B18 - Biomedical Modelling and Monitoring

Wearable Technology - Laboratory

Organiser(s): . . . . . Prof. Mauricio Villarroel
Instructor(s): . . . . . Dr. Mayue Shi, Dr. Xiao Gu

Department of Engineering Science
University of Oxford

Hillary term 2025 revision 2

Table of contents

> **Tip**
>
> It is recommended to follow the electronic/PDF version of this lab sheet to enable clickable embedded hyperlinks, depicted in blue.

1 Overview

The aim of this laboratory is to gain hands on experience with different biomedical signals that can be collected from wearable sensors, such as: electrocardiogram (ECG), electroencephalogram (EEG) and smartphone accelerometry. You will learn how to process and analyse sensor-based measurements from clinically recorded data. The goal of this laboratory is to understand the role of signal conditioning, how to choose suitable parameters of signal processing algorithms, and the various technological approaches biomedical engineers have to understand diseases from wearable sensor data.

The laboratory will be run in person in the Software Lab A at the Thom building on the following four days:

- Thursday Week 5: 20^{th} of February
- Friday Week 5: 21^{st} of February
- Thursday Week 6: 27^{th} of February
- Friday Week 6: 28^{th} of February

Each day, there will be two slots available: morning (10am - 1pm) and afternoon (2pm-5pm). The student should choose only one day and one session to attend. Use the register on Canvas to sign up to the session that best fit your availability.

1.1 Requirements

Before coming to the lab, make sure that you:

1. Install MATLAB in your laptop, download the data and code for the laboratory, and bring your laptop to the laboratory session with you. For this, please follow the instructions described on section 2.

2. Complete the preparatory work described on section 3. Write all your answers in your logbook and bring it to the lab with you. Have a demonstrator to sign up your work.

1.2 Submission and marking

At the end of your laboratory session, You are required to submit your work to the demonstrators by email as a digital file (for example a .doc, .pdf, or similar file). It is recommended you use Microsoft Word, Microsoft OneNote, LaTeX, or similar software to complete your laboratory sheet exercises.

Students are expected to complete all exercises and document results in their laboratory write-up, including any figures and tables generated. You should cite references where used in any of your answers. All figures should have their axis (and units) labelled where applicable. Figures and tables should be saved an inserted into documents, or can be copy and pasted into a document. Details on how to copy figures to clipboards can be found in the MATLAB documentation[1]. Please label all exercises clearly, failure to clearly document your answers and discussions may result in loss of marks. Students should submit their accompanying MATLAB scripts, which should be clearly labelled. It is recommended you make use of code sections to help with layouts and formatting, see MATLAB documentation[2].

The lab demonstrators will mark your work at the end of each laboratory session.

2 Getting started

2.1 Downloading data

This laboratory requires a MATLAB vR2019b (or later, for example vR2023b) installation on your machine (The MathWorks, Natick, MA, USA). To download the latest version of MATLAB, visit the "How to get software" webpage at `https://help.it.ox.ac.uk/sls/fulllist`. Follow the instructions to download and install MATLAB using the licence-key provided by the Department of Engineering Science. More details on how to install MATLAB can be found at `https://eng.ox.ac.uk/matlab/install`. Ensure you have installed the signal processing toolbox (`https://uk.mathworks.com/products/signal.html`).

2.2 Downloading data

Once MATLAB is installed, you must download the data and accompanying code for the laboratory. The laboratory materials, scripts and data can be downloaded from: `https://github.com/maurovm/b18_laboratory_code`. Download the lab files and unzip the content to your **own** MATLAB code directory (this is usually somewhere like: " `/user/code/B18/`"). Type `userpath` into MATLAB's command window to find this. Next, run the `startup.m` file to extract and unzip the data and files needed for the B18 laboratory. It is recommended that you kepp all data in the "`data/`" and all scripts

[1] `https://uk.mathworks.com/help/matlab/creating_plots/copy-figure-to-clipboard-from-edit-menu.html`

[2] `https://uk.mathworks.com/help/matlab/matlab_prog/run-sections-of-programs.html`

and code in the "`scripts/`" folders (Note that the `startup.m` script will automatically created these folders).

Remember to ensure that you use the correct `pathname` when loading your data, as this is the biggest troubleshooting problem students typically encounter. See the official MATLAB documentation[3] for further useful information.

3 Preparatory work

3.1 The Electrocardiogram

The electrocardiogram (ECG) records the electrical activity generated by the heart as it undergoes depolarisation and repolarisation of the atria and ventricles. The electrical currents that are generated from this process propagate through the body. A typical ECG waveform is shown in figure 1.

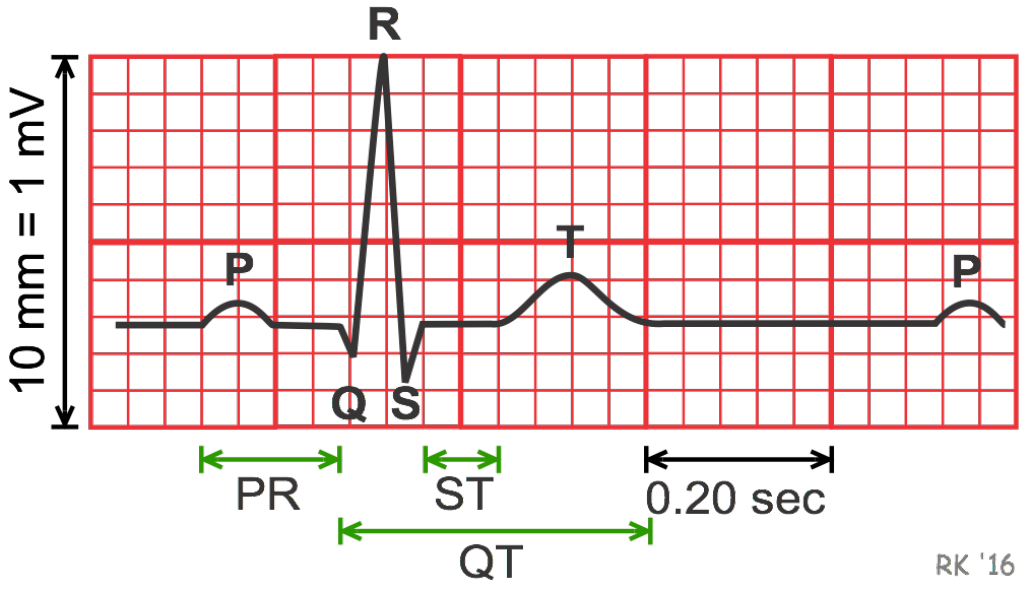


Figure 1: The typical ECG waveform (image credit: http://www.cvphysiology.com/Arrhythmias/A009.htm).

3.1.1 Recording ECG

There are multiple different approaches to measure the ECG, with varying degrees of precision, viewpoint, and localisation by placing electrodes at different positions on the human body. The concept behind the ECG is based on Einthoven's triangle, an imaginary formation of three limb leads in an equilateral triangle with the heart at the centre, which forms a reference system to analyse ECG waveforms. There are three standard lead placements forming the *bipolar* standard limb leads: lead I,

[3] https://uk.mathworks.com/help/matlab/matlab_env/what-is-the-matlab-search-path.html

an axis from left arm (+ electrode) to right arm (− electrode); lead II, axis goes from the right arm (−) to the left leg (+); and lead III, an axis from the left arm (−) to the left leg (+), as depicted in figure 2(a).

Additionally there are three augmented *unipolar* limb leads which are single positive electrodes that are referenced against a combination of the other limb electrodes. These leads can be "augmented" or constructed from the bipolar leads, allowing us to use the same electrode placement as the *bipolar* form. The positive electrodes for these augmented leads are located on the left arm (aVL), the right arm (aVR), and the left leg (aVF), as shown in figure 2(d).

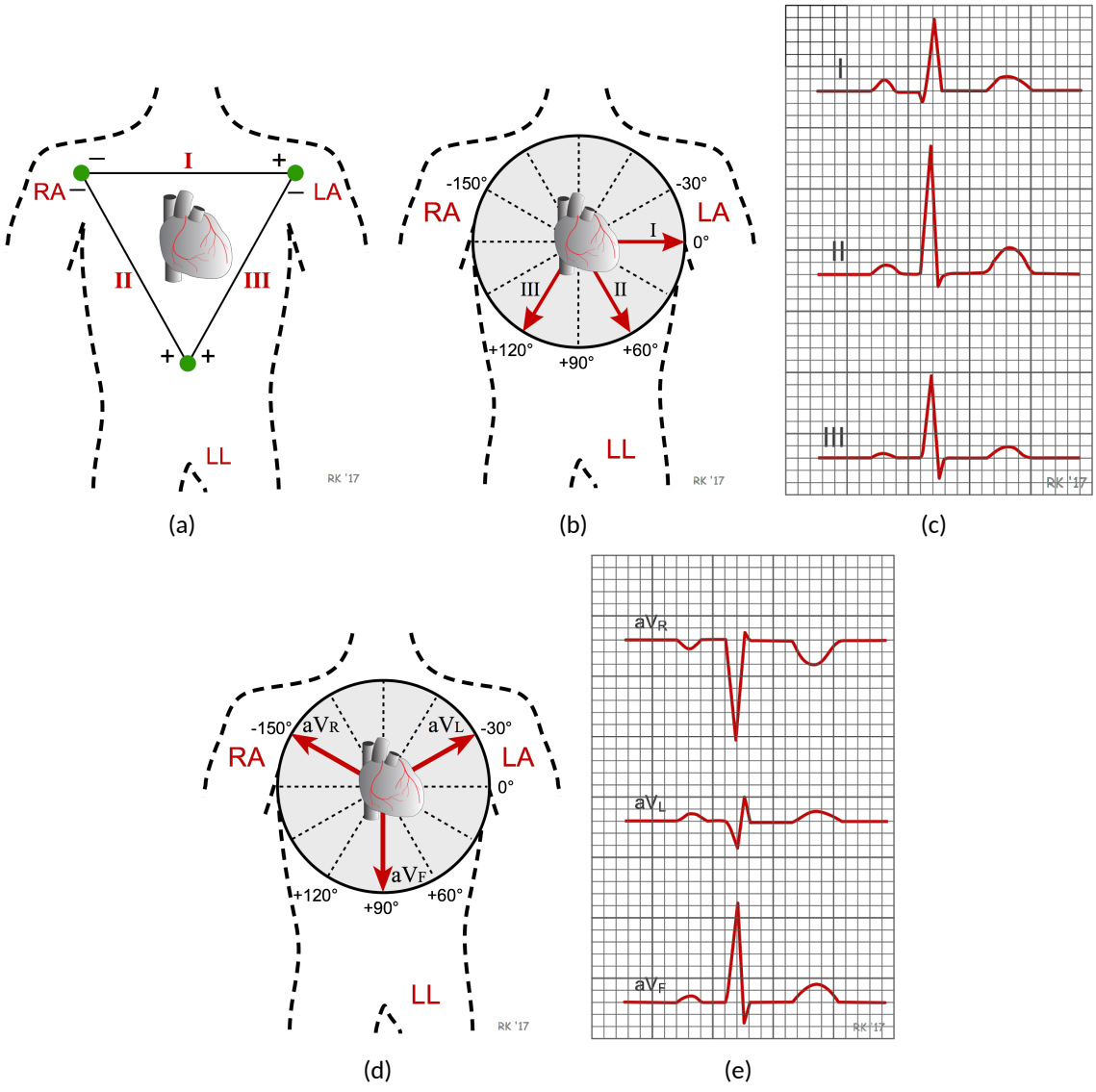


Figure 2: Electrode placement for (a) the standard *bipolar* 3-lead configuration and Einthoven's triangle construction. (b) *Bipolar* limb lead axis and (c) resulting ECG trace.(d) *Unipolar* augmented 3-lead axis and (e) resulting ECG trace.

The combination of the 6 *bipolar* and *unipolar* leads records electrical activity along a single plane, termed the frontal plane relative to the heart (see figure 3(a)). There are also six precordial, unipolar

chest leads which are often applied in rigorous clinical settings. These places six positive electrodes on the surface of the chest over different regions of the heart in order to record the electrical activity in a plane perpendicular to the frontal plane (see figure 3(b)).

Following your lecture notes, diagnostic information can be obtained from the ECG waveform by analysing the amplitude and relative timing of the various segments. ECG can be used for ambulatory monitoring, exercise and stress analysis or even for foetal ECG monitoring. Informative ECG features such as the heart rate, the timing of the PQRST complexes or heart rate variability (HRV).

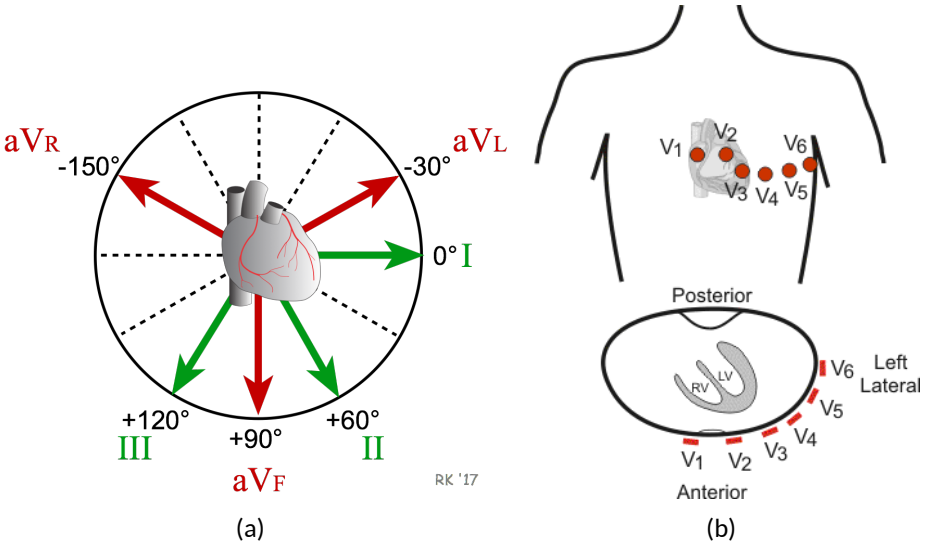


Figure 3: A full 12-lead ECG axis: (a) the combined *bipolar* and *unipolar* 6-axis ECG axis; (b) the ECG chest leads placement. Image credit: http://www.cvphysiology.com/Arrhythmias/A013c.htm.

3.1.2 Heart defects and diseases

As you've learned in the lecture notes, the heart beat rhythm is governed by pacemaker cells within the sinoatrial (SA) node which dictates the depolarisation and repolarisation of the atria and ventricles. Unfortunately, when this rhythm becomes irregular, too fast (tachycardia) or too slow (bradycardia), or the frequency of the atrial and ventricular beats are different, arrhythmias occur. Atrial fibrillation is one of the most common rhythm disturbances where this condition causes an irregular and often abnormally fast heart rate. Atrial fibrillation (AF) affects around 1 million people in the UK, affecting about 7 in 100 people aged over 65. Patients may describe an arrhythmia as a palpitation or fluttering sensation in the chest. Luckily, many arrhythmias can be treated with supression antiarrhythmic drugs. It is therefore critical to identify arrhythmias, such as AF, as early as possible so that patients can be provided with vital treatments. Other common heart defects and diseases include cardiac valve diseases, coronary artery diseases, hypertension and hypotension.

3.1.3 The Electrocardiogram - Preparatory questions

1. Explain briefly how the components of the ECG trace (P wave, QRS complex and T waves) are correlated with the electrical activity of the atrial and ventricular muscle.

2. Referring to the position of the electrodes, explain why you might expect lead II to give the largest amplitude of the QRS complex.

3. Describe some of the typical characteristics of an ECG recording which are noticeable when a patient has atrial fibrillation (AF).

3.2 Physical activity monitoring

Human mobility and physical activity patterns are well-established indicators of health and quality of life [1, 2, 3]. The rapid evolution of wearable sensors and devices for the collection of health-related data offers the opportunity to monitor mobility, gait and daily-physical activity patterns in patients. Nowadays, consumer wearable sensors such as smartphone and smartwatches specifically, are ubiquitous in everyday life. Often termed inertial measurement units (IMU), or inertial sensors, these wearable devices generally contain 3-axis accelerometer, gyroscope, and magnetometer sensors. Figure 4 demonstrates the smartphone and smartwatch device inertial sensor coordinates.

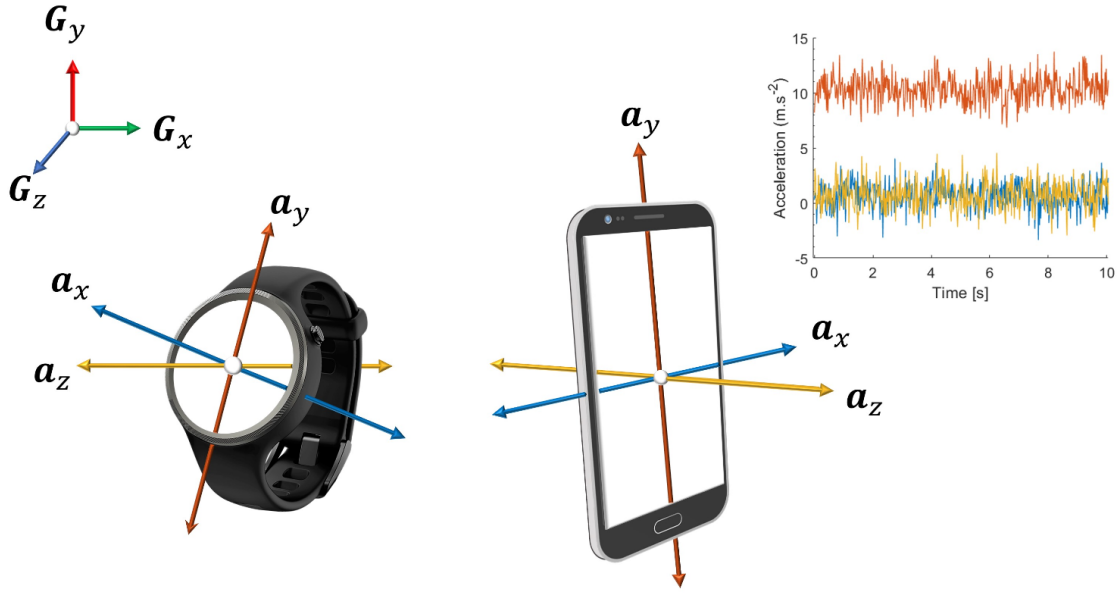


Figure 4: Smartphone and smartwatch device inertial sensor coordinates. Both devices contain a 3-axis accelerometer sensor, (a_x, a_y, a_z), which at any instantaneous time point t is incident to the direction of gravity (vertical), defined globally as ($G_x = 0$, $G_y = -9.81$m s^{-2}, $G_z = 0$). Image credit:[4].

Many studies have demonstrated the benefits of wearable-based daily activity monitoring to indicate health statuss, quality of life and to monitor patients. For example, it has been shown that a greater number of daily steps was significantly associated with lower mortality rates in a representative sample of US adults[5] (see figure 5). Other studies have shown that physical activity decreases with age in a representative sample of UK adults[3]. Furthermore, a wide range of diseases have a negative effect on physical activity, affecting the amount, type and way that patients perform certain activities and tasks[6].

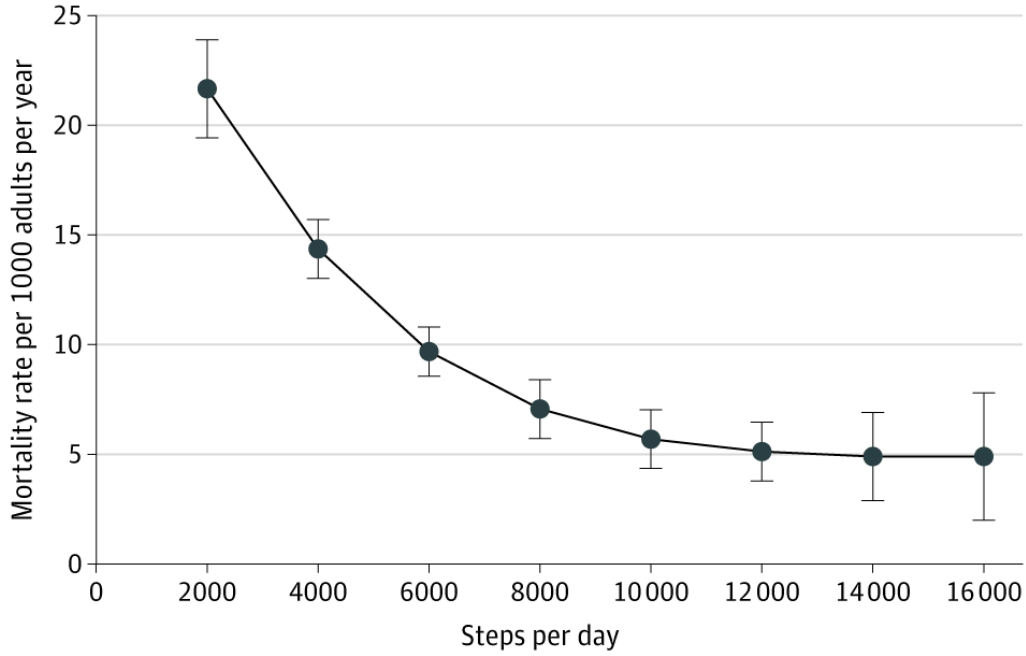


Figure 5: The association of daily step count with mortality rate amoug US adults at least 40 years old (n=4840). Credit: [5].

3.2.1 Physical activity monitoring - Preparatory questions

1. Smartphone and smartwatch devices typically contain a wealth of other sensors besides IMUs. Name one other sensor that these devices could have and a give potential health application this sensor could be used for.

2. Often you will need to re-orientate a device's inertial sensor with that of gravity, as shown in figure 4. Why might this be an important step to consider in processing IMU data?

3. Figure 6 presents various boxplot distributions. Briefly describe how to interpret a boxplot and its properties.

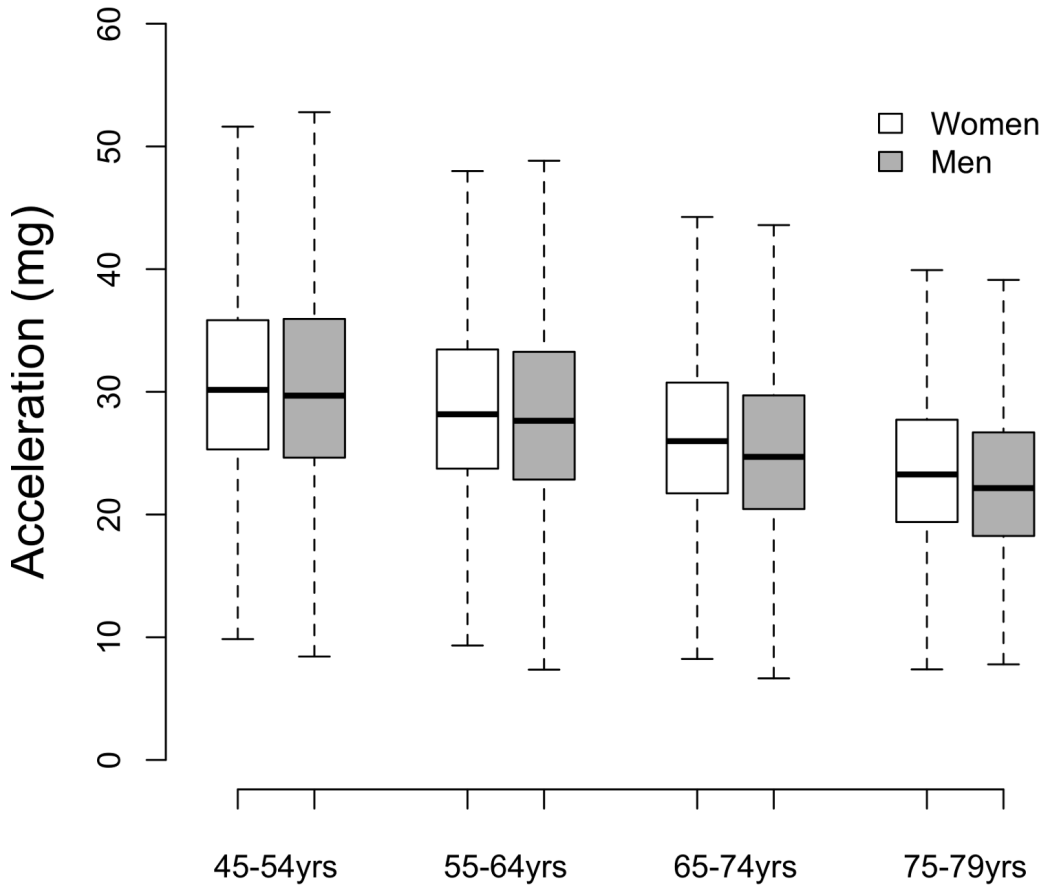


Figure 6: The distribution of the daily-average smartwatch acceleration vector magnitude (milli-gravity units [mg]) by sex and age in the UK Biobank study (n=96,600). Credit: [3].

4 Section A: The Electrocardiogram

This section aims to introduce some standard approaches to analysing electrocardiogram (ECG) data. The example exercise uses clinical ECG data from the PhysioNet/CinC 2017 Challenge[7]. PhysioNet is a community resource and archive of digital recordings of physiologic signals, time series, and related data for use by the biomedical research community.

The 2017 PhysioNet/CinC Challenge aimed to encourage the development of algorithms to classify, from a single short ECG lead recording (between 30 seconds and 60 seconds in length and all signals are sampled at 300Hz), whether the recording shows normal sinus rhythm, atrial fibrillation (AF), an alternative rhythm, or is too noisy to be classified. A sample recording is shown in figure 7. More details on the PhysioNet 2017 Challenge can be found at `https://physionet.org/content/challenge-2017/1.0.0`. The data consists of a set of ECG signals sampled at 300Hz and divided by a group of experts into four different ECG classes: Normal (N), AFib (A), Other Rhythm (O), and Noisy Recording (∼).

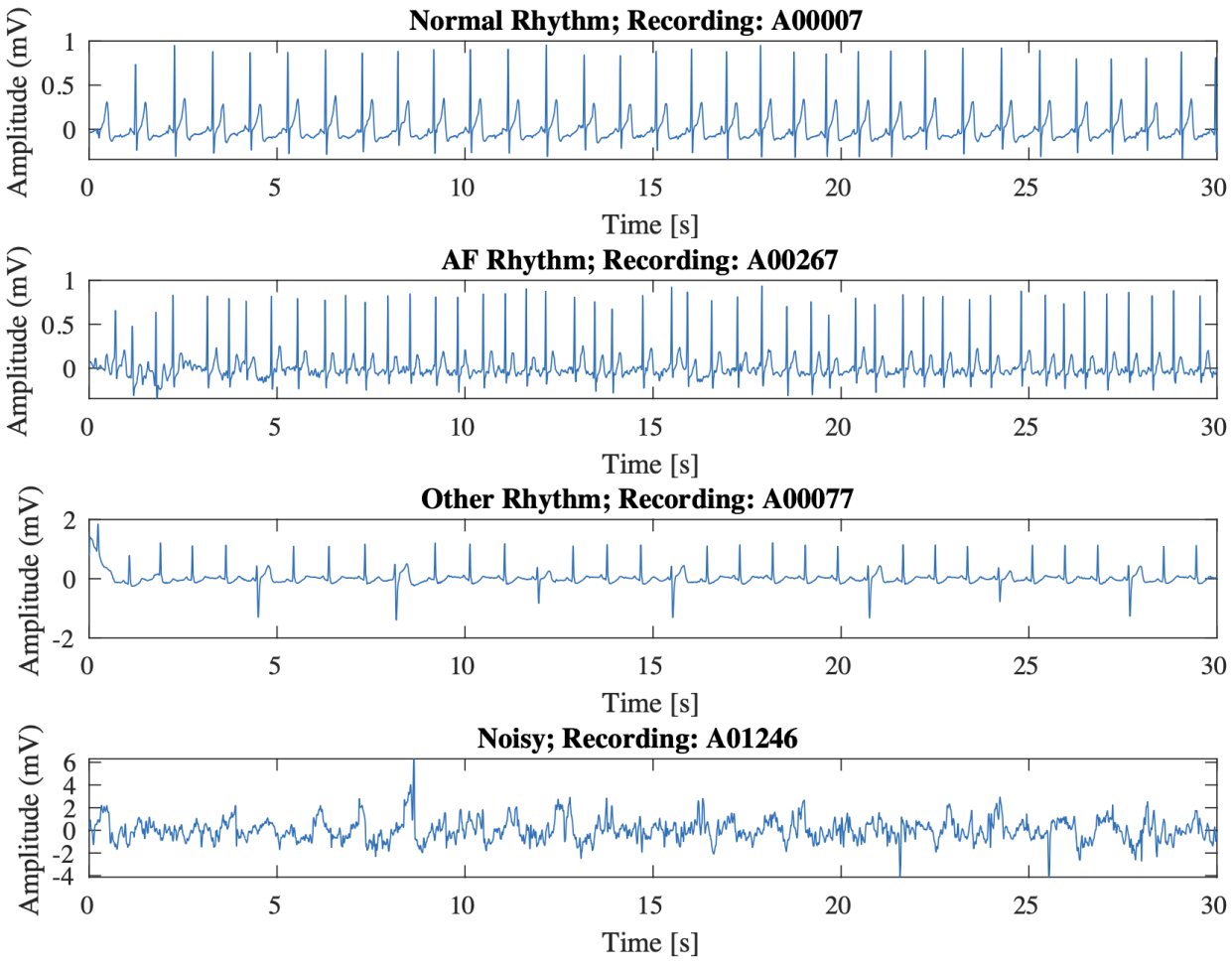


Figure 7: Examples of ECG waveforms for a normal rhythm, AF, other rhythm and a noisy recording.

4.1 Exercise 1: Exploratory ECG data analysis

This section begins the laboratory session by exploring the data in the PhysioNet challenge. In this exercise, we will plot a number of ECG examples from various subjects and recordings for different healthy (normal) and abnormal (AF, or other) examples.

You have been provided with an example script to perform the data loading, data wrangling, and plotting utilities, needed to complete this section. See the "`B18_ECG_SectionA_Exercise_1.m`" file.

Important

Choose any recordings you want from the whole PhysioNet data for all subsequent analysis in this exercise. Do NOT use the same example recordings in this lab sheet, namely: A00007, A00267, A00077 or A01246.

Tasks to perform:

1. Load the PhysioNet data into MATLAB;
2. Plot a suitable portion of ECG data from a normal subject, which depicts at least 2 heartbeats. Identify and label the PQRST points;
3. Visualise a segment of one ECG signal from each of the 4 classes from any chosen set of examples. Label each recording. See figure 7 for an example;
   (a) Describe some of the signal characteristics observed that distinguish normal ECG versus AF ECG in your chosen examples. Use extra plots to illustrate your discussion points if required;
   (b) With respect to your chosen noisy ECG example, or any artefacts observed in other examples, briefly discuss some of the reasons why ECG signals can be corrupted or noisy.

4.1.1 Hints

- The proportion of Normal (N), AFib (A), Other Rhythm (O), and Noisy Recording ($\sim$) classes in the dataset can be determined from the RecordingInfo table, using the "`tabulate.m`" function;
- The text.m function (see MATLAB documentation for some examples) can be useful for adding annotations to plots directly in MATLAB.

- To convert samples to time, simply create a separate monotonic vector of samples, converting samples to time as we all know $f = 1/t$ as shown in listing 1.

Listing 1: Sample to time vector conversion.

```
%FYI: the PhysioNet data is sampled at 300 Hz.
fs=300; %[user set variable]: Sampling frequency (Hz)
% 'signal' is your vector of ecg values
samples=1:length(signal);
%'time' is a corresponding monotonic vector of time values
time=samples./fs;
```

4.2 Exercise 2: Pre-processing the Electrocardiogram

Signal conditioning and pre-processing are important steps to clean and prepare data for further analysis later. Sometimes ECG signals contain an underlying trend, such as a change in the mean over time. Detrending is therefore applied by removing (subtracting) the mean component from the signal. Typically, the signal will be "*normalised*" or "*standardised*" to have unit variance. This standardisation is often performed using the z-score. For sample data x with mean $\bar{x}$ and standard deviation σ, the z-score of a data point x_i is:

$$z_i = \frac{x_i - \bar{x}}{\sigma} \tag{1}$$

where the sample standard deviation σ is given by:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1}} \tag{2}$$

Filtering ECG signals appropriately becomes the next step in pre-processing. Often signals are filtered with a combination of low-pass and high-pass filters, i.e. band-pass filters. You have been provided with a separate function to perform filtering "`filter_ecg.m`" using a Butterworth IIR band-pass. Butterworth filters have a smooth monotonic frequency response that is maximally flat in the passband, whereby they sacrifice rolloff steepness in favour of flatness. Using the normal and AF recording examples identified as part of exercise 3 in section 4.1:

1. Detrend the data by subtracting the mean component; then, standardise the detrended data using the z-score approach.

2. Using the same standardised and detrended normal and AF recording examples above, apply band-pass filtering to each recording (after detrending) using the "`filter_ecg.m`" file provided.

(a) Identify suitable filtering parameters: high-pass and low-pass frequency thresholds, filter order.

(b) Justify and discuss your choice of (1) high-pass frequency threshold; (2) low-pass frequency threshold with respect to your understanding of ECG signals.

3. Plot the resulting recordings for both your chosen normal and AF examples after detrending, standardisation and filtering has been applied.

4.2.1 Hints

- MATLAB have in-build functions for detrending (`detrend`) and normalising (`zscore`).
- You can easily implement detrending and normalisation yourselves. Remember, using element-wise operators in MATLAB reduces the need to construct for loops. For example the operation $x./y$ divides each element of x by the corresponding element of y. See MATLAB documentation for more details[4].
- See listing 2 for an example of the usage of the `filter_ecg.m` function.
- This is a question students can get stuck on. Consider what makes sensible filtering parameters. We want to filter out high-frequency noise in the signal (for example, 50Hz from A/C power supply) and low-frequency noise (for example breathing < 0.5Hz), yet retain important frequency components in the ECG (for example the HEART RATE usually at around 60beats$/$min ≈ 1Hz).

Listing 2: Usage of the filter_ecg function.

```
org_signal; %[user set variable]: the original unfiltered ECG signal
fs =300;    %[user set variable]: sampling frequency [Hz]
f_low=100;  %[user set variable]: low-pass frequency [Hz]
f_high=0.5; %[user set variable]: high-pass frequency [Hz]
norder =4;  %[user set variable]: filter order

% filter_ecg.m filters the original signal and outputs
% the band-pass filtered ECG signal
filtered_signal=filter_ecg(org_signal, fs, f_high, f_low, norder);
```

[4] https://uk.mathworks.com/help/matlab/ref/rdivide.html

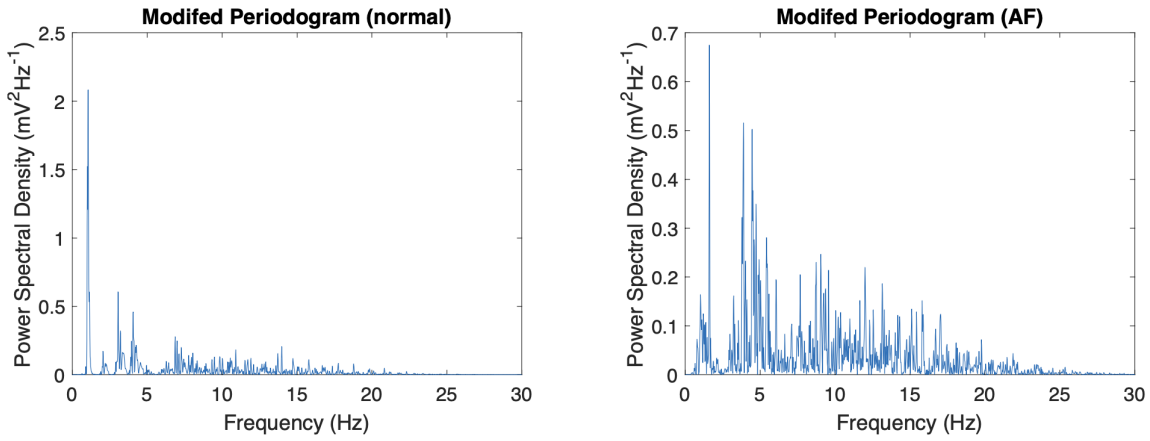
4.3 Exercise 3: Frequency representation of ECG Signals

A periodogram can be used to determine the frequency representation of an ECG. The periodogram is a non-parametric estimate of the power spectral density (PSD) of a signal which uses discrete Fourier transform (DFT) through the fast Fourier transform (FFT) algorithm. The PSD estimate of ECG vector $x \in \mathbb{R}^{1\times N}$ can be performed using the modified periodogram (see figure 8). The modified periodogram multiplies the input time series by a window function, in this case with a Hamming window function h_n. The modified periodogram PSD is defined by:

$$\hat{P}(f) = \frac{\Delta t}{N}\left|\sum_{n=0}^{N-1}(h_n x_n e^{-j2\pi fn})\right|^2 \tag{3}$$

where the periodogram is the DFT of the biased estimate of the autocorrelation sequence for the n^{th} sample x_n, sampled at fs per unit time; Δt is the sampling interval.

The periodogram is not a consistent estimator of the true PSD however. The Welch's Overlapped Segment Averaging Spectral Estimation aims to reduce the variance of the periodogram by segmenting the time series into overlapping segments and computing the modified periodogram for each segment. These PSD segments can then averaged give a more consistent estimate of the power spectral density.



(a) (b)

Figure 8: Modified periodogram power spectral density (PSD) estimation of (a) a normal ECG recording; (b) an AF ECG recording.

Using the same standardised, detrended and band-pass filtered data from normal and AF recording examples in exercise 3 in section 4.2:

1. Plot the modified periodogram PSD estimate for your chosen normal and AF recording examples.

   (a) Identify the dominant frequency and any other prominent peaks in the PSD; comment on their amplitude, frequency and your understanding with respect to ECG.

2. Plot Welch's power spectral density estimate for your chosen normal and AF recording examples.

   (a) Briefly comment on the differences between both PSD methods. Why might Welch's PSD estimate be preferable for analysing biomedical signals?

4.3.1 Hints

Listing 3 shows an example on how to compute the modified periodogram power spectral density (PSD) estimate.

Listing 3: Periodogram power spectral density (PSD) estimate.

```
% To return the modified periodogram power spectral density (PSD)
% estimate, pxx, of the input signal, x, using a hamming window:

x; % [user set variable]: input ECG signal, measured in mV
   % Note: ensure x is detrended for prior to input into the
   % periodogram / FFT;

f_low; %[user set variable]: the low-pass frequency used during
       % filtering previously

window=hamming(length(x)); % a Hamming window function;

% (1) modified-periodogram:
[pxx,f] = periodogram(x,window,[],fs,'psd');

% (2) Welch's power spectral density estimate
% (pwelch.m automatically implements a Hamming window):
% [pxx_pwelch,f_pwelch] = pwelch(x,[],[],[], fs,'psd');

%to plot the PSD
figure
plot(f, pxx);
xlim([0, f_low])
xlabel('Frequency (Hz)')
ylabel('Power Spectral Density (mV^{2}Hz^{-1})')
%hint: for decibels (dB) plot 10*log10(pxx) instead;
```

4.4 Exercise 4: Determining heart rate and heart rate variability

Digital ECG signals can be analysed for events of interest which can be used to determine and monitor a patient's health status. Beyond the morphology of ECG signals, such as the shape or magnitude of the QRS complex describing the depolarisation and re-polarisation of the heart chambers, ECG signals are frequently non-stationary meaning that their frequency content changes over time. Therefore the timing of these events plays a fundamental role in understanding or monitoring how various diseases can affect the heart. Heart rate, or the number of heart beats per minute, varies due to the physical needs of the body. The rhythm of the heart is controlled by the sinoatrial nerve which is regulated by sympathetic and parasympathetic activity, the two main division of the autonomic nervous system (ANS).

Calculating the heart rate is the most basic time-series based measurement we can collect. There are many methods and complex algorithms to determine heart rate in the literature, however a threshold- based detection of the large R-peak is the simplest approach. This section will make use of the existing MATLAB "`findpeaks.m`" function (See MATLAB documentation[5] for further details).

4.4.1 Computing heart rate through R-peak detection

Using the same standardised, detrended and band-pass filtered data from normal and AF recording examples in exercise 3 in section 4.2:

1. Perform R-peak detection using the MATLAB `findpeaks.m` function for your chosen normal and AF recording examples.

   (a) Choose suitable parameter values for the `findpeaks.m` function in order to best identify normal and AF R-peaks correctly.

   (b) Justify your parameter value choices.

   (c) Plot the normal and AF recording examples with the with the automatically-detected R-peak locations annotated.

   (d) Discuss the limitations of this approach to R-peak detection. Briefly comment on any other methodologies for R-peak detection you have found in the literature, including a citation from a relevant paper.

2. Determine the mean heart rate from the automatically detected R-peaks for your chosen normal and AF recording examples.

[5] https://uk.mathworks.com/help/signal/ref/findpeaks.html

4.4.2 Computing heart rate variability measures

Heart rate variability (HRV) analysis aims to understand the variation in time between each heartbeat[8]. Cardiac autonomic regulation modulates the heart rate, the variation of heart rate is mainly mediated by the sinus (sino-atrial) node, acting on heart-brain communications as well as the autonomic nervous system (ANS). HRV reflects regulation of a range of subtle physiological processes for example autonomic balance, blood pressure (BP), gas exchange, or vascular tone, which refers to the diameter of the blood vessels that regulate BP.

To evaluate HRV, the variation in the R-peak-to-R-peak interval (RR) sequence of the ECG needs to be quantified. Traditional HRV analysis can be separated into two time domain measures and frequency domain measures[8, 9]. In this example you will quantify common HRV measures from the (1) time-domain: the distribution, mean and variance in heart rate; and (2) the frequency domain: estimates of the distribution of absolute or relative power into certain frequency bands. Heart rate oscillations can be divided into: very low frequency (VLF) band (0.0033–0.04Hz), low frequency (LF) band (0.04–0.15Hz), and high frequency (HF) components (0.15–0.40Hz). It hypothesised that during "at rest" (i.e. normal sinus RR-intervals), high frequency variations in sinus rhythm (RR-intervals) reflect parasympathetic (vagal) modulation of respiration (0.15-0.40Hz). Higher frequencies of > 0.5Hz typically reflect non-normal sinus rhythms (i.e. not at rest), such as during strenuous exercise. Low frequency variations in RR-intervals (0.04 - 0.15Hz) are physiologic oscillations associated with a combination of both parasympathetic and sympathetic modulation and non-autonomic factors.

As we learned in section 4.3, a periodogram can be used to determine the frequency representation of an ECG. The RR interval sequences however will likely be unevenly sampled (i.e. the RR intervals are not uniformly spaced as they are not occurring at the same rate). The Lomb-Scargle periodogram instead has been shown to produce a more accurate estimation of the PSD than DTFT-based periodograms for unevenly sampled data[10]. For a vector $x \in \mathbb{R}^{1\times N}$ (the RR intervals) with x_n, sampled at times t_n, where $n = 1, ..., N$, the Lomb-Scargle periodogram PSD is defined as:

$$\hat{P}_{ls}(f) = \frac{1}{2\sigma^2}\left\{\frac{\left[\sum_{n=1}^{N}(x_n-\bar{x})cos(2\pi f(t_n-\tau))\right]^2}{\sum_{n=1}^{N}cos^2(2\pi f(t_n-\tau))} + \frac{\left[\sum_{n=1}^{N}(x_n-\bar{x})sin(2\pi f(t_n-\tau))\right]^2}{\sum_{n=1}^{N}sin^2(2\pi f(t_n-\tau))}\right\} \tag{4}$$

where $\bar{x}$ is the mean of x; σ^2 is unbiased estimate of the variance in x; the time offset. The time delay τ is determined by:

$$tan(2(2\pi f)\tau) = \frac{\sum_{n=1}^{N}sin(2(2\pi f)t_n)}{\sum_{n=1}^{N}cos(2(2\pi f)t_n)} \tag{5}$$

Any shift $tn \rightarrow tn + T$ in the time measurements results in an identical shift in the offset: $\tau \rightarrow \tau + T$

1. Plot the R-R interval times over the duration of the recording from the automatically detected R-peaks for your chosen normal and AF recordings; See figure 9 for an example of this.

2. Plot a histogram of the R-R interval times over the duration of the recording from the automatically detected R-peaks for your chosen normal and AF recording examples.

3. Plot the Lomb-Scargle periodogram PSD of RR-intervals for your chosen normal and AF recording examples.

4. Briefly discuss your findings comparing your chosen normal and AF recording examples in exercise A.4.1 and A.4.2 with respect to heart rate, R-R interval times and heart-rate variability. Compare these findings to what you might expect to observe.

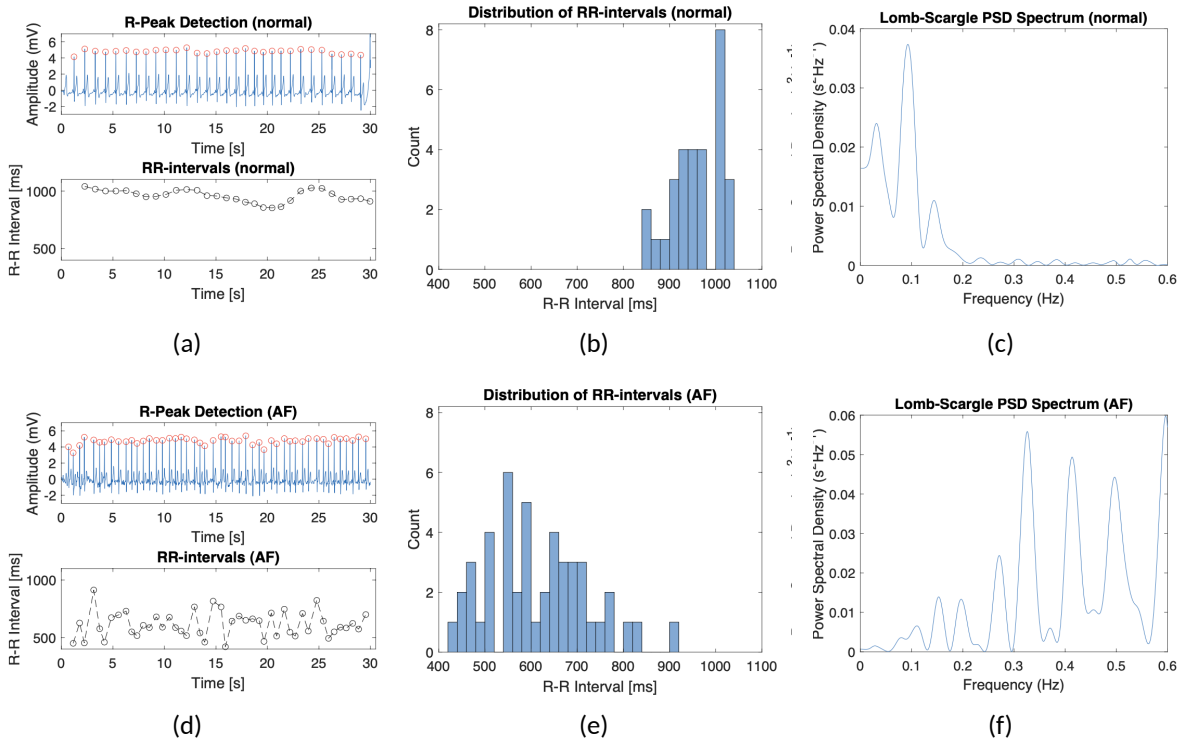


Figure 9: Panel plot demonstrating the calculation of heart rate and heart rate variability (HRV) measures from ECG. The top row (a)-(c) depicts a normal ECG, with an average heart rate: 62.83 beats/min; the bottom row. (d)-(f) Depicts a AF ECG, with an average heart rate: 100.80 beats/min. Panels (a) and (d) represent an annotation of the automatically detected R-peaks, with the corresponding RR-interval [ms] between successive beats for (a) normal and (d) AF rhythms respectively. Panels (b) and (e) represent the distribution of the RR-interval series as histograms for (b) normal and (e) AF rhythms. The Lomb-Scargle periodogram PSD estimation of RR-intervals are represented for (c) normal and (f) AF rhythms.

4.4.3 Hints

- If your chosen ECG recording has artefacts it is often pertinent to return to exercise 2(a) in section 4.2 and redetermine a filtering setting with a lower low-pass threshold.
- It is recommended to use the "`MinPeakHeight`" and "`MinPeakDistance`" parameters in `findpeaks.m` to initially threshold R-peaks. Further usage of this function can be found in the MATLAB doc- umentation[6]. Remember, the threshold of "`MinPeakDistance`" will be represented in samples, not seconds. As such, you may need to convert your threshold from seconds to number of samples, again using $f = 1/t$. Experiment with other parameter settings of `findpeaks.m` as you wish (see listing 4).

Listing 4: Find_peaks usage example.

```
1
2  % findpeaks usage
3  signal; % [user set variable]: the input ECG signal;
4  time;   % [user set variable]: monotonic vector of
5          % time values;
6  % [user set variable]: Your chosen scalar threshold values:
7  % MinPeakHeight; MinPeakDistance;
8
9  % findpeaks.m function will output the R-peaks
10 % detected (R_pks) and the locations of the peaks
11 % (R_locs);
12
13 [R_pks,R_locs]=findpeaks(signal, 'MinPeakHeight',MinPeakHeight,
14     'MinPeakDistance', MinPeakDistance);
15
16 %plot the R-peak annotations;
17 figure
18 plot(time, signal);
19 hold on;
20 plot(time(R_locs), signal(R_locs), 'ro');
```

- Listing 5 shows sample code to construct a simple histogram (See MATLAB documentation[7] for further details).

[6] https://uk.mathworks.com/help/signal/ref/findpeaks.html

[7] https://uk.mathworks.com/help/matlab/ref/matlab.graphics.chart.primitive.histogram.html

Listing 5: Example to plot a histogram.

```
1
2  % to get the RR-intervals in seconds, use the diff.m function.
3  % diff(X) calculates differences between adjacent elements of X.
4  % Calculate the difference between consecutive RR intervals;
5
6  % convert samples to seconds by dividing by the sampling rate:
7  RRs=diff(R_locs)./fs;
8
9  %convert from [s] to [ms]:
10 RRms=RRs.*1000
11
12 % Constructing a simple probabilistic histogram
13 BinWidth=20; %[user set variable]: the width of the bins figure
14 histogram(RRms, 'BinWidth', BinWidth)
15 xlabel('R-R Interval [ms]')
16 ylabel('Count')
```

- Listing 6 shows code to evaluate the Lomb-Scargle periodogram PSD of RR-intervals using the plomb.m MATLAB function. See MATLAB documentation[8].

Listing 6: Example to evaluate the Lomb-Scargle periodogram.

```
1
2  % Lomb-Scargle PSD RR-interval Spectrum;
3  RRs; %The RR-interval in seconds [s];
4  RRs=RRs-mean(RRs); %detrend the RR intervals (best practice);
5
6  % Determine the time points in the ecg each R-peak occurred
7  % from the first interval (i.e. the second R-peak detected);
8  t=time(R_locs(2:end));
9
10 % Denote the frequencies we are interest in evaluating:
11 % [0.001, 0.6] Hz;
12 f_interest =0.001:0.001:0.6;
13
14 % Compute the Lomb-Scargle periodogram;
```

[8] https://uk.mathworks.com/help/signal/ref/plomb.html

```
[pxx_plomb,f_plomb] = plomb(RRs, t, f_interest, 'psd');

%to plot the PSD
figure
plot(f_plomb , pxx_plomb);
xlabel('Frequency (Hz)')
ylabel('Power Spectral Density (s^{2}Hz^{-1})')
```

- For you information: simple HRV time-domain and frequency-domain measures can be extracted for recording examples using the provided MATLAB function hrv measures.m. Listing 7 shows a sample usage case.

Listing 7: Example to compute simple HRV metrics.

```

% RRms=RRs*1000: A vector of RR intervals,
% converted from seconds to miliseconds [ms];

% f_plomb   : The frequency vector returned by the
%             -LombScargle periodogram ;
% pxx_plomb : The -LombScargle power spectral density (PSD)
%             estimate at f ;

% Function to return Heart Rate Variability (HRV) measures:
HRV=hrv_measures(RRms, f_plomb, pxx_plomb);

% display the HRV measures :
display(HRV)
```

5 Section B: Smartphone physical activity monitoring

In this section you will investigate smartphone accelerometry from the annotated accelerometer ubicomp2013 dataset by Brajdic and Harle (2013)[11]. Further information on the dataset can be found at: https://doi.org/10.17863/CAM.12982. The ubicomp2013 study collected smartphone inertial measurement unit (IMU) data (tri-axial accelerometer, gyroscope, magnetometer sensors) from a number of participants as they walked. The smartphone was placed in different locations on the body across different experiments. These included the front and back pockets, held in the subject's hand or in a backpack.

In this lab we are only interested in smartphone 3-axis accelerometer data and those experiments performed with the smartphone in the front pocket. Figure 10(a) depicts an example of a smartphone 3-axis accelerometer trace captured from a participant in the ubicomp2013 study with the smartphone in the front pocket. The ubicomp2013 study was performed under video observations, so each experiment file has been annotated for the actual number of steps taken and the times when the walk began and ended.

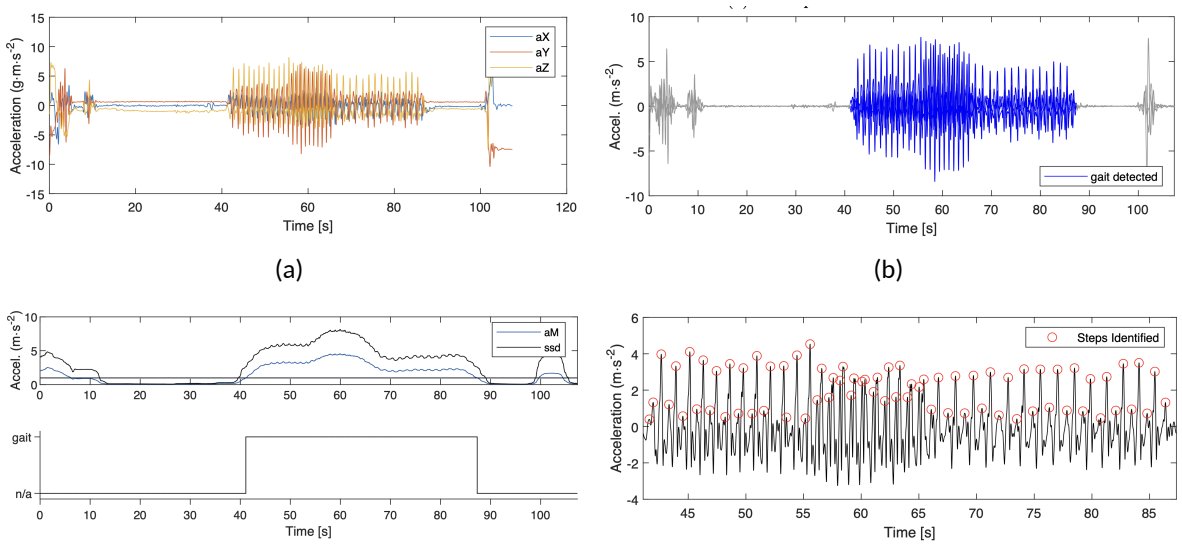


Figure 10: Smartphone walking identification and step detection algorithm pipeline. (a) Depicts the raw smartphone 3-axis accelerometer data. (b) Highlights the region of (processed) sensor signal identified as walking. (c) Demonstrates the parameterisation of the heuristic walking detection algorithm: the moving average acceleration (aM) and the combined standard deviation (ssd) threshold. (d) Annotates the steps automatically identified through peak detection across the detected walking bout.

The simplest gait-based health indicators are daily-step counts, which estimate the number of steps per-day and therefore the daily activity and activity intensity. To build a step detection algorithm we must first identify periods of walking (termed a bout). Next we perform step counting on those detected bouts. Figure 10 demonstrates the walking detection and step counting pipeline.

You are provided with two MATLAB source files: "`B18_GAIT_SectionC_Exercise_1.m`" and "`B18_GAIT_SectionC_Exercise_2.m`" for exercise 1 and 2 respectively, which will allow you to perform the analysis for this section.

Important

These files are pre-programmed to load and analyse the data files, however you will have to change some of the parameter options.

5.1 Walking detection algorithm

In this lab we will use a heuristic[9] walking detection algorithm based on the approaches introduced in [12, 13]. This method utilises a threshold approach to detect period of walking:

1. First a moving average window of the magnitude of acceleration (aM) is determined.
2. A moving average window of the summed standard deviation (ssd) across tri-axial accelerometer axis is calculated (i.e. the sum of the standard deviation of each a_x, a_y, and a_z).
3. The moving average ssd is compared to a threshold value "`ssd_threshold`" to detect periods of movement variation.
4. Periods of walking are determined by $aM > acc\_threshold$ and $ssd > ssd\_threshold$ simultaneously.

Figure 10(b) shows an example of smartphone-based walking detection, with figure 10(c) demonstrating the parameterisation of the walking detection algorithm.

5.2 Step counting algorithm

There are many methodological approaches to detect steps from wearable accelerometry. The simplest is a heuristic peak detection, much like you implemented during section 4.1, which can also be evaluated using the `findpeaks.m` function in MATLAB. You will need to determine the threshold values for step detection: `MinPeakDistance`, the minimum distance between consecutive peaks, in seconds [s]; the `MinPeakHeight` the minimum height of a step peak, measured in [m s^2]. Figure 10(c) demonstrates heuristic peak identification for step detection from an identified walking bout.

[9] From Wikipedia: "A heuristic technique, is any approach to problem solving or self-discovery that employs a practical method that is not guaranteed to be optimal, perfect, or rational, but is nevertheless sufficient for reaching an immediate, short-term goal or approximation."

5.3 Exercise 1: Building a smartphone walk detection and step counting algorithm

The goal of this exercise is to understand how the parameters for walking detection and step counting can be determined heuristically. In the laboratory source files, you will find one MATLAB source file `B18_GAIT_SectionB_Exercise_1.m`. You should aim to compare parameter choices across multiple different files:

1. Run the `B18_GAIT_SectionC_Exercise_1.m` file for any example file of your choice.
   (a) Pick suitable `acc_threshold` and `ssd_threshold` values for walking detection.
   (b) Pick suitable `MinPeakDistance` and `MinPeakHeight` values for step detection.
   (c) Record your parameter values.
   (d) Record the actual number of steps, the number of steps detected and the cadence for those parameter values.
2. Similar to figure 10, for your chosen file, plot:
   (a) Raw 3-axis smartphone acceleration signal.
   (b) The walking detection outcome.
   (c) The walking detection algorithm parameters chosen.
   (d) The step detection annotations.
3. Repeat steps 1 and 2 for at least one other recording.

5.3.1 Hint

- To load the data, you will need to change the "`pathname`" variable to the directory on your computer where the data is stored, as shown in listing 8.

Listing 8: Example to set the pathname variable.

```

% Load the Data
% the pathname where the data is stored; You will need to
% change this your directory where the data is stored,
% e.g.:
pathname='C:\MATLAB\DATA\' ;

% Set the device location; in this example we are only
```

```
% interested in smartphone accelerometry that was recorded
% when the phone was in the front trousers pocket.
device location='front_pocket';

% returns a table consisting of the filenames extracted
% and the corresponding subject ID as strings.
fileInfo=return_filenames_gait(pathname, device_location);
```

- See listing 9 as an example for how to change the parameter values.

Listing 9: Example to set the threshold parameters.

```

G=9.81; %[fixed variable]: gravity 9.81 m/^s2

% [user set variable]: set the moving average
% accel. threshold value to 15% of gravity
options.acc_threshold=0.15*G;

% [user set variable]: set the moving average
% accel. threshold value to 30% of gravity
options.ssd_threshold =0.3*G;


%[user set variable]: set to 0.5 [s]
options.MinPeakDistance = 0.5;

% [user set variable]: set to 10% of gravity
% [m/s^2]
options.MinPeakHeight = 0.1*G
```

- To chose a specific file, you will also need to change the "`file_index`" variable to a value of your choice corresponding to the filename you wish to analyse, as shown in listing 10.

Listing 10: Example to modigy the file_index variable.

```

% Run the walking detection and step detection algorithm
%
% Choose one of the filenames from the dataset and
```

```
% corresponding subject identifier

% [user set variable]: choose a value [1, 2, ..., N],
% where N are the number of files
file_index = 10;

% Get the corresponding filename and subject name
% from the fileInfo table :
filename = fileInfo.filename(file_index);
subject  = fileInfo.subject(file_index);
```

5.4 Exercise 2: Assessing step counting algorithm performance

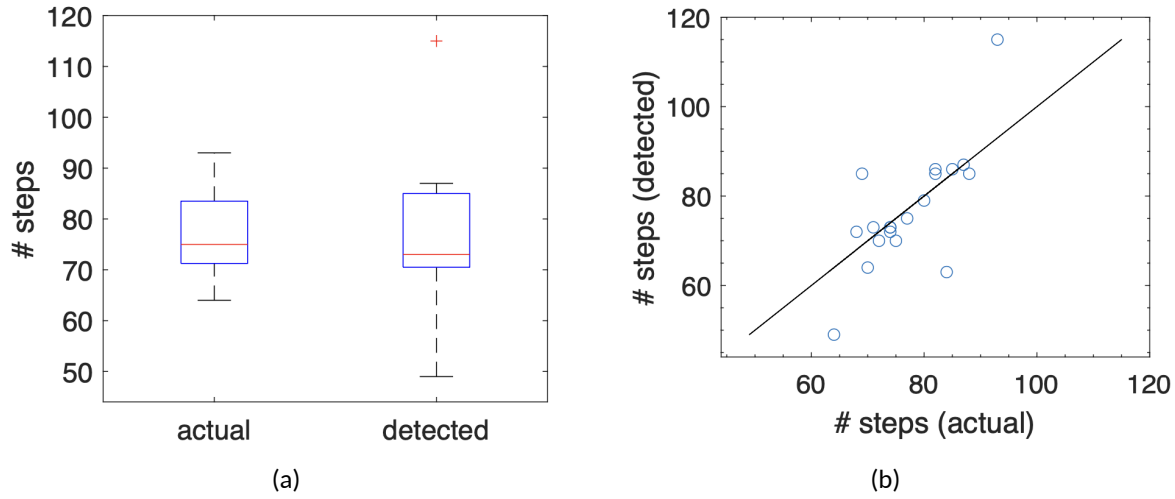


Figure 11: Analysis of step detection performance across the ubicomp2013 study. (a) Illustrates the boxplot distribution of actual vs. detected number of steps per subject; (b) depicts a scatter plot of the actual vs. detected number of steps per subject. A black line represents prefect step detection. (# steps observed: 77.3 ± 7.8 steps, # steps detected: 76.9 ± 13.4; $p = 0.86$; $MAE = 5.8$, $MSE = 80.9$).

In this exercise, you will assess the generalised step count algorithm performance across all participants. You are provided with the MATLAB file `B18_GAIT_SectionB_Exercise_2.m`. This file will run the walking detection and step detection for all participants in the entire ubicomp2013 study (as with exercise 1, we are only interested in front pocket smartphone locations for this lab). The `B18_GAIT_SectionB_Exercise_2.m` file will then plot the ubicomp2013 population distribution of the actual number steps observed versus the number of steps detected through the heuristic step detection algorithm (see figure 11). Further performance summary statistics which measure the error of our step detection algorithms are also evaluated, such as the mean absolute error (MAE) and mean squared error (MSE) between actual and detected number of steps per subject:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} \mid y_i - \hat{y} \mid \tag{6}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y})^2 \tag{7}$$

where y are the steps observed; $\hat{y}$ are the steps detected and N are the number of subjects. In this example the average and standard deviation in the number of steps per subject was observed as: 77.3 ± 7.8 steps, compared 76.9 ± 13.4 steps detected by our heuristic step detection algorithm. The mean absolute error (MAE) between steps observed and steps detected was determined as 5.8 steps; the MSE between steps observed and steps detected was determined as 80.9. This is relatively good step detection performance!

1. Run the `B18_GAIT_SectionC_Exercise_2.m` script with your chosen parameter values for `acc_threshold` and `ssd_threshold` values for walking detection and `MinPeakDistance`, `MinPeakHeight` values for step detection.

   (a) Plot the boxplot distribution of actual vs. detected number of steps per subject.

   (b) Plot the scatter plot of the actual vs. detected number of steps per subject.

   (c) Record the MAE and MSE error metrics between the actual vs. detected number of steps.

   (d) Briefly discuss the performance of your chosen parameters.

2. Briefly discuss some of the difficulties and limitations with the heuristic walking detection and step counting algorithm analysed at the start of section 5.

3. From the literature, suggest any another method for either walking detection or step counting.

5.4.1 Hint

- Remember to modify the `B18_GAIT_SectionC_Exercise_2.m` source file to include your chosen parameter values for `acc_threshold`, `ssd_threshold` values for walking detection and `MinPeakDistance`. `MinPeakHeight` values for step detection; To do this, refer to the hints for section B, exercise 1 previously.
- Don't get caught up spending time on parameter optimisation. The purpose of this example is not to obtain perfect performance, but to understand, implement, and critique simple heuristic methods that can perform walking and step detection.

Bibliography

[1] C.-C. Yang and Y.-L. Hsu, "A review of accelerometry-based wearable motion detectors for physical activity monitoring," *Sensors*, vol. 10, no. 8, pp. 7772–7788, 2010.

[2] J. J. Kavanagh and H. B. Menz, "Accelerometry: a technique for quantifying movement patterns during walking," *Gait & posture*, vol. 28, no. 1, pp. 1–15, 2008.

[3] A. Doherty, D. Jackson, N. Hammerla, T. Plötz, P. Olivier, M. H. Granat, T. White, V. T. Van Hees, M. I. Trenell, C. G. Owen, *et al.*, "Large scale population assessment of physical activity using wrist worn accelerometers: the uk biobank study," *PloS one*, vol. 12, no. 2, p. e0169649, 2017.

[4] A. P. Creagh, *The development of digital biomarkers for multiple sclerosis from remote smartphone- and smartwatch-based assessments*. PhD thesis, University of Oxford, 2020.

[5] P. F. Saint-Maurice, R. P. Troiano, D. R. Bassett, B. I. Graubard, S. A. Carlson, E. J. Shiroma, J. E. Fulton, and C. E. Matthews, "Association of daily step count and step intensity with mortality among us adults," *Jama*, vol. 323, no. 12, pp. 1151–1160, 2020.

[6] A. P. Creagh, C. Simillion, A. K. Bourke, A. Scotland, F. Lipsmeier, C. Bernasconi, J. van Beek, M. Baker, C. Gossens, M. Lindemann, *et al.*, "Smartphone-and smartwatch-based remote characterisation of ambulation in multiple sclerosis during the two-minute walk test," *IEEE journal of biomedical and health informatics*, vol. 25, no. 3, pp. 838–849, 2020.

[7] G. D. Clifford, C. Liu, B. Moody, H. L. Li-wei, I. Silva, Q. Li, A. Johnson, and R. G. Mark, "Af classification from a short single lead ecg recording: The physionet/computing in cardiology challenge 2017," in *2017 Computing in Cardiology (CinC)*, pp. 1–4, IEEE, 2017.

[8] U. Rajendra Acharya, K. Paul Joseph, N. Kannathal, C. M. Lim, and J. S. Suri, "Heart rate variability: a review," *Medical and biological engineering and computing*, vol. 44, pp. 1031–1051, 2006.

[9] F. Shaffer and J. P. Ginsberg, "An overview of heart rate variability metrics and norms," *Frontiers in public health*, p. 258, 2017.

[10] N. R. Lomb, "Least-squares frequency analysis of unequally spaced data," *Astrophysics and space science*, vol. 39, pp. 447–462, 1976.

[11] A. Brajdic and R. Harle, "Walk detection and step counting on unconstrained smartphones," in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pp. 225–234, 2013.

[12] A. Hickey, S. Del Din, L. Rochester, and A. Godfrey, "Detecting free-living steps and walking bouts: validating an algorithm for macro gait analysis," *Physiological measurement*, vol. 38, no. 1, p. N1, 2016.

[13] G. Lyons, K. Culhane, D. Hilton, P. Grace, and D. Lyons, "A description of an accelerometer-based mobility monitoring technique," *Medical engineering & physics*, vol. 27, no. 6, pp. 497–504, 2005.