Heart Rate Monitors

Avantika

December 1, 2016

Overview

The main focus of this project is to investigate the use of commercial wearables as possible patient monitoring devices by testing the precision of heart rate measurement by a wrist worn heart rate monitor (FitBit) against a portable ECG-based heart rate computer (Bitalino). The project will involve data collection, extraction, cleaning and conversion followed by a statistical model to test the accuracy of the FitBit band.

Introduction

Wearable devices are being studied and explored for use as diagnostic and patient monitoring devices replacing the current complicated and expensive medical devices.Now-a-days, wrist monitors like FitBit, Apple and Samsung are popular in the market. These devices use photoplethysmography which detects blood racing through the veins by optical sensors. Majority of the limited literature review proves that the bands are mostly accurate in heart rate measurement with few abnormalities at high bpm.

Heart rate variation is the physiological phenomenon of variation in the time interval between heartbeats. It is measured by the variation in the beat-to-beat interval. The SA node receives several different inputs from the Sympathetic and Parasympathetic Nervous system. Factors that affect the input are the baroreflex, thermoregulation, hormones, sleep-wake cycle, meals, physical activity, and stress. HRV can be used for stress response detection due to behavioural, bacterial or other abnormalities.

The problem addressed in this project covers the field of diagnostics, biomedical devices, signal processing and data analysis. The project layout consists of device set up and measurement of heart rate at different physical conditions(3 minutes sitting - 3 minutes walking - 3 minutes sitting after walking) with both the devices -Fitbit and Bitalino taking measurement at the same time. The data extracted was filtered and converted to appropriate units for comparison.

The statistical model involves quantification by Bland-Altman plots, calculation of mean bpm for every episode and correlation coefficient. I met Mr. Jeff Pennington from Children's Hospital of Philadelphia who is providing the devices used in this project. CHOP is interested in exploring the use of wearable devices and this project may aid them in determining the accuracy of these devices and setting up a bench-mark for their future study.

Methods

My project includes the set up, data acquisition, data extraction, signal processing, mathematical conversions and analysis of data from devices - FitBit and Bitalino ECG device. The analysis flow has been shown below -

Data Extraction

This is the first step after acquisiton of data. I am using R package fitbitscraper to extract data from the fitbit cloud. The data extracted consists of multiple variables like confidence level, date, time, heart rate zones, calories burned, etc. but the one I am interested in is the bpm (beats per minute data). The FitBit band measures data every one minute but allows us to extract data at the interval of 5 minutes only. They are currently working on the plug-in to give access to the entire heart rate data. This has proved to be a major setback of any wearable device.

FitBit

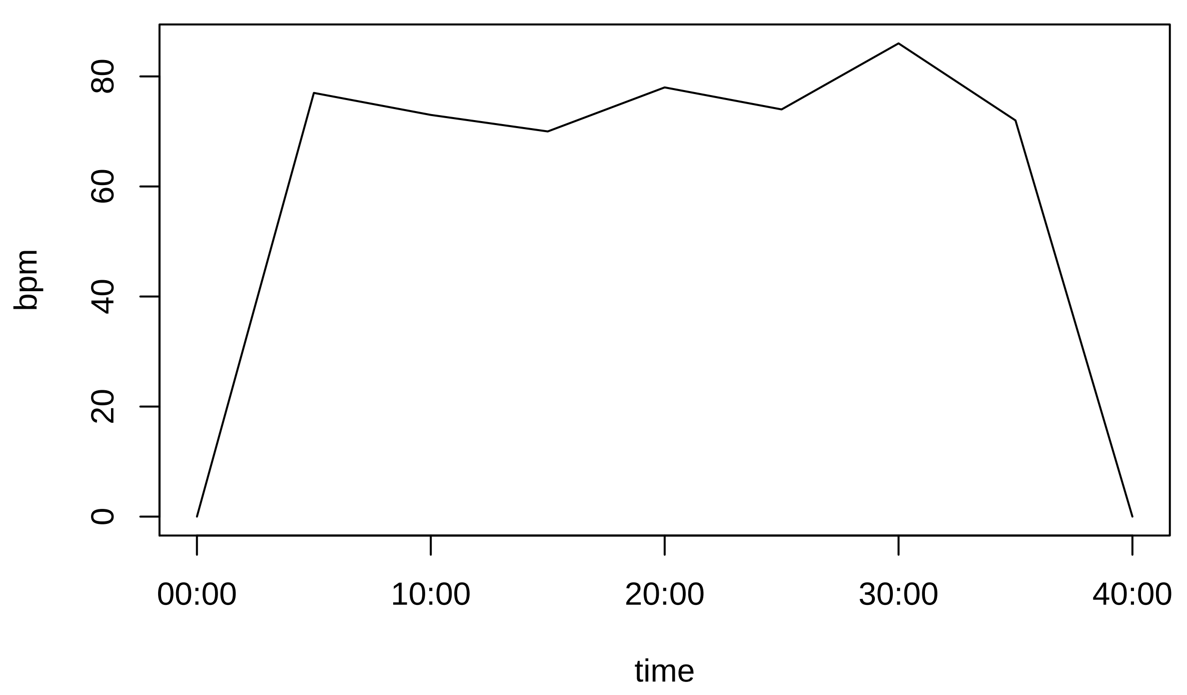
I constructed the login cookie for accessing fibit data from cloud and extracted heart rate data (bpm within 5 minute intervals)

```r
cookie = login("penningtonjeff@yahoo.com","wearable10", rememberMe = TRUE)
startdate = as.Date("2016-11-23", format="%Y-%m-%d")
#enddate = as.Date("2016-10-30",format="%Y-%m-%d")
#s= seq(startdate, enddate, by="days")

my.hrdata <- get_intraday_data(cookie,what="heart-rate",date=sprintf("%s",startdate))

#Printing out bpm for the measured time period. Rest all are zeros.
newdata <- my.hrdata[c(144:152),]
newdata
```

```
##     bpm confidence caloriesBurned defaultZone customZone
## 144   0         -1        5.40450          NA         NA
## 145  77          2        5.40450          NA         NA
## 146  73          1        6.37731          NA         NA
## 147  70          1        6.80967          NA         NA
## 148  78          2        6.70158          NA         NA
## 149  74          1        6.91776          NA         NA
## 150  86          1        5.94495          NA         NA
## 151  72          2       15.56496          NA         NA
## 152   0         -1        5.40450          NA         NA
##                dateTime                time
## 144 2016-11-23 12:00:00 2016-11-23 12:00:00
## 145 2016-11-23 12:05:00 2016-11-23 12:05:00
## 146 2016-11-23 12:10:00 2016-11-23 12:10:00
## 147 2016-11-23 12:15:00 2016-11-23 12:15:00
## 148 2016-11-23 12:20:00 2016-11-23 12:20:00
## 149 2016-11-23 12:25:00 2016-11-23 12:25:00
## 150 2016-11-23 12:30:00 2016-11-23 12:30:00
## 151 2016-11-23 12:35:00 2016-11-23 12:35:00
## 152 2016-11-23 12:40:00 2016-11-23 12:40:00
```

```r
#Adding frequency and time variables
my.hrdata$freq <- sapply(my.hrdata$bpm,function(x) x/60)
newdata$seconds <- c("00","00","00","00","00","00","180","480","780")

#Plotting the data
plot( newdata$time,newdata$bpm, type="l",xlab="time",ylab="bpm",main="Heart rate (bpm) after 5 min inte
```

Heart rate (bpm) after 5 min intervals for 9th October 2016



Bitalino

For the Bitalino, I am using the Opensignals software to take measurements using the device. #Bitalino data extraction The Opensignals software produces the data in two formats - h5 and txt. I am showing the method to extract data from both types of files but I will be using the data from txt file for my further analysis. nSeq is the sequence, while I1 -I4 are the inputs. The interesting and useful variable is the raw one.

```
#H5 file
h5f = H5Fopen("./testing.h5")

my.bitdata <- c(h5f$"20:15:10:26:64:88")

bitalino.df <- data.frame(my.bitdata$raw)

#txt file which will be mainly used
my.data <- read.table("demo.txt")
names(my.data) <- c("nSeq","I1","I2","I3","I4","raw")
head(my.data)
```

```
##   nSeq I1 I2 I3 I4 raw
## 1    1  1  1  1  1 568
## 2    2  1  1  1  1 483
## 3    3  1  1  1  1 471
## 4    4  1  1  1  1 513
## 5    5  1  1  1  1 471
## 6    6  1  1  1  1 487
```

```
str(my.data)
```

```
## 'data.frame':    54150 obs. of  6 variables:
##  $ nSeq: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ I1  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ I2  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ I3  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ I4  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ raw : int  568 483 471 513 471 487 446 459 460 446 ...
```

Data conversion/cleaning

The data received from the Bitalino device needs to be converted into the standard units of measurement, which in this case is Volts.I am converting the raw values obtained from bitalino into appropriate values by the formula: ECGV = (ECGB * Vcc / 2^n - Vcc / 2) / GECG

Where: ECGV - ECG value in Volts (V) ECGmV - ECG value in miliVolts (mV) ECGB - ECG value obtained from BITalino Vcc - Operating Voltage (V) n - number of bits (bit) GECG - ECG Sensor Gain

Values: Vcc = 3.3 (V) GECG = 1100 n= 10 (for Channel 3)

```
#Converting to ECG data in V
my.data$ECG <- sapply(my.data$raw,function(x)(x*3.3/2^10 - 3.3/2)/1100)
head(my.data$ECG,)
```

```
## [1]  1.640625e-04 -8.496094e-05 -1.201172e-04  2.929687e-06 -1.201172e-04
## [6] -7.324219e-05
```

```
#Write in file for Kubios software
ecgdatalist <- c(my.data$ECG)
write.table(ecgdatalist,file="ecgdata.txt")
```

Signal processing

I have used a code referred to from the internet to process the ECG data obtained from Bitalino to pass low and high pass filters so as to visualize the data. The sampling rate used was 100Hz. The purpose of this is to only visualization of the data.

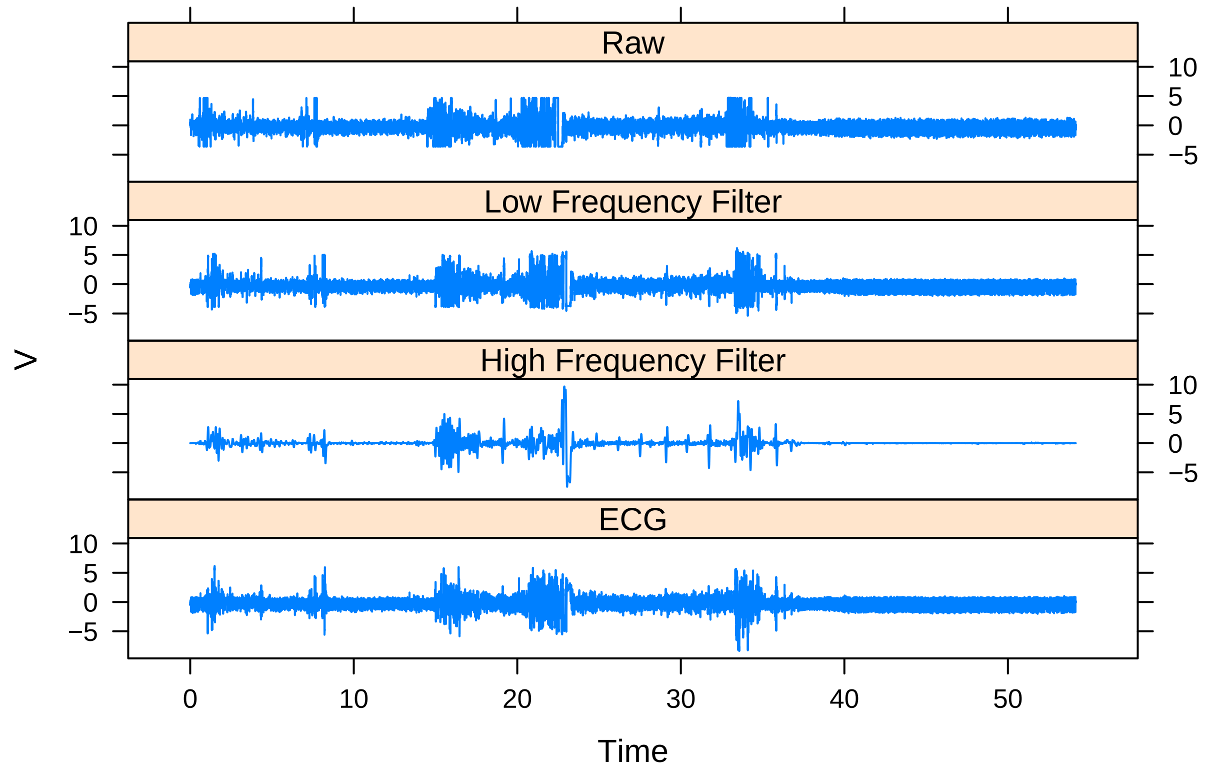
```
#Transform the real and imaginary portions of the
#FFT into magnitude and phase.
amplitude <- function( x ) { sqrt(Re(x)^2+Im(x)^2) }
phase     <- function( x ) { atan(Im(x)/Re(x)) }

#sinc function of frequency f
sinc      <- function( x, f ) { ifelse(x==0, 2*pi*f, sin(2*pi*f*x)/x) }

#Blackman window from 0..m
Blackman  <- function( m ) { 0.42-0.5*cos(2*pi*(0:m)/m)+0.08*cos(4*pi*(0:m)/m) }
```

```r
#Hamming window from 0..m
Hamming   <- function( m ) { 0.54-0.46*cos(2*pi*(0:m)/m) }

#simple low pass filter
#y - vector to filter
#t - time interval between measurements (s)
#f - low pass frequency (Hz)
lpf <- function( y, t, f ) {
  rc <- 1 / ( 2 * pi * f )
  a  <- t / ( t + rc )
  n  <- length( y )
  yf <- y
  for( i in 2:length(y) ) {
    yf[i] <- a * y[i] + (1-a) * yf[i-1]
  }
  return( yf )
}

#windowed sinc low pass filter
#y - vector to filter
#t - time interval between measurements (s)
#f - low pass frequency (Hz)
wlpf <- function( y, t, f ) {
  m  <- min(floor(length(y)/2), 500)
  #generate the sinc kernel
  rk <- sinc(-m:m, f*t)
  #apply the Blackman window
  bk <- Blackman(2*m) * rk
  #pad the filter with zeros
  k  <- c(bk, rep(0,length(y)-length(bk)))
  #convolve y with the filter kernel
  fy  <- fft(fft(k)*fft(y), inverse=TRUE)
  return(Re(fy))
}

dat  <- scan(file="ECG_converted.csv")
dat  <- (dat - mean(dat)) / sd(dat)

#filter high frequency noise
fdat <- wlpf(dat, 1/100, 30)
fdat <- (fdat-mean(fdat))/sd(fdat)

#filter low frequency noise
rdat <- wlpf(dat, 1/100, 1)
rdat <- (rdat-mean(rdat))/sd(rdat)

#use low pass filter in ecg signal
edat <- fdat - rdat

require(lattice)
```

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 3.3.2
```

```
xplot <- rep((0:(length(dat)-1))/1000,4)
yplot <- c(dat,fdat,rdat,edat)
gplot <- c(rep("Raw",length(dat)),
           rep("Low Frequency Filter",length(dat)),
           rep("High Frequency Filter",length(dat)),
           rep("ECG",length(dat)))

tp <- xyplot(yplot~xplot|gplot,type="l",layout=c(1,4), xlab="Time", ylab="V")

#uncomment the following to save an image
#trellis.device(png, file="ecgfilter.png", height=750, width=750)
print(tp)
```



```
#dev.off()
```

Kubios

Kubios is a software that is used to calculate heart rate variation. I inputted the ecgdata.txt file into the software to convert the ECG data into RR intervals.

Convert Bitalino data to RR interval - Kubios

```
hrv.kubios <- read.csv("ecg_hrv.csv")
names(hrv.kubios) <- c("Time","RR Interval","FFT Frequency","PSD","AR Frequency","PSD")
head(hrv.kubios)
```

```
##    Time RR Interval FFT Frequency      PSD AR Frequency      PSD
## 1 1.257       0.774         0.000 137015.2        0.000 103867.3
## 2 2.047       0.791         0.004 304276.5        0.004 207235.8
## 3 2.814       0.767         0.008 379153.5        0.008 205756.2
## 4 3.541       0.726         0.012 257172.4        0.012 203343.8
## 5 4.247       0.707         0.016 352241.2        0.016 200075.0
## 6 4.924       0.676         0.020 304595.4        0.020 196047.7
```

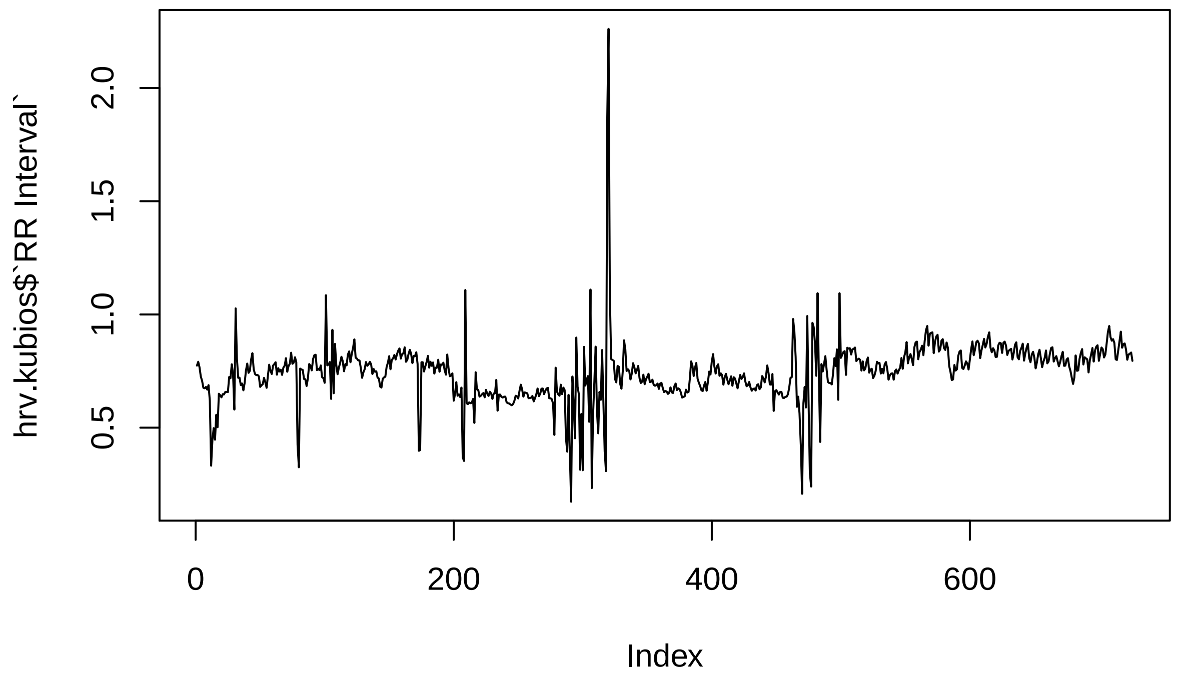
Data cleaning and conversion

The main problem I am currently addressing is the convertion of the mV data from Bitalino to beats per minute.

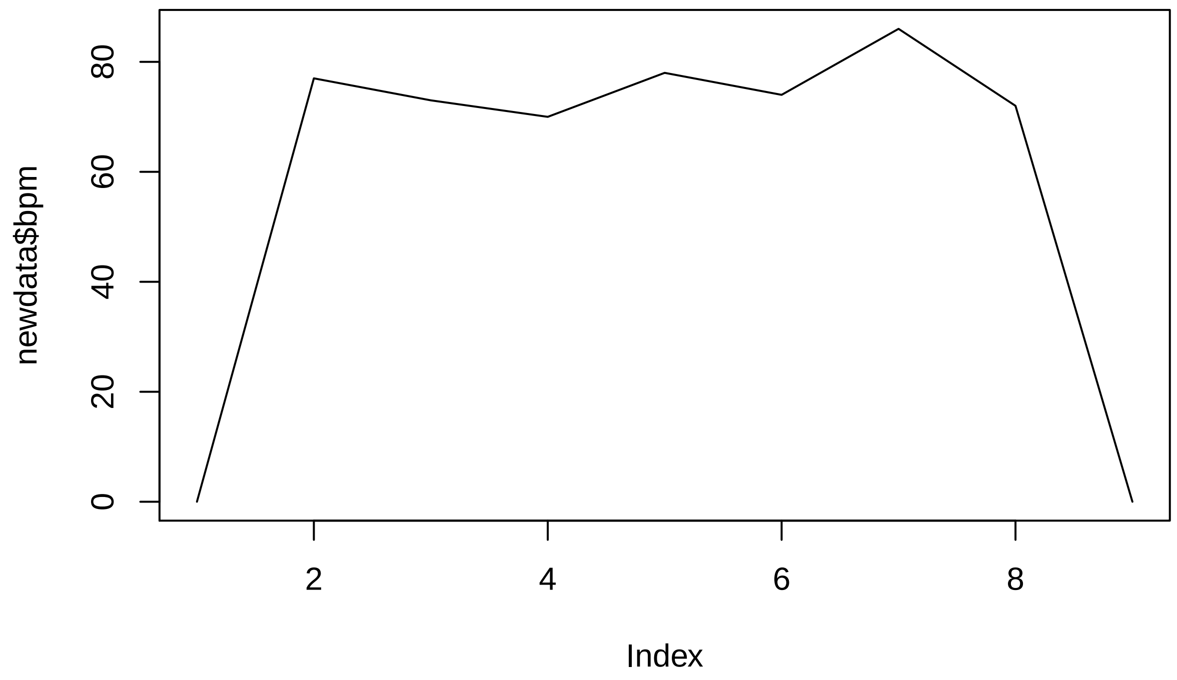
```
#Episode one - sitting for 3 minutes = 180 seconds
B1 <- 60/(mean(hrv.kubios[1:249,2])) #Bitalino
F1 <- newdata$bpm[6] #time=12:25

#Episode two - walking for 3 minutes = 180 seconds
B2 <- 60/(mean(hrv.kubios[250:506,2]))
F2 <- newdata$bpm[7] #time=12:30

#Episode three - sitting for 3 minutes = 180 seconds after walking
B3 <- 60/(mean(hrv.kubios[507:726,2]))
F3 <- newdata$bpm[8] #time=12:35


bitalino <- c(B1, B2, B3)
fitbit <- c(F1, F2, F3)
```

The data can be visualized to check overall trend #visualization

```
plot(hrv.kubios$`RR Interval`,type="l")
```



```r
plot(newdata$bpm, type = "l")
```

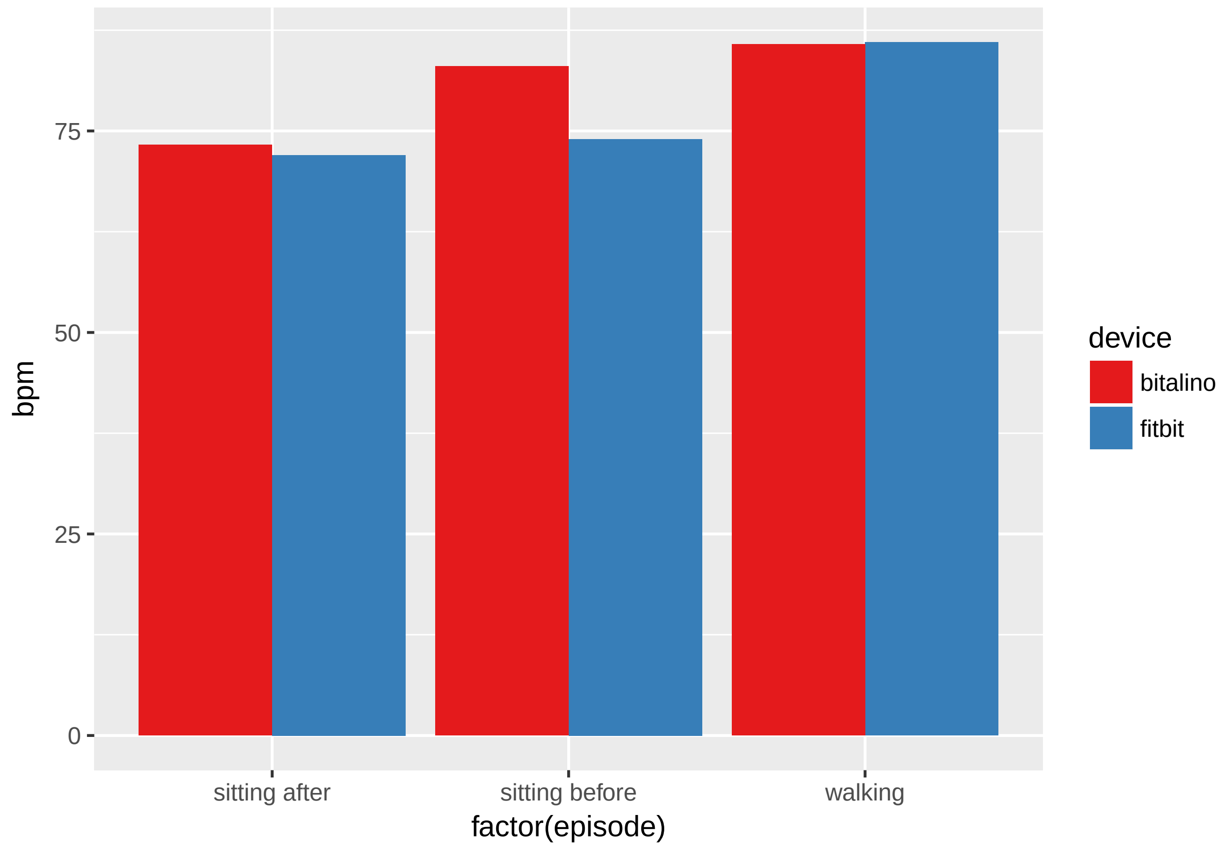


Results

The results of the data can be first visualized by histograms:

Histograms

```r
total <- data.frame(c(fitbit,bitalino))
names(total)<- c("bpm")
total$episode <- c("sitting before","walking","sitting after")
total$device <- c("fitbit","fitbit","fitbit","bitalino","bitalino","bitalino")

ggplot(total, aes(factor(episode),bpm, fill = device)) +
  geom_bar(stat="identity", position = "dodge") +
  scale_fill_brewer(palette = "Set1")
```



Box plots

T-test

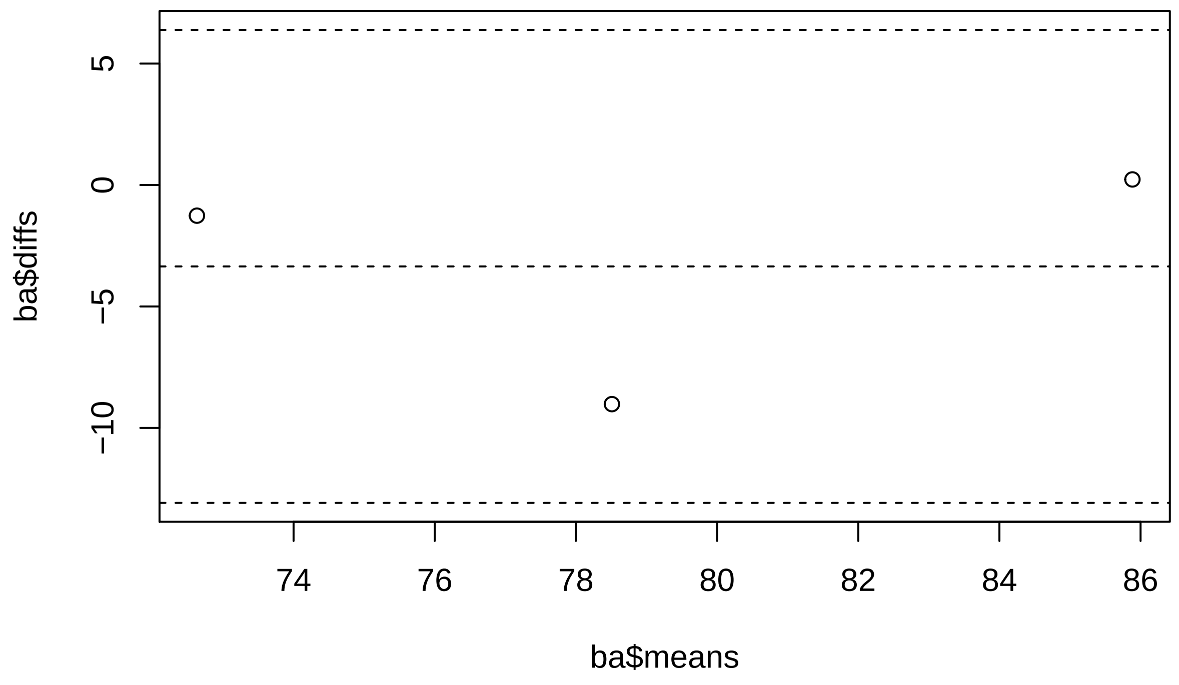
The data can be analysed by performing t-test

```
t.test(bitalino,fitbit)
```

```
##
##  Welch Two Sample t-test
##
## data:  bitalino and fitbit
## t = 0.57885, df = 3.9226, p-value = 0.5943
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -12.84797  19.54992
## sample estimates:
## mean of x mean of y
##  80.68431  77.33333
```

The converted results can be analysed by plotting BlandAltman plots to check the accuracy of the measurements. #BlandAltman Plots

```r
bland.altman.plot(fitbit, bitalino)
```



```
## NULL
```

```r
#bland.altman.plot(newdata$bpm, my.data$ECG)
```

We can check for correlation between the two measurements. Other statistics calculated are as follows: 726 287 #correlation

```r
#cor()
```