03 Exercise Notebook 3

March 23, 2023

1 Exercise 3

In this exercise, you will analyse a dataset obtained from the London transport system (TfL). The data is in a filled called `tfl_readership.csv` (comma-separated-values format). As in Exercise 2, we will load and view the data using `pandas`.

[ ]:
```
# If you are running this on Google Colab, uncomment and run the following␣
 ↪lines; otherwise ignore this cell
# from google.colab import drive
# drive.mount('/content/drive')
```

[ ]:
```
import math
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

[ ]:
```
# Load data
df_tfl = pd.read_csv('tfl_ridership.csv')
# If running on Google Colab change path to '/content/drive/MyDrive/
 ↪IB-Data-Science/Exercises/tfl_ridership.csv'

df_tfl.head(13)
```

[ ]:

	Year	Period	Start	End	Days	Bus cash (000s)	\
0	2000/01	P 01	01 Apr '00	29 Apr '00	29d	884	
1	2000/01	P 02	30 Apr '00	27 May '00	28d	949	
2	2000/01	P 03	28 May '00	24 Jun '00	28d	945	
3	2000/01	P 04	25 Jun '00	22 Jul '00	28d	981	
4	2000/01	P 05	23 Jul '00	19 Aug '00	28d	958	
5	2000/01	P 06	20 Aug '00	16 Sep '00	28d	984	
6	2000/01	P 07	17 Sep '00	14 Oct '00	28d	1001	
7	2000/01	P 08	15 Oct '00	11 Nov '00	28d	979	
8	2000/01	P 09	12 Nov '00	09 Dec '00	28d	971	
9	2000/01	P 10	10 Dec '00	06 Jan '01	28d	912	
10	2000/01	P 11	07 Jan '01	03 Feb '01	28d	943	
11	2000/01	P 12	04 Feb '01	03 Mar '01	28d	975	
12	2000/01	P 13	04 Mar '01	31 Mar '01	28d	974	

	Bus Oyster PAYG (000s)	Bus Contactless (000s)	\
0	0	0	
1	0	0	
2	0	0	
3	0	0	
4	0	0	
5	0	0	
6	0	0	
7	0	0	
8	0	0	
9	0	0	
10	0	0	
11	0	0	
12	0	0	

	Bus One Day Bus Pass (000s)	Bus Day Travelcard (000s)	…	\
0	210	231	…	
1	214	205	…	
2	209	221	…	
3	216	241	…	
4	225	248	…	
5	243	236	…	
6	205	216	…	
7	199	221	…	
8	184	212	…	
9	192	211	…	
10	193	186	…	
11	194	210	…	
12	186	204	…	

	Tube Contactless (000s)	Tube Day Travelcard (000s)	\
0	0	655	
1	0	605	
2	0	650	
3	0	708	
4	0	730	
5	0	702	
6	0	639	
7	0	668	
8	0	640	
9	0	631	
10	0	556	
11	0	617	
12	0	584	

	Tube Season Travelcard (000s)	Tube Other incl free (000s)	\
0	1066	200	

```
1                            1168                            217
2                            1154                            212
3                            1196                            214
4                            1165                            165
5                            1164                            151
6                            1286                            196
7                            1298                            220
8                            1302                            242
9                             993                            195
10                           1259                            234
11                           1237                            246
12                           1262                            266

    Tube Total (000s)  TfL Rail (000s)  Overground (000s)  DLR (000s)  \
0                2509                0                  0          96
1                2598                0                  0          93
2                2623                0                  0          98
3                2761                0                  0         105
4                2643                0                  0         103
5                2608                0                  0         100
6                2763                0                  0         107
7                2819                0                  0         113
8                2839                0                  0         114
9                2359                0                  0          90
10               2634                0                  0         110
11               2688                0                  0         120
12               2699                0                  0         119

    Tram (000s)  Air Line (000s)
0          45.8              0.0
1          46.5              0.0
2          47.1              0.0
3          50.8              0.0
4          50.3              0.0
5          49.2              0.0
6          48.8              0.0
7          51.5              0.0
8          54.0              0.0
9          55.3              0.0
10         50.1              0.0
11         50.5              0.0
12         47.7              0.0

[13 rows x 26 columns]
```

Each row of our data frame represents the average daily ridership over a 28/29 day period for various types of transport and tickets (bus, tube etc.). We have used the `.head()` command to display the top 13 rows of the data frame (corresponding to one year). Focusing on the "Tube

Total" column, notice the dip in ridership in row 9 (presumably due to Christmas/New Year's), and also the slight dip during the summer (rows 4,5).

[ ]:
```
#df_tfl.sample(3)  #random sample of 3 rows
df_tfl.tail(3)  #last 3 rows
```

[ ]:
```
        Year Period       Start         End Days  Bus cash (000s)  \
242  2018/19   P 09  11 Nov '18  08 Dec '18  28d                0
243  2018/19   P 10  09 Dec '18  05 Jan '19  28d                0
244  2018/19   P 11  06 Jan '19  02 Feb '19  28d                0

     Bus Oyster PAYG (000s)  Bus Contactless (000s)  \
242                    1110                    1089
243                    1001                     949
244                    1036                    1075

     Bus One Day Bus Pass (000s)  Bus Day Travelcard (000s)  …  \
242                            0                         41  …
243                            0                         38  …
244                            0                         30  …

     Tube Contactless (000s)  Tube Day Travelcard (000s)  \
242                     1399                         249
243                     1110                         242
244                     1310                         204

     Tube Season Travelcard (000s)  Tube Other incl free (000s)  \
242                           1017                           334
243                            632                           259
244                            924                           305

     Tube Total (000s)  TfL Rail (000s)  Overground (000s)  DLR (000s)  \
242               4221               996                557         355
243               3279               750                414         270
244               3809               929                517         333

     Tram (000s)  Air Line (000s)
242         84.1              2.6
243         66.3              3.2
244         79.3              2.3

[3 rows x 26 columns]
```

The dataframe contains $N = 245$ counting periods (of 28/29 days each) from 1 April 2000 to 2 Feb 2019. We now define a numpy array consisting of the values in the ' Tube Total (000s)' column:

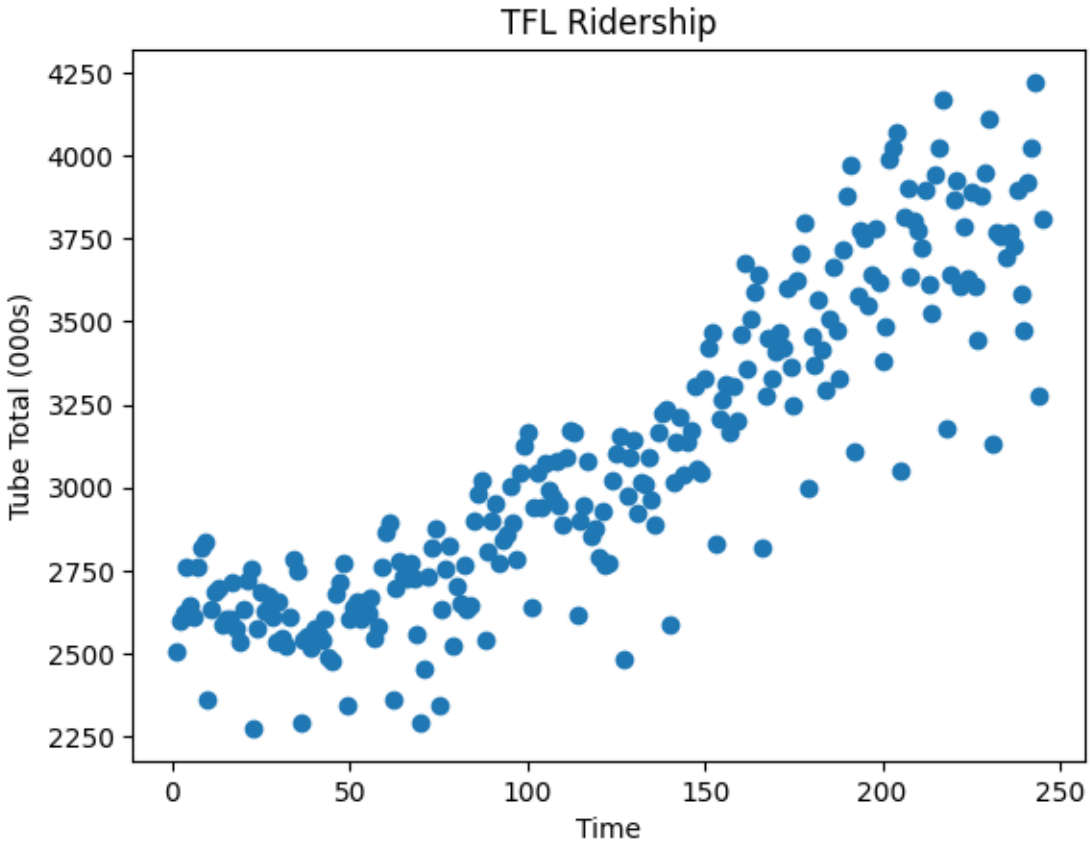
[ ]:
```
yvals = np.array(df_tfl['Tube Total (000s)'])
N = np.size(yvals)
```

```
xvals = np.linspace(1,N,N) #an array containing the values 1,2....,N
```

We now have a time series consisting of points (x_i, y_i), for $i = 1, \ldots, N$, where y_i is the average daily tube rideship in counting period $x_i = i$.

1.1 2a) Plot the data in a scatterplot

[ ]:
```
#Your code for scatterplot here
plt.scatter(xvals,yvals)
plt.xlabel('Time')
plt.ylabel('Tube Total (000s)')
plt.title('TFL Ridership')
plt.savefig('Exercise 2 tfl_ridership_scatter data.png')
plt.show()
```



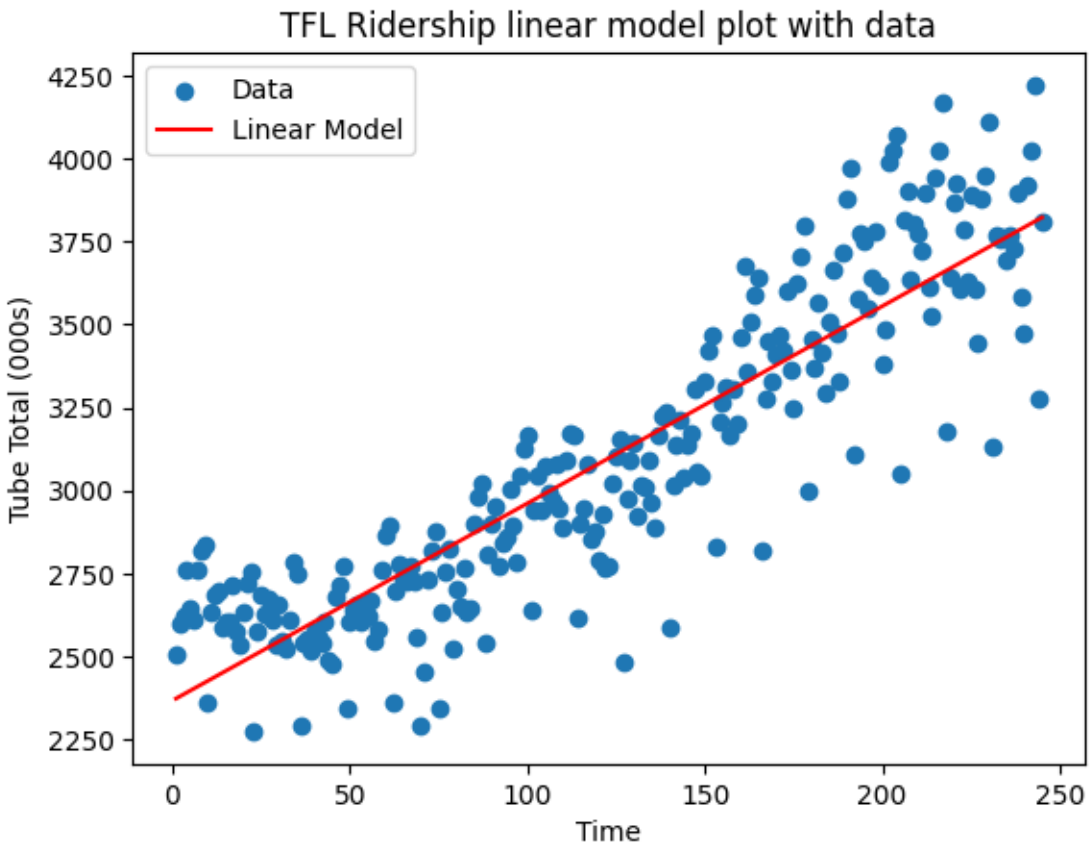
1.2 2b) Fit a linear model $f(x) = \beta_0 + \beta_1 x$ to the data

- Print the values of the regression coefficients β_0, β_1 determined using least-squares.
- Plot the fitted model and the scatterplot on the same plot.
- Compute and print the **MSE** and the R^2 coefficient for the fitted model.

All numerical outputs should be displayed to three decimal places.

[ ]:
```
#Your code here
X = np.column_stack((np.ones(N),xvals))
XT = X.T #transpose of X
beta_linear = np.linalg.inv(XT.dot(X)).dot(XT.dot(yvals)) #linear regression␣
 ↪coefficients using least squares
print(f'b0 = {beta_linear[0]:.2f}, b1 = {beta_linear[1]:.2f}')

linear_vals = beta_linear[0] + beta_linear[1]*xvals
plt.scatter(xvals,yvals, label='Data')
plt.plot()
plt.plot(xvals,linear_vals,'r', label='Linear Model')
plt.legend()
plt.xlabel('Time')
plt.ylabel('Tube Total (000s)')
plt.title('TFL Ridership linear model plot with data')
plt.savefig('Exercise 2b linear model plot with data.png')
plt.show()

SSE_linear = np.sum((yvals - linear_vals)**2)
MSE_linear = SSE_linear/(N)
R2_linear = 1 - SSE_linear/np.sum((yvals - np.mean(yvals))**2)
print(f'The MSE is {MSE_linear:.3f}, and the R^2 is {R2_linear:.3f}')
```

```
b0 = 2367.38, b1 = 5.94
```



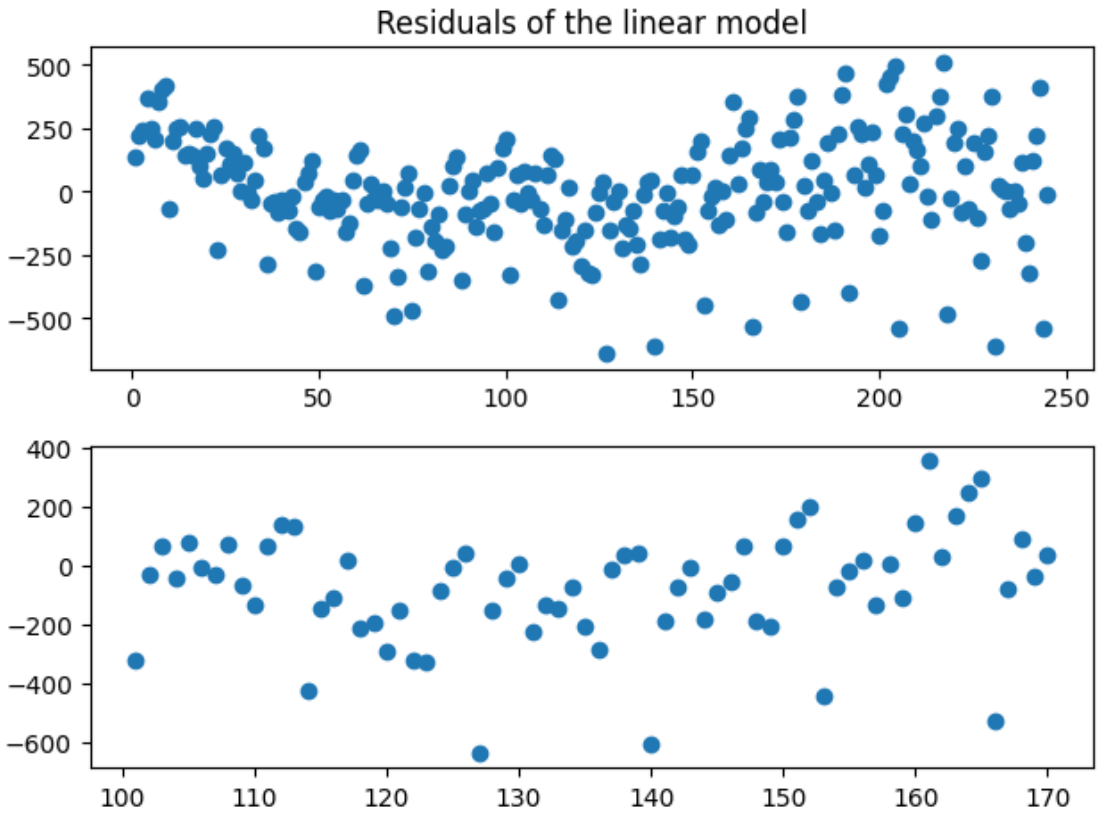

```
The MSE is 45323.636, and the R^2 is 0.796
```

1.3 2c) Plotting the residuals

- Plot the residuals on a scatterplot
- Also plot the residuals over a short duration and comment on whether you can discern any periodic components.

[ ]:
```
# Your code here
linear_residuals = yvals - linear_vals
plt.xlabel('Time')
plt.ylabel('Tube Total (000s)')

plt.subplot(211)
plt.scatter(xvals, linear_residuals)
plt.title('Residuals of the linear model')
plt.subplot(212)
plt.scatter(xvals[100:170], linear_residuals[100:170])
plt.savefig('Exercise 2c Residuals of linear plot.png')
plt.tight_layout()
```

```
plt.show()
```



< Comment on periodic components here >

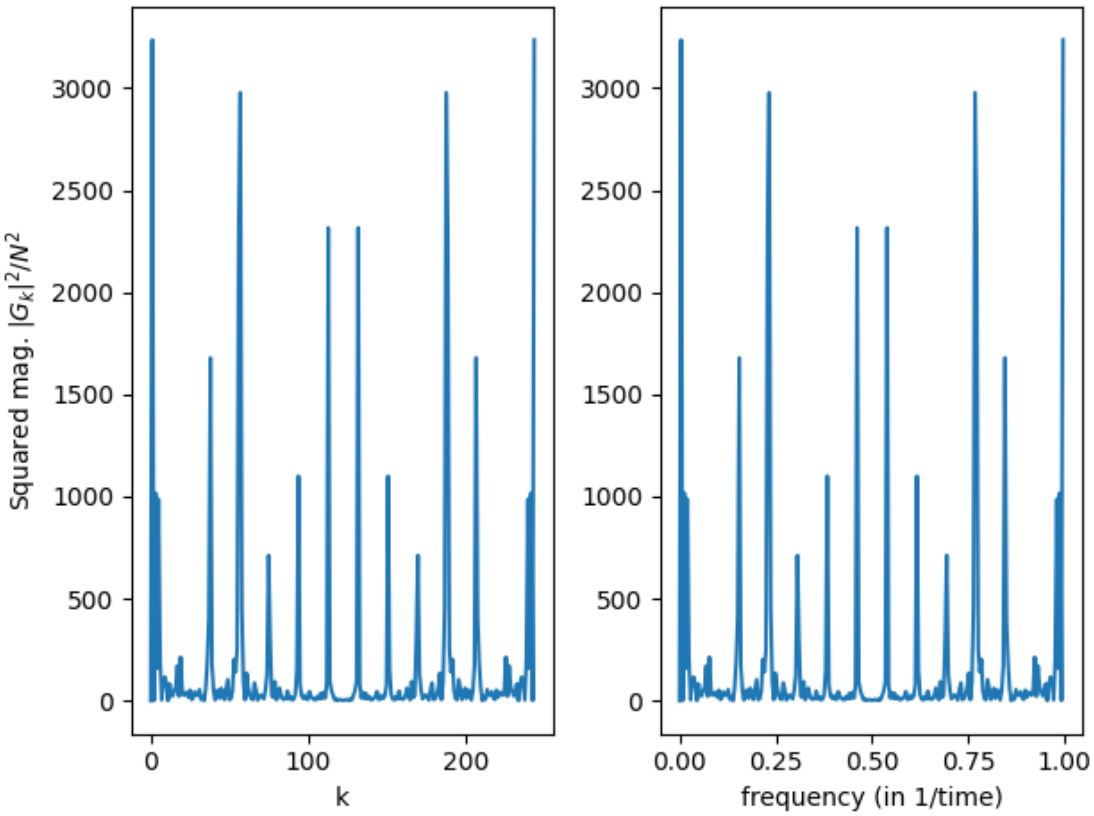
There seems to be a periodic nature every 12 x axis points, indicating a yearly residual nature

2d) Periodogram

- Compute and plot the peridogram of the residuals. (Recall that the periodogram is the squared-magnitude of the DFT coefficients.)
- Identify the indices/frequencies for which the periogram value exceeds **50%** of the maximum.

[ ]:
```
# Your code to compute and plot the histogram
T = xvals[1] - xvals[0]  # This can be the time interval between any two␣
 ↪successive values.

# Compute the squared magnitudes of the DFT coefficients -- this is known as␣
 ↪the "periodogram"
pgram = np.abs(np.fft.fft(linear_residuals, N)/N)**2 #We normalize by N, but␣
 ↪this is optional
indices = np.linspace(0, (N-1), num = N)
freqs_in_hz = indices/(N*T)
```

```
freqs_in_rads = freqs_in_hz*2*math.pi

plt.subplot(121)
plt.plot(indices, pgram)
plt.xlabel('k')
plt.ylabel('Squared mag. $|G_k|^2/N^2$')
plt.subplot(122)
plt.plot(freqs_in_hz, pgram)
plt.xlabel('frequency (in 1/time)')  # Since units of T is time
plt.savefig('exercise 2d periodogram of residuals.png')
plt.tight_layout()
plt.show()
```



[ ]:
```
# Your code to identify the indices for which the periodogram value exceeds 50%␣
 ↪of the maximum
def find_indicies(pgram):
    top_inds = indices[(pgram > 0.5*np.max(pgram))]
    return top_inds[:len(top_inds)//2]

large_indicies = find_indicies(pgram)
```

```
large_freqs_in_rads = large_indicies/(N*T) *2*math.pi
print(f'The frequencies in rads are {large_freqs_in_rads}')
```

```
The frequencies in rads are [0.02564565 0.97453486 1.43615664 1.4618023
2.89795894]
```

1.4 2e) To the residuals, fit a model of the form

$$\beta_{1s}\sin(\omega_1 x) + \beta_{1c}\cos(\omega_1 x) + \beta_{2s}\sin(\omega_2 x) + \beta_{2c}\cos(\omega_2 x) + \ldots + \beta_{Ks}\sin(\omega_K x) + \beta_{Kc}\cos(\omega_K x).$$

The frequencies $\omega_1, \ldots, \omega_K$ in the model are those corresponding to the indices identified in Part 2c. (Hint: Each of the sines and cosines will correspond to one column in your X-matrix.)

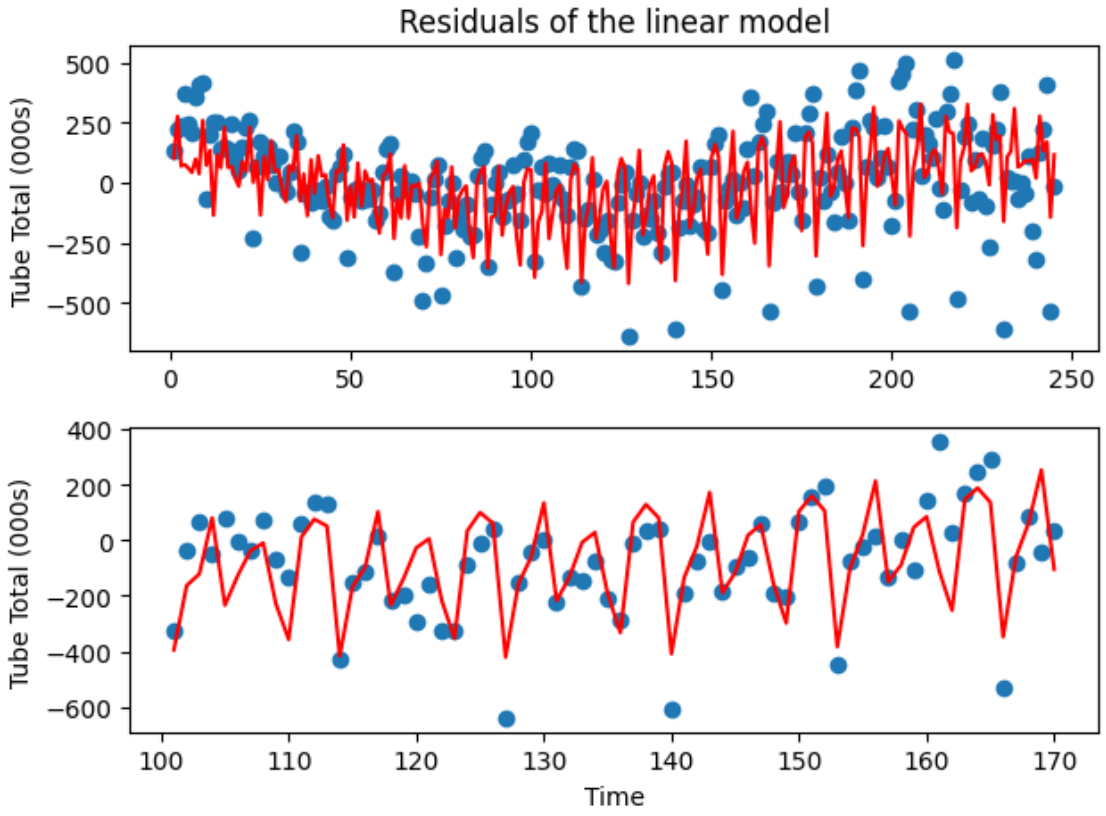
- Print the values of the regression coefficients obtained using least-squares.
- Compute and print the final **MSE** and R^2 coefficient. Comment on the improvement over the linear fit.

All numerical outputs should be displayed to three decimal places.

[ ]:
```
# Your code here

XT = np.vstack([[np.sin(w*xvals), np.cos(w*xvals)] for w in␣
 ↪large_freqs_in_rads])
X = XT.T

beta_sins = np.linalg.inv(XT.dot(X)).dot(XT).dot(linear_residuals) # we want to␣
 ↪fit the sinusoids to the residuals from the linear fit
print(f'Beta values are {beta_sins.round(3)}')

sin_values = X.dot(beta_sins)



plt.subplot(211)
plt.ylabel('Tube Total (000s)')
plt.scatter(xvals, linear_residuals)
plt.plot(xvals, sin_values, 'r')
plt.title('Residuals of the linear model')
plt.subplot(212)
plt.xlabel('Time')
plt.ylabel('Tube Total (000s)')
plt.scatter(xvals[100:170], linear_residuals[100:170])
plt.plot(xvals[100:170], sin_values[100:170], 'r')
plt.savefig('Exercise 2e sinusodial model to fit linear residuals.png')
plt.tight_layout()
plt.show()

SSE_sin = np.sum((linear_residuals - sin_values)**2)
MSE_sin = SSE_sin/(N)
```

```
R2_sin = 1 - SSE_sin/np.sum((linear_residuals - np.mean(linear_residuals))**2)
print(f'The MSE is {MSE_sin:.3f}, and the R^2 is {R2_sin:.3f}')
```

```
Beta values are [-51.253 101.556  61.628 -54.006 -15.581 -94.797  81.659  72.381
32.472
  90.589]
```




```
The MSE is 20297.501, and the R^2 is 0.552
```

1.5 2f) The combined fit

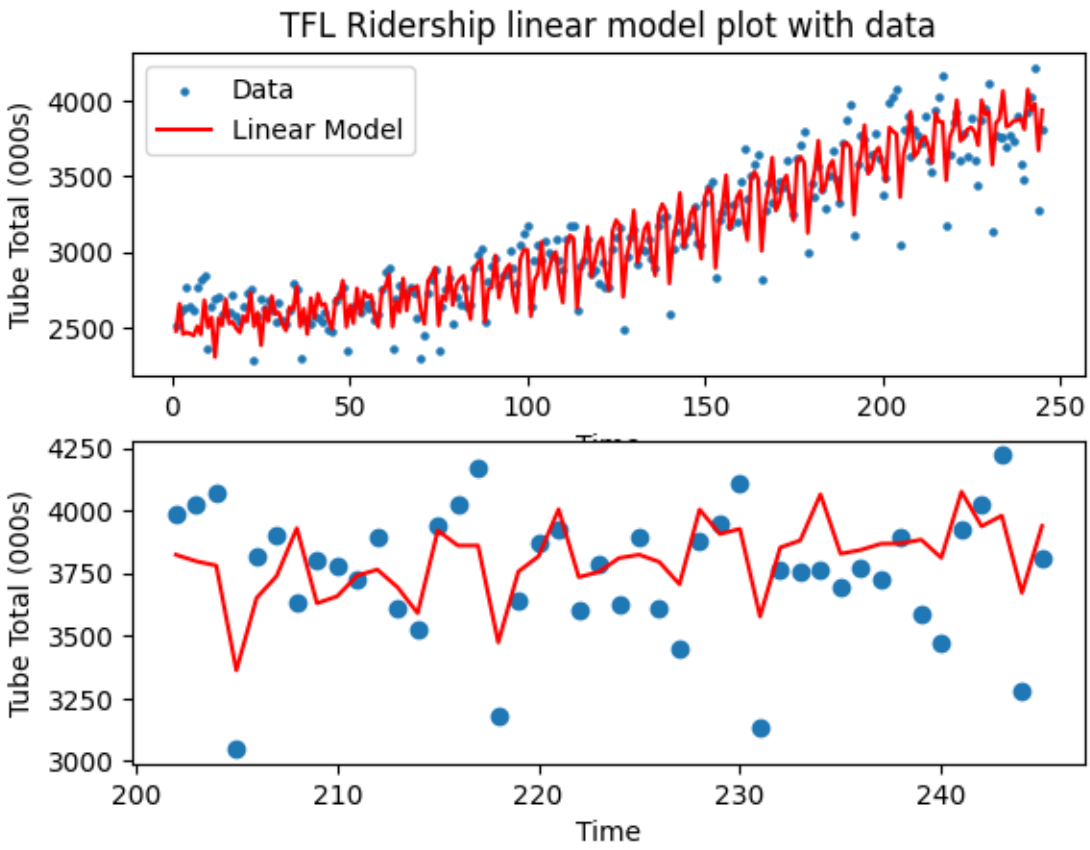
- Plot the combined fit together with a scatterplot of the data
- Compute and print the final **MSE** and R^2 coefficient. Comment on the improvement over the linear fit.

The combined fit, which corresponds to the full model

$$f(x) = \beta_0 + \beta_1 x + \beta_{s1}\sin(\omega_1 x) + \beta_{c1}\cos(\omega_1 x) + ... + \beta_{sk}\sin(\omega_k x) + \beta_{ck}\cos(\omega_k x),$$

can be obtained by adding the fits in parts 2b) and 2e).

[ ]:
```
# Your code here
combined_vals = linear_vals + sin_values
plt.subplot(211)
plt.scatter(xvals,yvals, label='Data',s=5)
plt.plot(xvals,combined_vals,'r', label='Linear Model')
plt.legend()
plt.xlabel('Time')
plt.ylabel('Tube Total (000s)')
plt.title('TFL Ridership linear model plot with data')

plt.subplot(212)
plt.xlabel('Time')
plt.ylabel('Tube Total (000s)')
plt.scatter(xvals[201:250], yvals[201:250])
plt.plot(xvals[201:250], combined_vals[201:250], 'r')
plt.savefig('Exercise 2e linear+sin model plot with data.png')
plt.show()

SSE_combined = np.sum((yvals - combined_vals)**2)
MSE_combined = SSE_combined/(N)
R2_combined = 1 - SSE_combined/np.sum((yvals - np.mean(yvals))**2)
print(f'The MSE of the combined fit is {MSE_combined:.3f}, and the R^2 is␣
 ↪{R2_combined:.3f}')
print(f'The MSE of the linear fit is {MSE_linear:.3f}, and the R^2 is␣
 ↪{R2_linear:.3f}')
print(f'Therefore the combined fit MSE is {(MSE_linear - MSE_combined)/
 ↪MSE_linear*100:.3f}% better than the linear fit')
print(f'The combined fit R^2 is {(R2_combined - R2_linear)/R2_linear*100:.3f}%␣
 ↪better than the linear fit')
```




```
The MSE of the combined fit is 20297.501, and the R^2 is 0.908
The MSE of the linear fit is 45323.636, and the R^2 is 0.796
Therefore the combined fit MSE is 55.217% better than the linear fit
The combined fit R^2 is 14.185% better than the linear fit
```