COLMAN_201532472

February 22, 2024

1 Data handling assessment

```
[212]: from DataGen import *   # generates your data
       import matplotlib.pyplot as plt ## this has imported matplotlib's pyplot as plt
       plt.rcParams.update({'font.size': 16, 'figure.figsize': [10, 7]}) # sets better␣
        ↪defaults for matplotlib graphs

       import numpy as np  ## this has imported numpy which you can refer to as np
       import scipy.stats as stats ## this has importated SciPy's statistics package␣
        ↪as stats, you may find this useful.
       import scipy.signal as signal ## imports the signal processing module as signal
       import scipy.optimize as fit  ## imports the optimization tool box as fit

       import glob  # imports the glob module
       import warnings  # prevents printing of some warnings
       warnings.filterwarnings('ignore')
```

1.0.1 Each student has a unique data set to process.

1) To initailise your unique data set insert your 9 digit student number below as a string i.e. inside the inverted commas and run the cell.

The length of ID should be 9

```
[213]: ID = '201532472' #
       print(len(ID))
```

```
9
```

1.0.2 1) Loading files

- It is useful to be able to load multiple data sets from a folder.

  1) Run the cell below to initialize your data sets, they will appear in the 'Data sets' folder where you launched this notebook from.

```
[214]: dataFolder(ID) ## this creates your data files to process
```

- The data that you want to load is contained in .csv files

- Each file contains a single number

  2) Make a sorted list of these csv file names.

  3) Then loop through this list and load each csv file.

  4) On each iteration of the loop print the data you have loaded.

You should be able to do this in 5 lines or less

[215]:
```python
files = sorted(glob.glob('Data sets/*.csv')) # Sorts the files
for file in files:
    data=np.genfromtxt(file) # Loops through the sorted list and loads each file
    print(file,data) # Prints each file name (column 1) and the contents of the␣
 ↪file (column 2)
```

```
Data sets/Data_A.csv 2.0
Data sets/Data_B.csv 7.0
Data sets/Data_C.csv 4.0
Data sets/Data_D.csv 2.0
Data sets/Data_E.csv 3.0
Data sets/Data_F.csv 5.0
Data sets/Data_G.csv 1.0
Data sets/Data_H.csv 0.0
Data sets/Data_I.csv 2.0
```

1.0.3 2) An experimenter has made an extracellular recording from a motor nerve and wants to determine the change in the mean firing rate of this nerve during a voluntary contraction.

- The sample rate of the recording was 10kHz
- The data is recorded in µV
- The recording is 15 seconds in length
- The voluntary contraction began at 5 seconds and lasted 5 seconds.

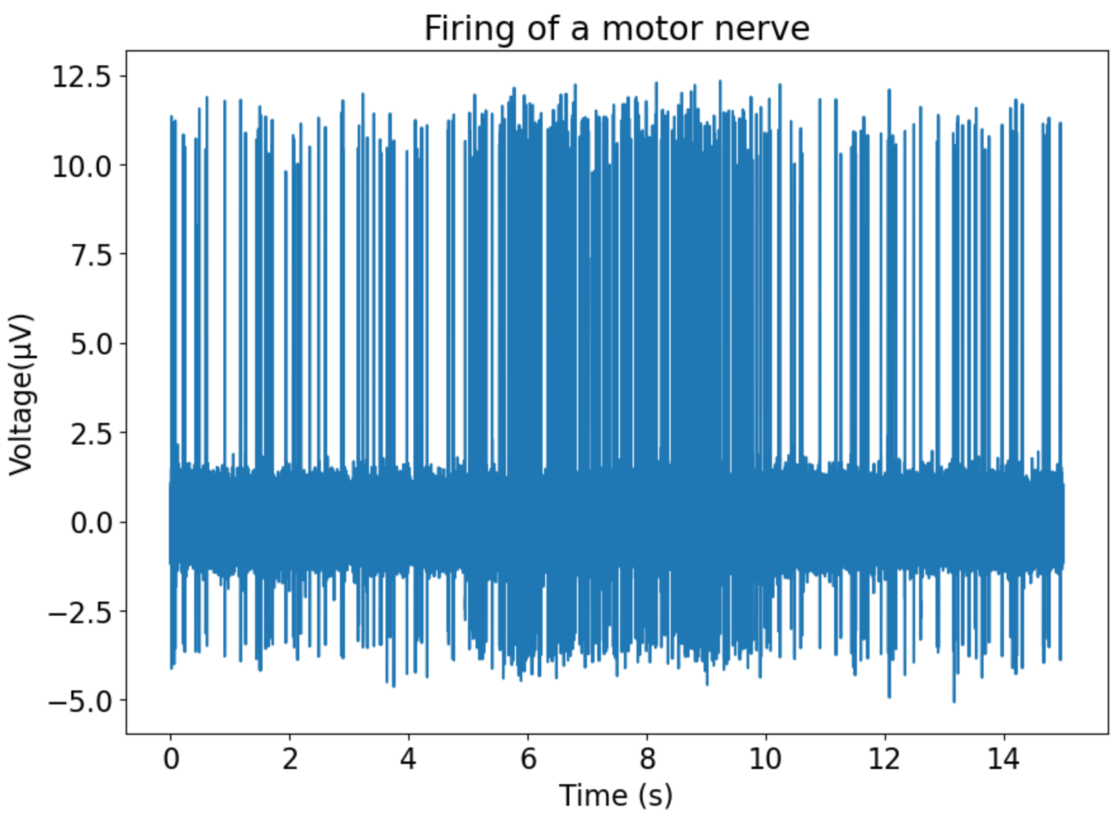
  1) Run the cell below to generate your nerve data as an array

[216]:
```python
nerve = getSpike(ID) # this loads your data for this task in to the numpy array␣
 ↪called 'nerve'
```

  2) In the cell below plot the nerve recording with the correct time scale.

[217]:
```python
sampleRate=10000 # Equivalent to 10kHz
xScale=np.arange(0,nerve.shape[0]/sampleRate,1/sampleRate);
plt.plot(xScale,nerve)
plt.xlabel('Time (s)')
plt.ylabel('Voltage(µV)')
plt.title('Firing of a motor nerve')
```

[217]:
```
Text(0.5, 1.0, 'Firing of a motor nerve')
```



3) Detect spikes in the recording and then:

A) Detemrmine the the mean spike rate in Hz during the contraction.

B) Determine the change in mean spike rate evoked by the contraction.

[218]:
```
# 3)
peaks, props=signal.find_peaks(nerve,height=5.0)
# Height set to 5.0 to detect only largest spikes
print('3)')
print('There were',len(peaks),'spikes in the recording.')


# A)
contTime=np.column_stack((xScale,nerve)) # Creates two-column list
volCont = contTime[50000:100000,:] # Slices time 5-10s from list
peaks,props=signal.find_peaks(volCont[:,1],height=5.0)
spikes=len(peaks) # 'spikes' now contains the number of spikes that occurred␣
 ↪during the voluntary contraction.
rate=(spikes*1.0)/5.0 # 'rate' contains the calculated rate (Hz) of spikes.
#print(rate)
```

```
print(len(peaks),'spikes occurred during the voluntary contraction.')


# B)
nerveBef = contTime[0:49999,:]
nerveAft = contTime[100001:150000,:]
# Follows the same process as above but slices the list at 0-4.9s and 10.1-15s.

peaks,props=signal.find_peaks(nerveBef[:,1],height=5.0)
spikes=len(peaks)
rateBef=(spikes*1.0)/5.0
#print(rateBef)

peaks,props=signal.find_peaks(nerveAft[:,1],height=5.0)
spikes=len(peaks)
rateAft=(spikes*1.0)/5.0
#print(rateAft)
# 'rateBef' and 'rateAft' now contain the calculated rates of spikes before and␣
 ↪after the voluntary contraction.
```

```
3)
There were 322 spikes in the recording.
210 spikes occurred during the voluntary contraction.
```

4) Print some text reporting the answers to 3A & 3B.

[219]:
```
print('A)')
print('The mean spike rate during the voluntary contraction was',rate,'Hz.')
print('\n') # I printed line spaces to make some answers more readable
print('B)')
print('The mean spike rate before the voluntary contraction was',rateBef,'Hz.')
print('The mean spike rate after the voluntary contraction was',rateAft, 'Hz.')
print('\n')
print('The spike rate increased by',rate-rateBef, 'Hz due to the contraction.')
```

```
A)
The mean spike rate during the voluntary contraction was 42.0 Hz.


B)
The mean spike rate before the voluntary contraction was 11.2 Hz.
The mean spike rate after the voluntary contraction was 11.2 Hz.


The spike rate increased by 30.8 Hz due to the contraction.
```

1.0.4 3) An experimenter wants to determine whether a new drug is effective at increasing platelet count in patients with thrombocytopenia (low platelet count).

- A platelet count below 150,000 platelets µl-1 indicates a clinical risk.
- The experimenter has administered the new drug to 900 thrombocytopenia patients and measured platelet counts before and after treatment with the drug.

1) Run the cell below to load your platelet count data

```
[220]: platelets = getPlatelets(ID) # this loads your data for this task in to the␣
        ↪numpy array called 'mRNAs'
```

- The 1st column of "platelets" is the pre-treatment counts and the 2nd contains the data for the same patients post-treatment.
- Each row corresponds to the platelet count per µl for each patient.

2) Plot a single graph that clearly displays both histograms of the platelet counts. Use 40 bins for each histogram.

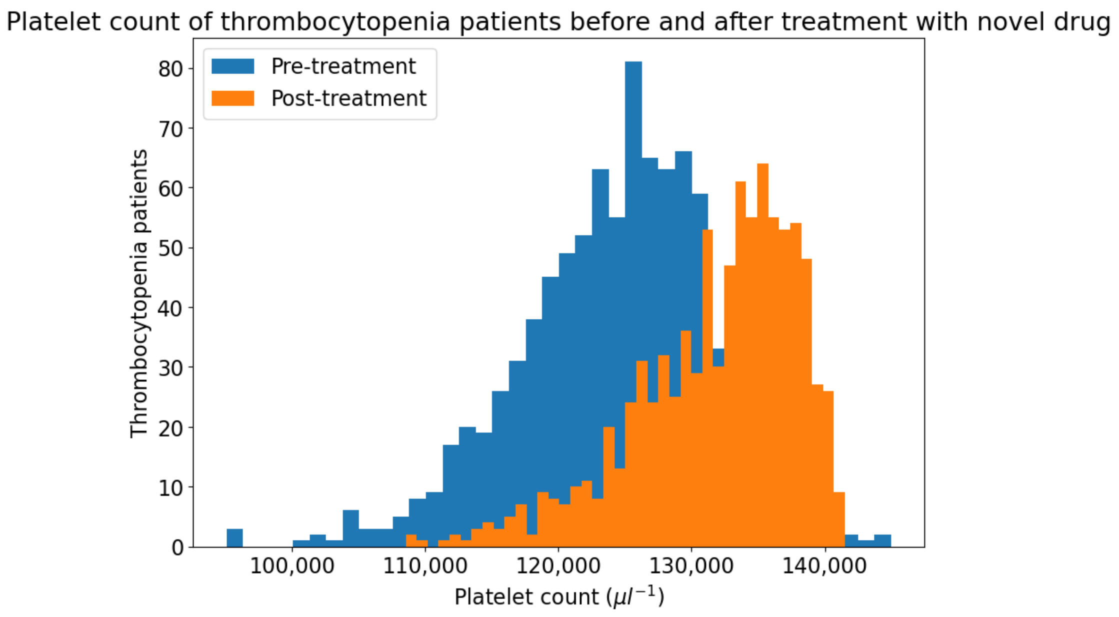
```
[221]: plt.hist(platelets[0], bins=40,label='Pre-treatment')
       plt.hist(platelets[1], bins=40,label='Post-treatment')
       # Plots and labels the platelet data

       plt.legend(loc='upper left')
       plt.xlabel('Platelet count $(µl^{-1})$')
       plt.ylabel('Thrombocytopenia patients')
       # Legend position and axes labels

       current_values = plt.gca().get_xticks()
       plt.gca().set_xticklabels(['{:,.0f}'.format(x) for x in current_values])
       # Formats the numbers on the x axis with a comma, making them more readable.

       plt.title('Platelet count of thrombocytopenia patients before and after␣
        ↪treatment with novel drug')
       #plt.legend((platelets[0], platelets[1]), ('Pre-treatment', 'Post-treatment'),␣
        ↪loc='upper left')
       #plt.legend((platelets[1]),'Post-treatment')

       plt.show()
```



3) Determine whether the drug has a significant effect. Print some text justifying your conclusion.

[222]:
```
# Mean of platelets 0 and 1
meanPre=np.mean(platelets[0])
meanPost=np.mean(platelets[1])
meanPre=round(meanPre,1)
meanPost=round(meanPost,1) # Rounds the mean values to one decimal place
#print(meanPre,meanPost)

# Stdev of platelets 0 and 1
stdPre=np.std(platelets[0])
stdPost=np.mean(platelets[1])
#print(stdPre,stdPost)

print('The mean platelet count was',meanPre,'before treatment␣
 ↪and',meanPost,'after treatment.')
print('\n')

t_stat,p_value=stats.ttest_ind(platelets[0],platelets[1])
# Performs T-test to test for significance between data

print("T:",t_stat)
print("P:",p_value)
print('\n')
sig=0.05 # Sets significance to 0.05
```

```
if p_value<sig:
    print("The drug had a significant effect on platelet count.")
else:
    print("The drug did not have a significant effect on platelet count.")
# Prints the appropriate statement depending on the result of the T-test.
```

```
The mean platelet count was 124006.4 before treatment and 131890.1 after
treatment.


T: -25.328811173022373
P: 2.682566954789918e-121


The drug had a significant effect on platelet count.
```

4) Print some text describing whether the new drug is effective at increasing platelet counts to healthy levels?

[223]:
```
print('The T-test found a significant difference between platelet counts before␣
 ↪and after treatment with the new drug.')
print('\n')
print('Because the drug significantly increased platelet count, it is effective␣
 ↪in treating thrombocytopenia.')
print('\n')
print('However, the platelet counts were still lower than 150,000 µl^-1 after␣
 ↪treatment, indicating the need for a more potent drug or a change to the␣
 ↪dosage regimen.')
```

```
The T-test found a significant difference between platelet counts before and
after treatment with the new drug.


Because the drug significantly increased platelet count, it is effective in
treating thrombocytopenia.


However, the platelet counts were still lower than 150,000 µl^-1 after
treatment, indicating the need for a more potent drug or a change to the dosage
regimen.
```

1.0.5 4) An experimenter has recorded the time of wheel running events in an experiment using mice and wants to plot the total wheel turns per day.

- The experiemnt ran for 35 days
- The time stamps of full wheel turns were saved to a .csv file with 1 file for each day > 1) Run the cell below to generate your data which will appear in the 'Exercise data' folder

[224]:
```
getExercise(ID)
```

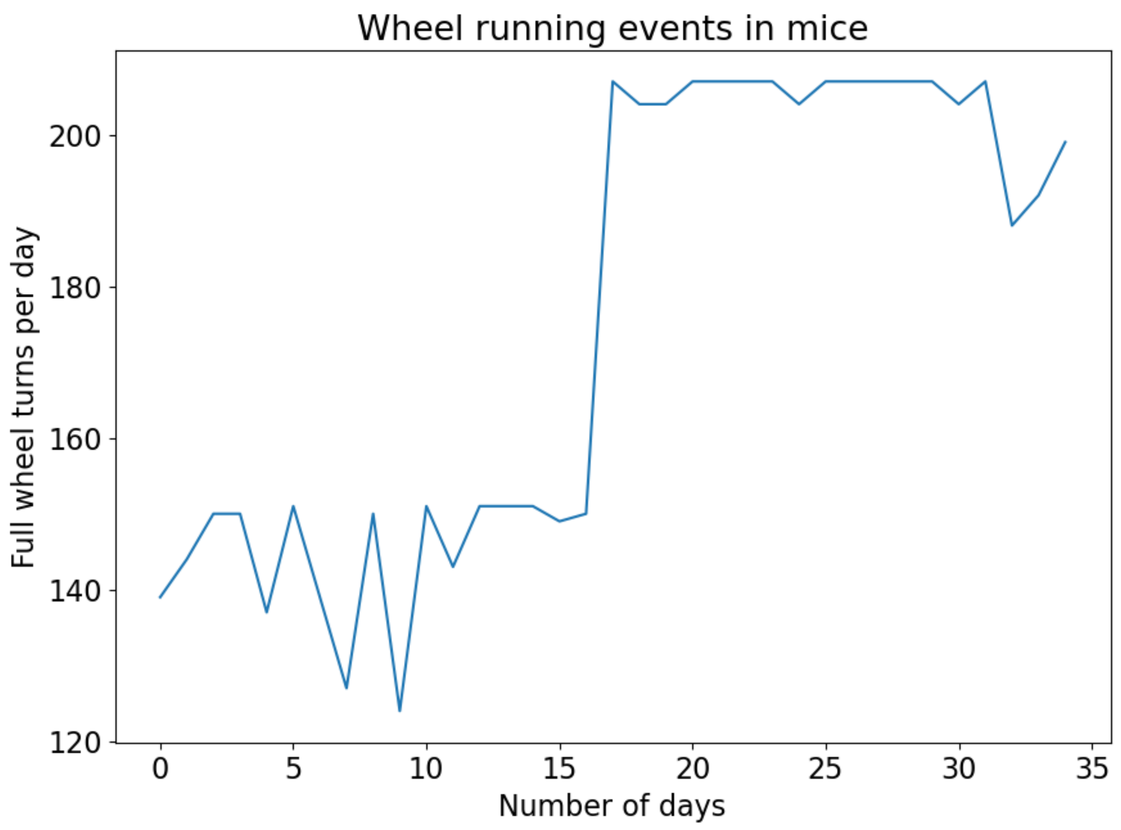
2) Load the data from the 'Exercise data' folder and generate a graph displaying the total daily wheel turns against day.

[225]:
```
turns=[] # Empty list to append the wheel turns data.

files=sorted(glob.glob('Exercise data/*.csv'))
for file in files:
    data=np.genfromtxt(file)
    turns.append(data.shape) # Counts the full wheel turns and appends the data␣
 ↪to the 'turns' list.

xScale=np.arange(0,35,1)
plt.plot(xScale,turns)
plt.xlabel('Number of days')
plt.ylabel('Full wheel turns per day')
plt.title('Wheel running events in mice')
```

[225]:
```
Text(0.5, 1.0, 'Wheel running events in mice')
```



• Methylphenidate, which causes increased locomotor activity in mice, was added to the dirnking water. Judging from your graph on what day was the durg added?

3) Print some text with your answer

[226]:
```
print('The number of wheel turns drastically increases on day 16, indicating␣
↪this is when the methylphenidate was added to the drinking water.')
```

```
The number of wheel turns drastically increases on day 16, indicating this is
when the methylphenidate was added to the drinking water.
```

1.0.6 5) An experimenter is measuring the drug concentration in the blood to estimate its elimation rate.

- Samples were taken every 5 minutes
- Each data point is the concentration of drug in µg l-1

1) Run the cell below to generate your drug measurements

[227]:
```
drug = getDrug(ID) # this loads your data for this task in to the numpy array␣
↪called 'drug'
```

2) Plot the data as a scatter plot i.e. points for markers rather than lines between points.

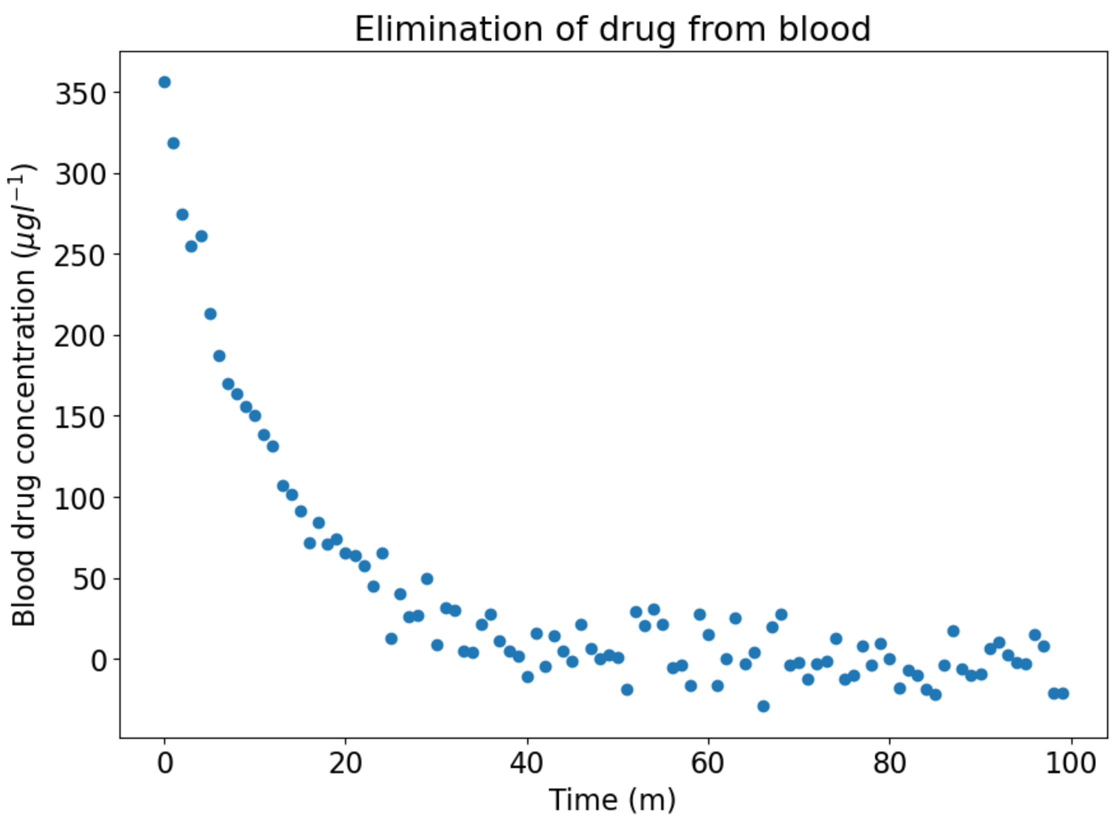
3) Define the appropriate equation for an exponential decay which describes the clearnce of a drug from the body.

4) Fit this equation to your data and plot the best fit.

5) Determine the rate constant of drug clearance and print this with some descriptive text.

[228]:
```
plt.plot(drug,'o')
plt.xlabel('Time (m)')
plt.ylabel('Blood drug concentration $(µg l^{-1})$')
plt.title('Elimination of drug from blood')
```

[228]:
```
Text(0.5, 1.0, 'Elimination of drug from blood')
```

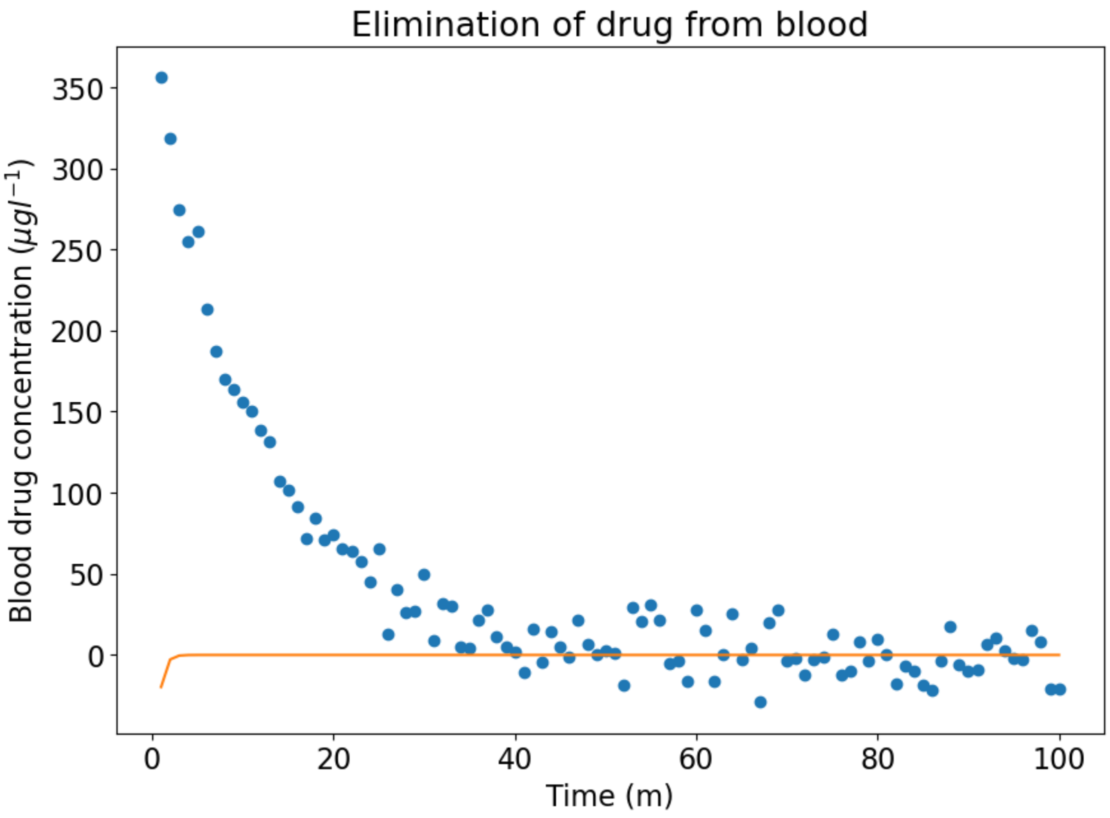


[229]:
```
x_data = np.arange(1,101,1)
#print(x_data)
y_data = drug
curve_fit=np.polyfit(x_data,y_data,1)
#print(curve_fit)

y=140.097*-np.exp(-1.96*x_data)
# Fitting constant and exponential term to the data

plt.plot(x_data, y_data, 'o')
plt.plot(x_data,y)
plt.xlabel('Time (m)')
plt.ylabel('Blood drug concentration $(µg l^{-1})$')
plt.title('Elimination of drug from blood')
# Unable to fit curve properly
```

[229]:
```
Text(0.5, 1.0, 'Elimination of drug from blood')
```



1.0.7 6) Summarising nerve conduction velocities.

An experimenter has measured the coduction velocity of sensory nerve fibres at 3 different sites corresponding to the different nerve fibers shown in the below table.

Nerve type	Conduction velocity (m/s)
C	0.5 - 2
A	80 -120
A	33 - 75

- The experimentor recorded from each site in 40 subjects and stored their data in the 'Fiber data' folder.
- Each of the 40 data files contain one measuremnet from each site.
- The sites were always recorded in the same order.

1) Run the cell below to initialize your conduction velocity data sets, they will appear in the 'Fibre data' folder where you launched this notebook from.

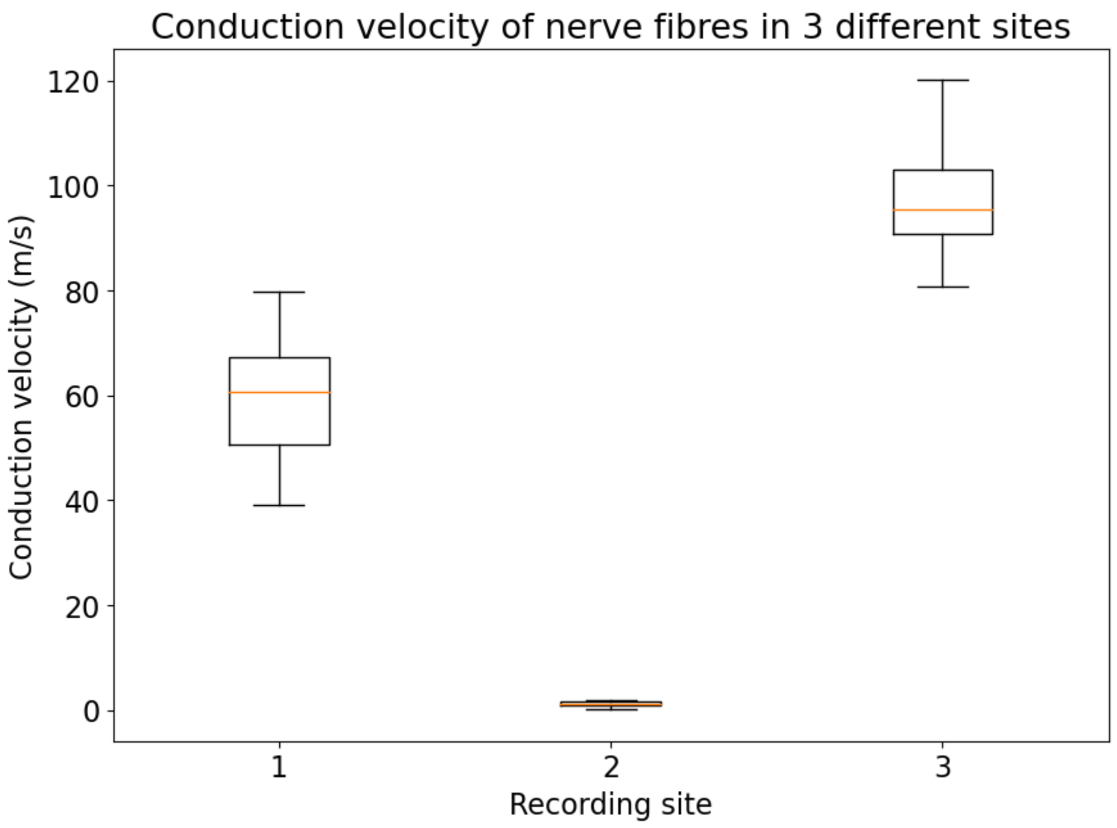
[230]:
```
getfibres(ID) # this places your conduction velocity data into the 'Fiber data'␣
  ↪folder
```

2) Load all the data and generate box plots to summarise the measured conduction velocities at each site.

[231]:
```
fibres1=[]
fibres2=[]
fibres3=[]
# Creates an array for each recording type

files=sorted(glob.glob('Fibre data/*.csv'))
for file in files:
    data=np.genfromtxt(file) # Sorts and opens the files
    #print(data[0])
    fibres1.append(data[0])
    fibres2.append(data[1])
    fibres3.append(data[2])
    # Appends each data type into the appropriate file

#dataSet = np.vstack((fibres1, fibres2,fibres3))
fig, ax = plt.subplots()
ax.boxplot((fibres1,fibres2,fibres3))
ax.set_xlabel('Recording site')
ax.set_ylabel('Conduction velocity (m/s)')
ax.set_title('Conduction velocity of nerve fibres in 3 different sites')
fig.show()
```



3) Print some text to indicate which recording site corresponds to which fibre type.

[234]:
```python
print('Series 1 recording site fires at around 60 m/s which corresponds to␣
 ↪A-beta fibre type.')
print('Series 2 recording site fires at around 2 m/s which corresponds to C␣
 ↪fibre type.')
print('Series 3 recording site fires at around 90 m/s which corresponds to␣
 ↪A-alpha fibre type.')
```

```
Series 1 recording site fires at around 60 m/s which corresponds to A  fibre
type.
Series 2 recording site fires at around 2 m/s which corresponds to C fibre type.
Series 3 recording site fires at around 90 m/s which corresponds to A  fibre
type.
```

1.0.8 7) Statement on use of generative AI

This is an amber assessment, use of generative AI in an assistive capacity is permitted.
You must acknowledge the use of any AI, see the university's guidelines on acknowledging AI use
If you have used generative AI as part of completing this assessment give a brief statement describing which tool you used and how you used it. e.g. "I used chatGPT 3.5 (OpenAI) to check the syntax for generating histograms"

[ ]:
```

```