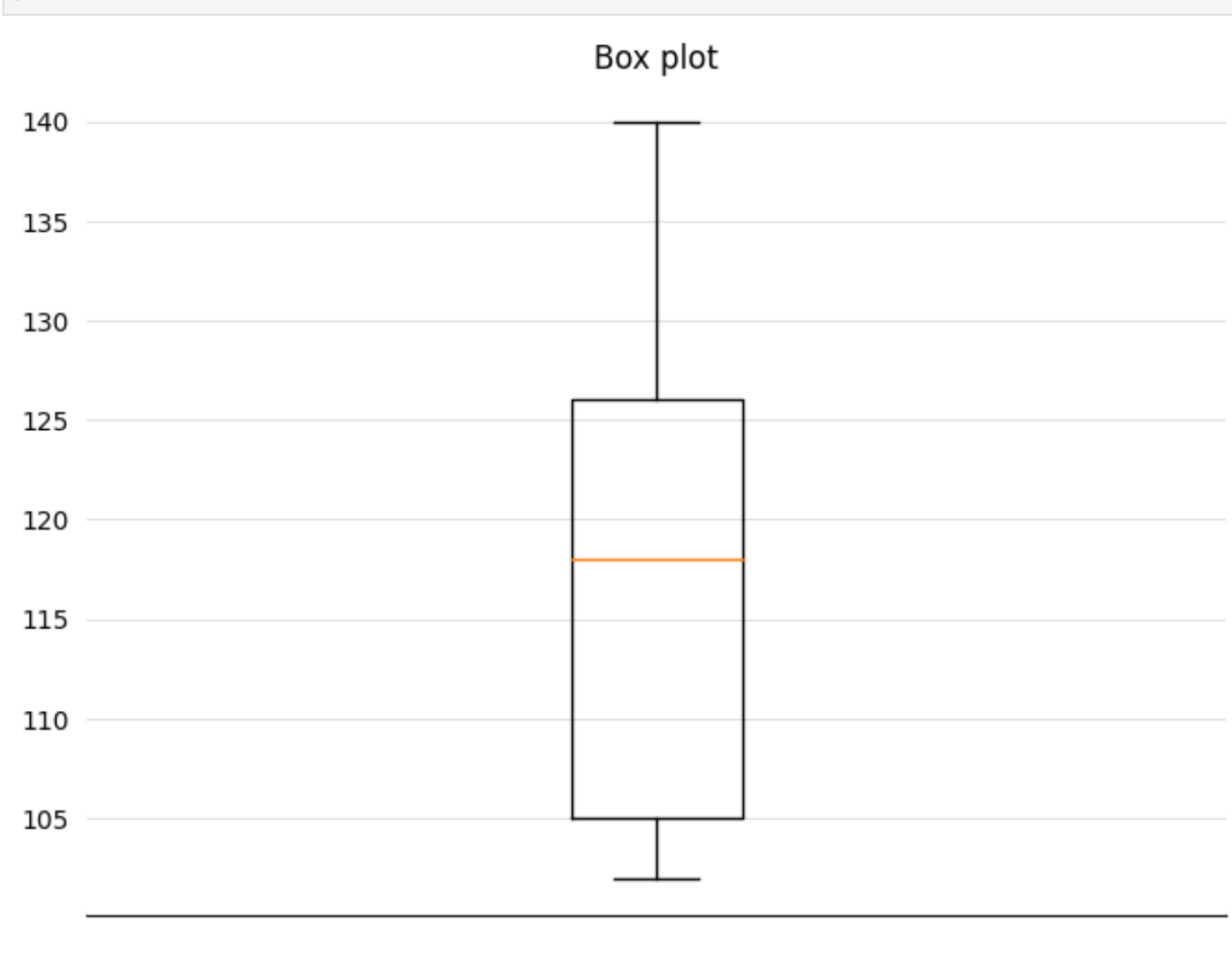
```
Scarto interquartile: 21.0
```

Box plot

Come si evince dal grafico, non vi sono outliers. Se essi fosse presenti, sarebbero indicati nel grafico come dei punti rossi. Il segmento arancione rappresenta la media campionaria.

In [305…

```
fig, ax = plt.subplots()
ax.boxplot(x, flierprops=dict(marker='s', markerfacecolor='red'))
ax.set_title('Box plot')
ax.yaxis.set_ticks_position('none')
ax.grid(axis='y', linestyle='-', linewidth=0.5, color='lightgrey')
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
ax.get_xaxis().set_visible(False)
plt.show()
```



Intervalli di previsione

Gli intervalli di previsione non sono quelli tipici di una distribuzione normale, ma questo lo si poteva già evincere dall'istogramma delle classi di ampiezza 1.

In [306…

```
intervallo1_x = (x_mean - s_x, x_mean + s_x)
intervallo2_x = (x_mean - 2 * s_x, x_mean + 2 * s_x)
intervallo3_x = (x_mean - 3 * s_x, x_mean + 3 * s_x)

count1 = 0
count2 = 0
count3 = 0
for x_i in x:
    if intervallo1_x[0] < x_i < intervallo1_x[1]:
        count1 += 1
    if intervallo2_x[0] < x_i < intervallo2_x[1]:
        count2 += 1
    if intervallo3_x[0] < x_i < intervallo3_x[1]:
        count3 += 1

print(f'Intervalli di previsione:\n{intervallo1_x} {(count1 / n) * 100:.2f}%\n{intervallo2_x} {(count2 / n) * 1
```

```
Intervalli di previsione:
(105.92939068297724, 128.91543690322965) 54.31%
(94.43636757285103, 140.40846001335586) 100.00%
(82.94334446272482, 151.90148312348208) 100.00%
```

Parsing dei dati delle calorie assunte

In [307…

```
with open('calorie.csv', 'r') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    calorie = []
    reader.__next__()
    reader.__next__()
    for row in reader:
        calorie.append([row[0], float(row[8])])

df_calorie = pd.DataFrame(calorie, columns=['data', 'calorie'])
df_calorie['data'] = pd.to_datetime(df_calorie['data'])
df_calorie = df_calorie.groupby(pd.Grouper(key='data', freq='D')).sum().round(0)
df_calorie = df_calorie.reset_index()
df_calorie = df_calorie.groupby(pd.Grouper(key='data', freq='W')).mean().round(0)
df_calorie = df_calorie.reset_index()
print(f'Numero di ampiezza dei dati prima della eliminazione delle settimane in cui non vi sono state pesate co
df_calorie
```

```
Numero di ampiezza dei dati prima della eliminazione delle settimane in cui non vi sono state pesate corporee:
138
```

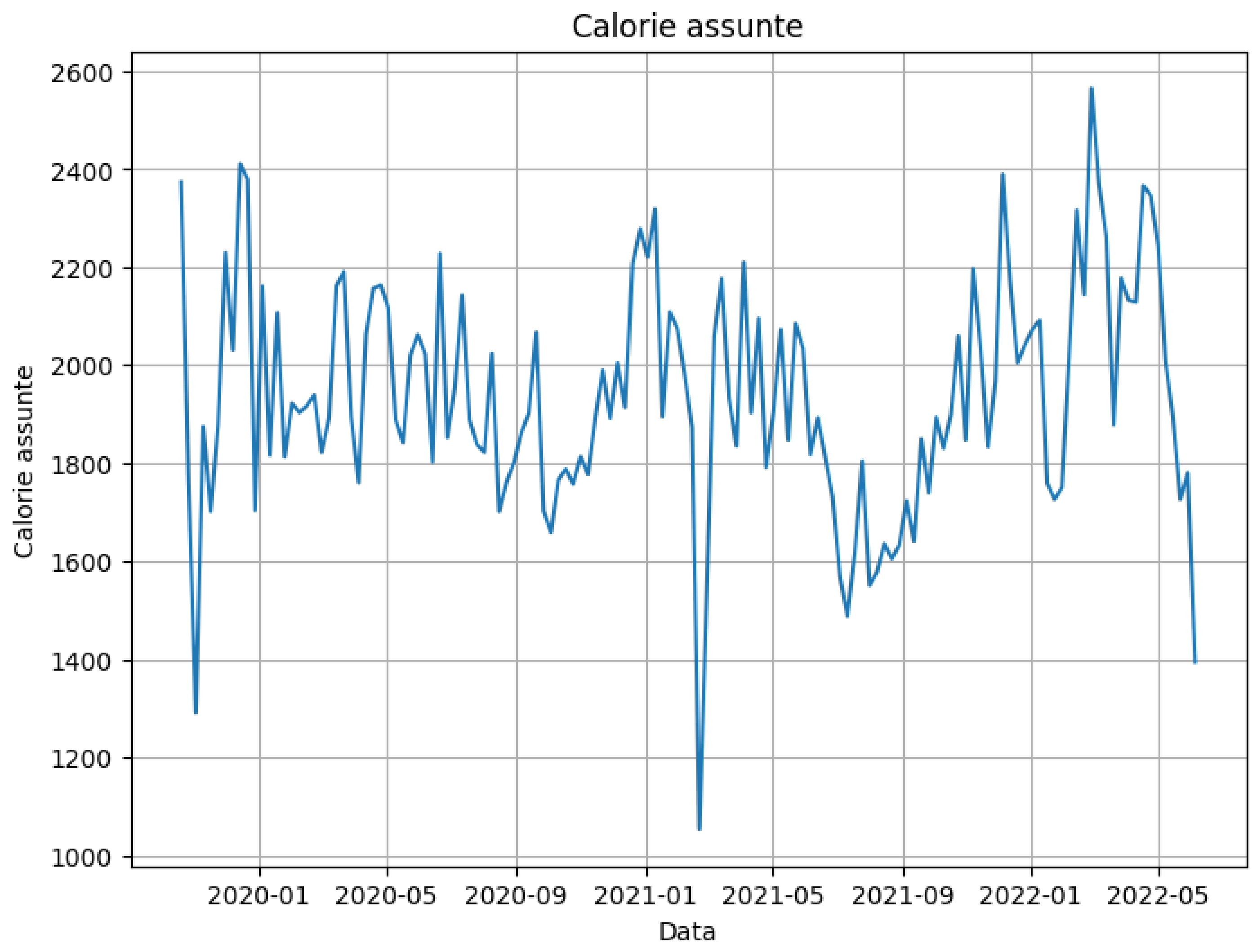
Out[307]:

	data	calorie
0	2019-10-20	2374.0
1	2019-10-27	1768.0
2	2019-11-03	1291.0
3	2019-11-10	1875.0
4	2019-11-17	1702.0
...	...	...
133	2022-05-08	2008.0
134	2022-05-15	1894.0
135	2022-05-22	1727.0
136	2022-05-29	1780.0
137	2022-06-05	1394.0

138 rows × 2 columns

In [308…

```
fig, ax = plt.subplots()
ax.plot(df_calorie['data'], df_calorie['calorie'])
ax.set_title('Calorie assunte')
ax.set_xlabel('Data')
ax.set_ylabel('Calorie assunte')
ax.grid()
plt.show()
```



In [309…

```
df_merge = pd.merge(df_peso, df_calorie, on='data', how='inner')
df_merge = df_merge.dropna()
df_merge = df_merge.reset_index()

print(f'Numero di ampiezza dei dati dopo l\'eliminazione delle settimane in cui non vi sono state pesate corpor
df_merge
```

```
Numero di ampiezza dei dati dopo l'eliminazione delle settimane in cui non vi sono state pesate corporee: 116
```

Out[309]:

	index	data	peso	calorie
0	0	2019-12-29	140.0	1704.0
1	1	2020-01-12	140.0	1816.0
2	2	2020-01-26	139.0	1813.0
3	3	2020-02-02	139.0	1922.0
4	4	2020-02-23	138.0	1939.0
...	...	...	...	...
111	111	2022-05-08	105.0	2008.0
112	112	2022-05-15	104.0	1894.0
113	113	2022-05-22	104.0	1727.0
114	114	2022-05-29	103.0	1780.0
115	115	2022-06-05	102.0	1394.0

116 rows × 4 columns

Dati bidimensionali

Calcolo dei coefficienti di correlazione campionario

In [310…

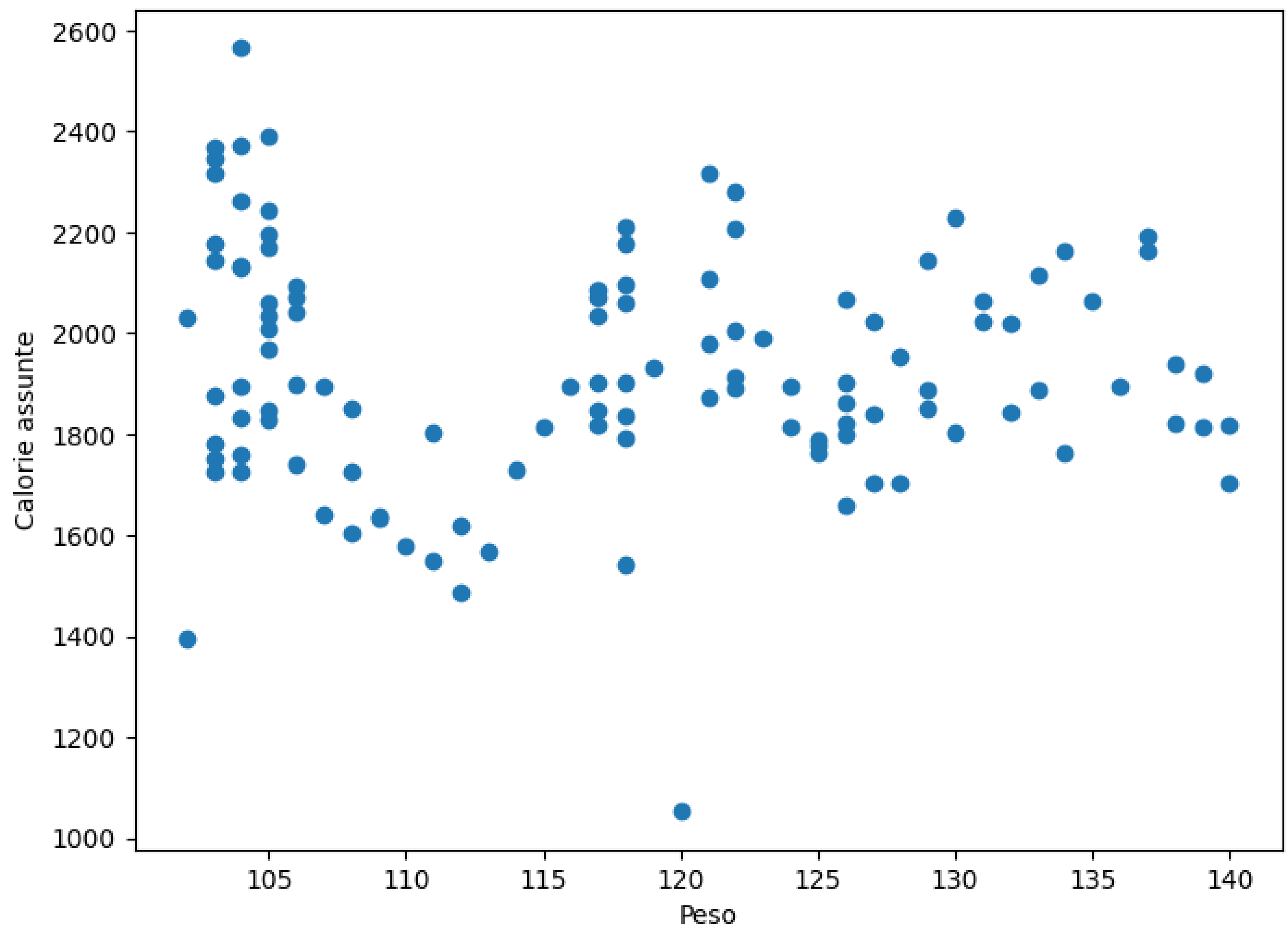
```
n = len(df_merge)
peso_mean = df_merge['peso'].mean()
calorie_mean = df_merge['calorie'].mean()
s_peso = df_merge['peso'].std()
s_calorie = df_merge['calorie'].std()

r = 0
for i in range(n):
    r += (df_merge['peso'][i] - peso_mean) * (df_merge['calorie'][i] - calorie_mean)
r = r / ((n - 1) * s_peso * s_calorie)

if r > 0:
    print(f'Coefficiente di correlazione campionario: {r}, è presente una correlazione positiva')
elif r < 0:
    print(f'Coefficiente di correlazione campionario: {r}, è presente una correlazione negativa')
else:
    print(f'Coefficiente di correlazione campionario: {r}, è presente una correlazione nulla')
```

```
Coefficiente di correlazione campionario: -0.07087210023488409, è presente una correlazione negativa
```

Diagramma a dispersione (scatter plot)

In [311…

```
fig, ax = plt.subplots()
ax.scatter(df_merge['peso'], df_merge['calorie'])
ax.set_xlabel('Peso')
ax.set_ylabel('Calorie assunte')

plt.show()
```



Quello che emerge dalla correlazione negativa tra il peso e le calorie è che quando il peso era alto, vi era un minore consumo di calorie. Al diminuire del peso, le calorie assunte sono aumentate. In effetti nella realtà questo accadde. Tuttavia, questo non è da interpretare come un evento negativo, poiché l'aumento del consumo delle calorie fu dato dalla maggiore attività fisica.

Con i dati da Samsung Health si potrebbero fare molti altri studi, come ad esempio quanto ha influito l'attività fisica sul peso, come è cambiato il battito cardiaco e la pressione sanguigna (la pressione scese di molto), ecc. Con l'incrocio dei dati si potrebbero fare tante cose interessanti.