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 with 2 samples. The first sample is heart rate measured 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 second sample is heart rate measured over the course of an hour for use in accuracy testing. The data extracted was filtered and converted to appropriate units for comparison.

The statistical model involves quantification of the second sample by Bland-Altman plots, t-test calculation and correlation coefficient. The first sample variation in heart rate over the different episodes is visualized by plots and histograms to get an idea of the heart rate variation. 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. I have 2 samples - 1)

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

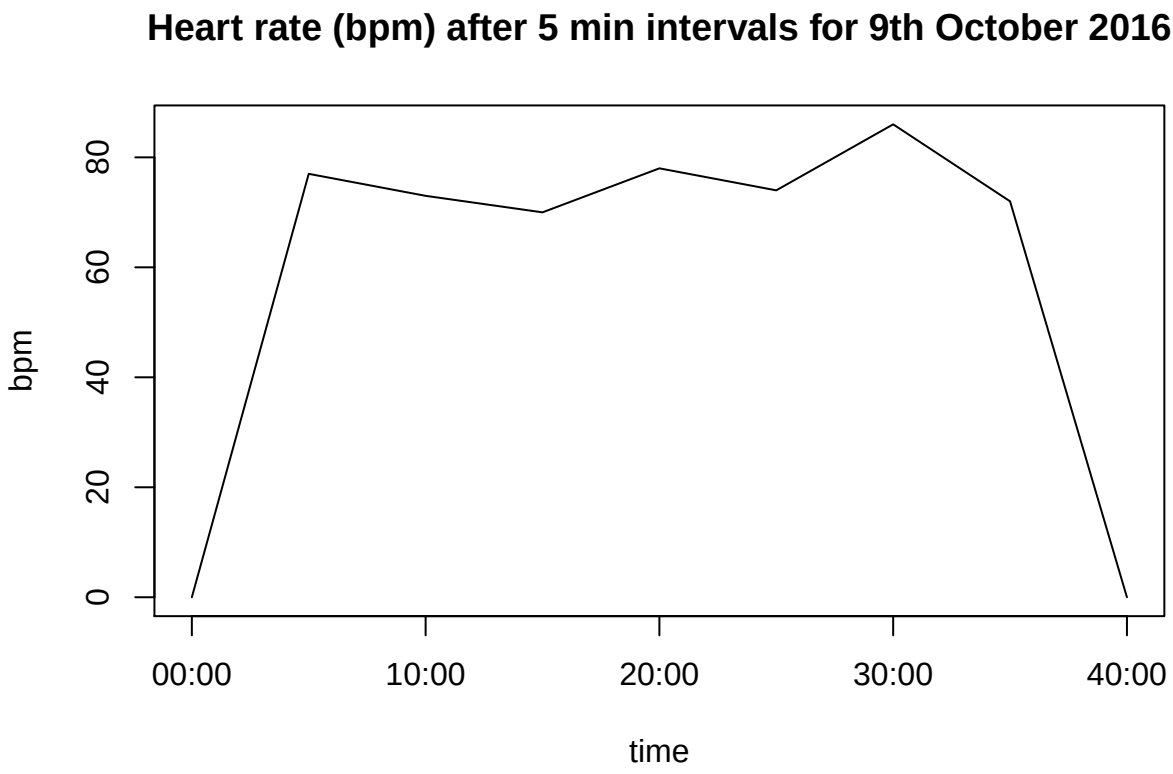
Sample1

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

```
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
```

```
#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
```



#Sample2 A separate sample of 1 hour of heart rate measurement taken during a weekend doing regular activity like sitting, standing, lifting things and typing on the laptop was taken along with a simultaneous bitalino sample in order to check accuracy of measurement as sample1 had too less data points for a statistical analysis.

FitBit data- To get data from the 1 hour sample from fitbit cloud

```
cookie = login("anupamalur@gmail.com","anupam123", rememberMe = TRUE)
startdate = as.Date("2016-12-02", format="%Y-%m-%d")
#enddate = as.Date("2016-10-30",format="%Y-%m-%d")
#s= seq(startdate, enddate, by="days")

my.hrdata2 <- get_intraday_data(cookie,what="heart-rate",date=sprintf("%s",startdate))
fitdata <- my.hrdata2[219:230,]
```

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.

Sample 1

```
#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("sample1.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 ...
```

Sample 2

Bitalino data- To get the same 1 hour sample measured with Bitalino from the txt file generated

```
bitdata <- read.table("sample2.txt")
names(bitdata) <- c("nSeq","I1","I2","I3","I4","raw")
head(bitdata)
```

```
##   nSeq I1 I2 I3 I4 raw
## 1    0  1  1  1  1 444
## 2    1  1  1  1  1 448
## 3    2  1  1  1  1 444
## 4    3  1  1  1  1 448
## 5    4  1  1  1  1 448
## 6    5  1  1  1  1 444
```

```r
str(bitdata)
```

```
## 'data.frame':    360315 obs. of  6 variables:
##  $ nSeq: int  0 1 2 3 4 5 6 7 8 9 ...
##  $ 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  444 448 444 448 448 444 448 445 448 446 ...
```

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) #Sample 1

```r
#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
```

```r
#Write in file for Kubios software
ecgdatalist1 <- c(my.data$ECG)
write.table(ecgdatalist1,file="ecgdata1.txt")
#I manually removed the first column from this file
```

Sample 2

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

```
## [1] -0.0001992188 -0.0001875000 -0.0001992188 -0.0001875000 -0.0001875000
## [6] -0.0001992188
```

```r
#Write in file for Kubios software
ecgdatalist2 <- c(bitdata$ECG)
write.table(ecgdatalist2,file="ecgdata2.txt")
#I manually removed the first column from this file
```

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.

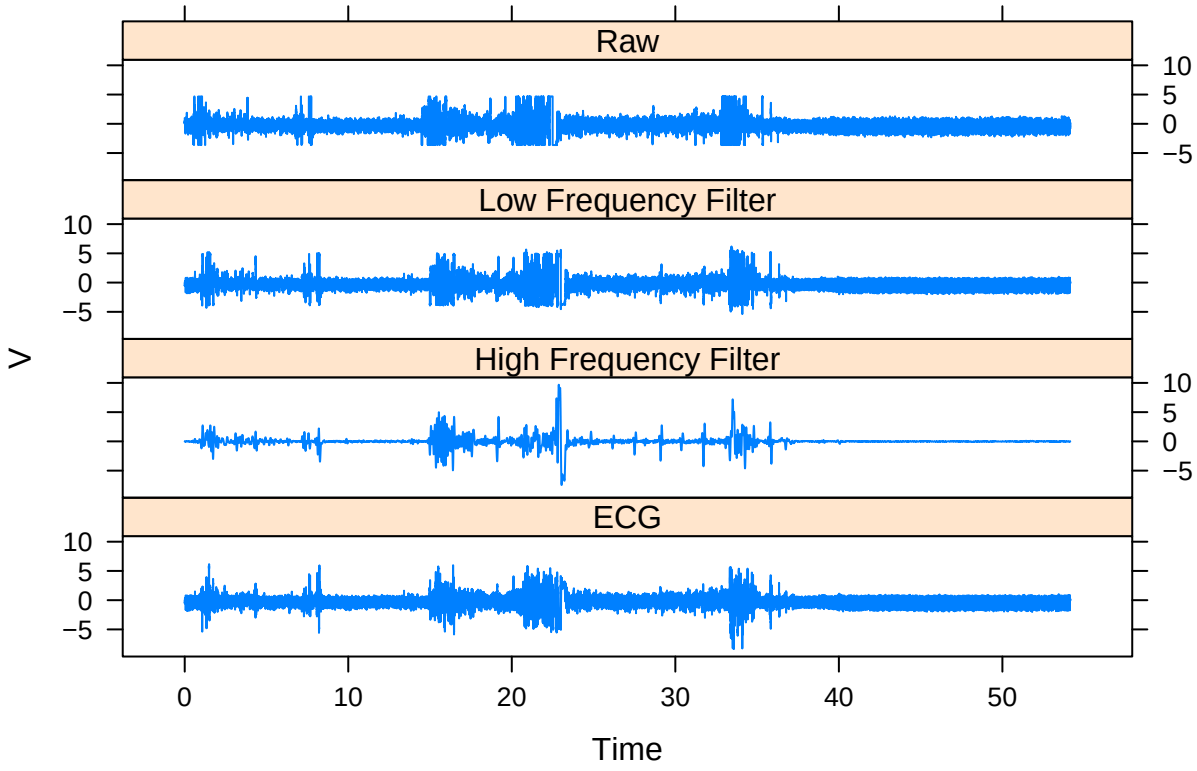
```
#Transform the real and imaginary portions of the
#FFT into magnitude and phase.
amplitude <- function( x ) { sqrt(Re(x)^2+Im(x)^2) }
phase     <- function( x ) { atan(Im(x)/Re(x)) }

#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) }

#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)
```

```r
#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
```

```r
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 using Kubios- opening the converted files

Sample1

```
hrv1.kubios <- read.csv("ecg_hrv.csv")
names(hrv1.kubios) <- c("Time","RR Interval","FFT Frequency","PSD","AR Frequency","PSD")
head(hrv1.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
```

Sample 2

```
hrv2.kubios <- read.csv("ecgdata2.csv")
names(hrv2.kubios) <- c("Time","RR Interval","FFT Frequency","PSD","AR Frequency","PSD")
head(hrv2.kubios)
```

```
##    Time RR Interval FFT Frequency       PSD AR Frequency      PSD
## 1 1.038       0.738         0.000 146670.29        0.000 42284.50
## 2 1.841       0.803         0.004 168941.70        0.004 83908.63
## 3 2.666       0.825         0.008  82098.87        0.008 81989.45
## 4 3.471       0.806         0.012  69517.98        0.012 78983.23
## 5 4.270       0.798         0.016  71810.11        0.016 75135.28
## 6 5.005       0.735         0.020  61683.29        0.020 70719.15
```

Data cleaning and conversion

The RR interval data has to be converted into beats per minute. The formula for conversion is RR interval for 1 minute = 1/bpm. As the RR is in seconds I am converting it to minutes and then inversing it to get the beats per minute value.

change indexes for mean calculation

Sample 1

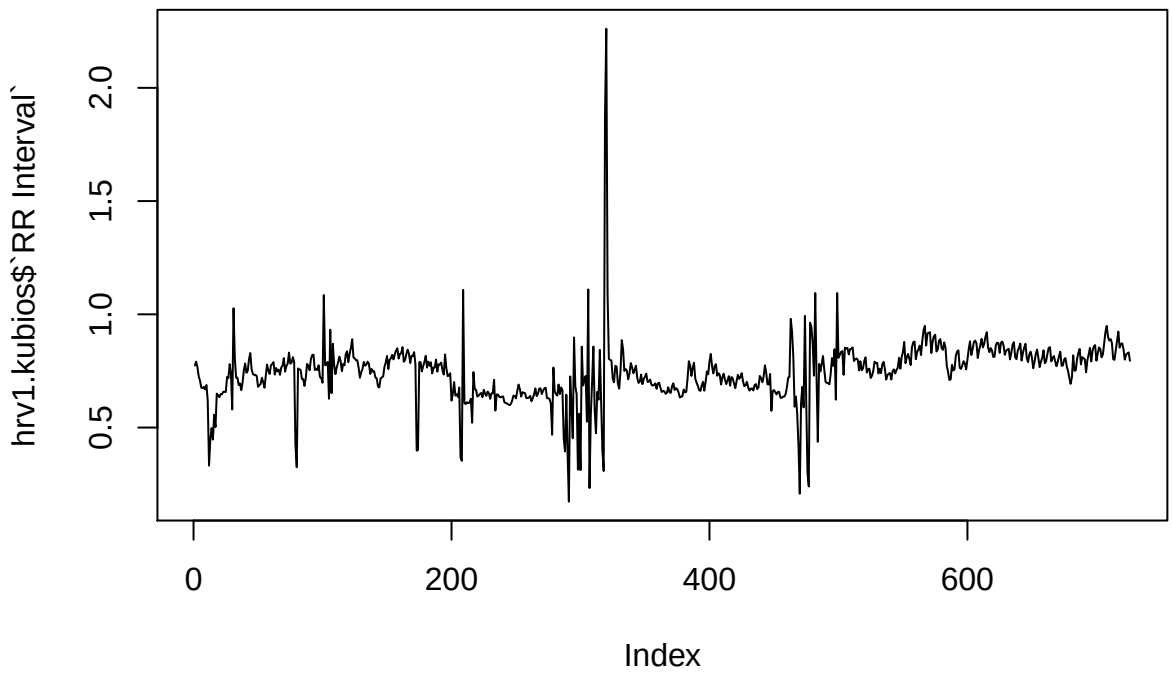
```
#Episode one - sitting for 3 minutes = 180 seconds
B1 <- 60/(mean(hrv1.kubios[1:249,2])) #Bitalino
F1 <- newdata$bpm[6] #time=12:25

#Episode two - walking for 3 minutes = 180 seconds
B2 <- 60/(mean(hrv1.kubios[250:506,2]))
F2 <- newdata$bpm[7] #time=12:30

#Episode three - sitting for 3 minutes = 180 seconds after walking
B3 <- 60/(mean(hrv1.kubios[507:726,2]))
F3 <- newdata$bpm[8] #time=12:35


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

Sample 2

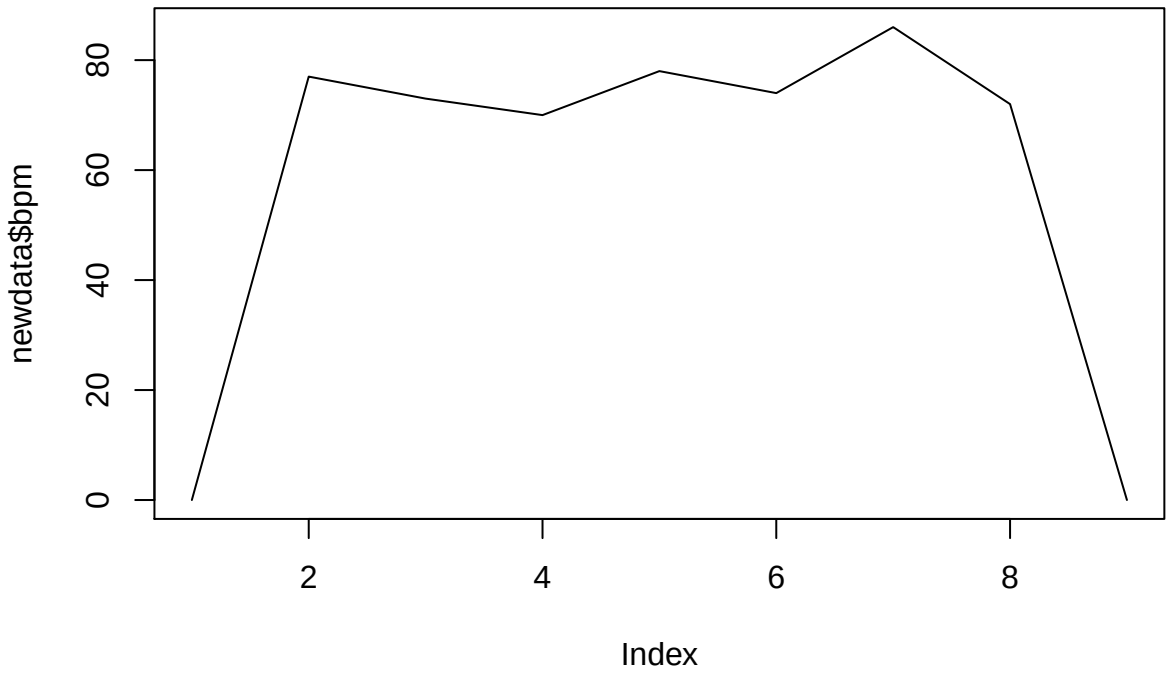
```
#Calculating for every 5 minutes - thus, calculating 12 data points
a <- 60/mean((hrv2.kubios[314:394,2])) #18:15 - 240 to 300 seconds
b <- 60/mean((hrv2.kubios[716:795,2]))  #18:20 - 540 to 600 seconds
c <- 60/mean((hrv2.kubios[1091:1164,2]))  #18:25 - 840 to 900 seconds
d <- 60/mean((hrv2.kubios[1443:1521,2]))  #18:30 - 1140 to 1200 seconds
e <- 60/mean((hrv2.kubios[1839:1893,2]))  #18:35 - 1460 to 1500 seconds
f <- 60/mean((hrv2.kubios[2181:2256,2]))  #18:40 - 1740 to 1800 seconds
g <- 60/mean((hrv2.kubios[2548:2620,2]))  #18:45 - 2040 to 2100 seconds
h <- 60/mean((hrv2.kubios[2917:2989,2]))  #18:50 - 2340 to 2400 seconds
i <- 60/mean((hrv2.kubios[3280:3354,2]))  #18:55 - 2640 to 2700 seconds
j <- 60/mean((hrv2.kubios[3654:3727,2]))  #19:00 - 2940 to 3000 seconds
k <- 60/mean((hrv2.kubios[4022:4097,2]))  #19:05 - 3240 to 3300 seconds
l <- 60/mean((hrv2.kubios[4401:4476,2]))  #19:10 - 3540 to 3600 seconds

bitalino2 <- c(a,b,c,d,e,f,g,h,i,j,k,l)
fitbit2 <- c(fitdata$bpm)
```

The data can be visualized to check overall trend #visualization

```
plot(hrv1.kubios$`RR Interval`,type="l")
```



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



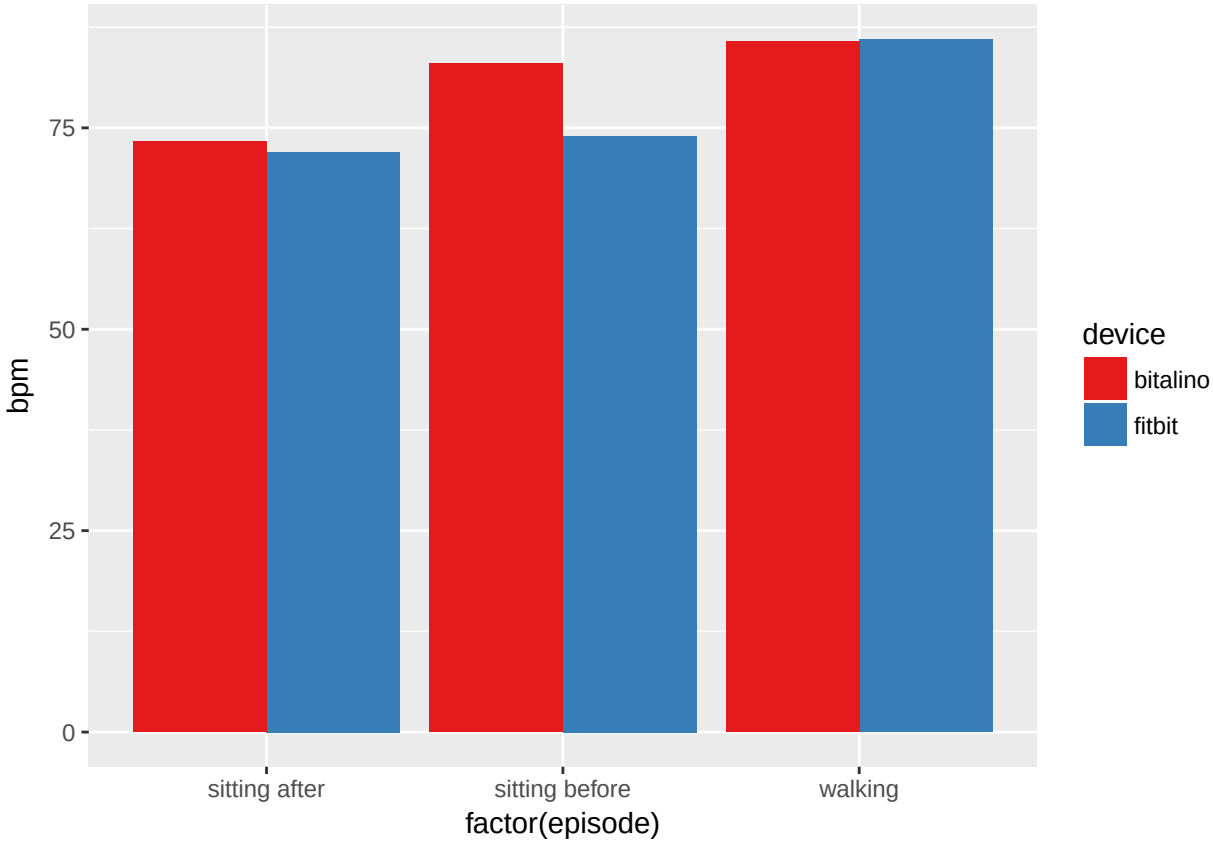
Results

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

Histograms

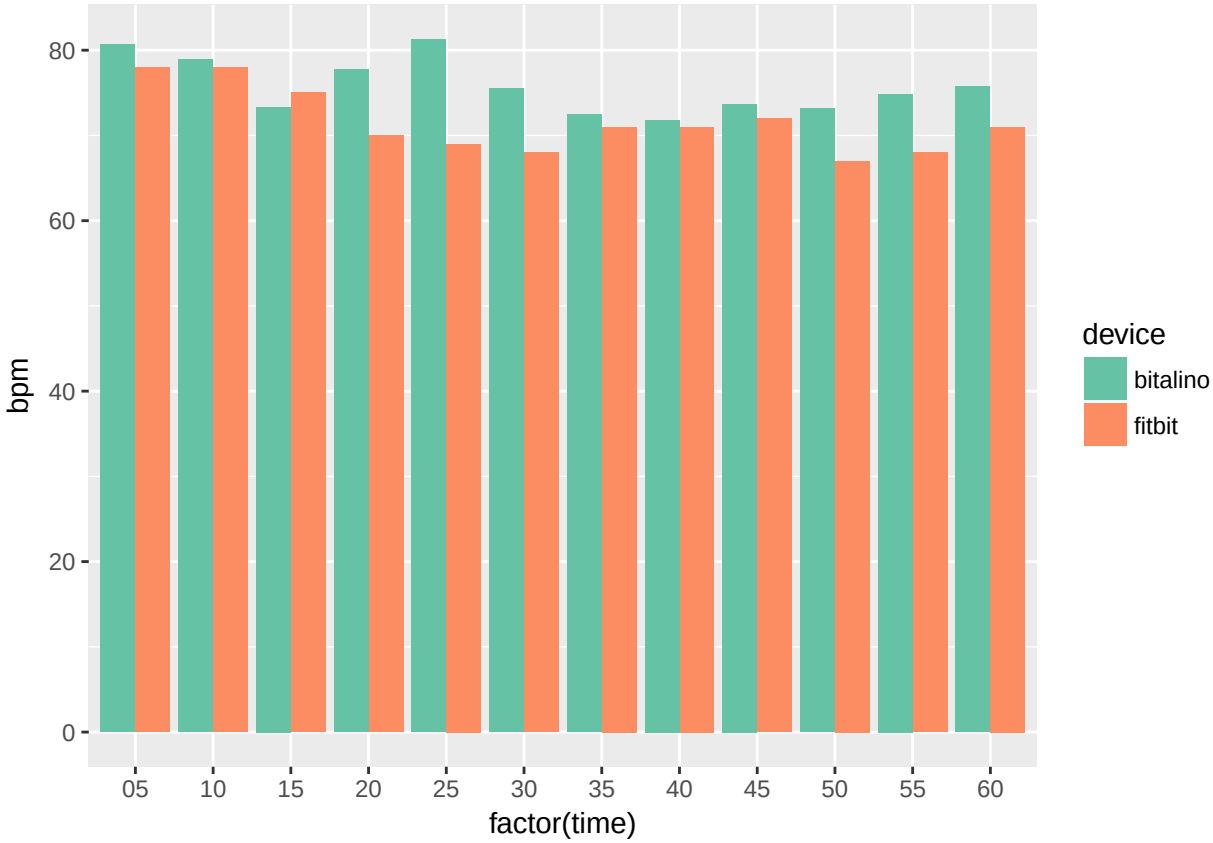
Sample 1

```
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")
```



Sample 2

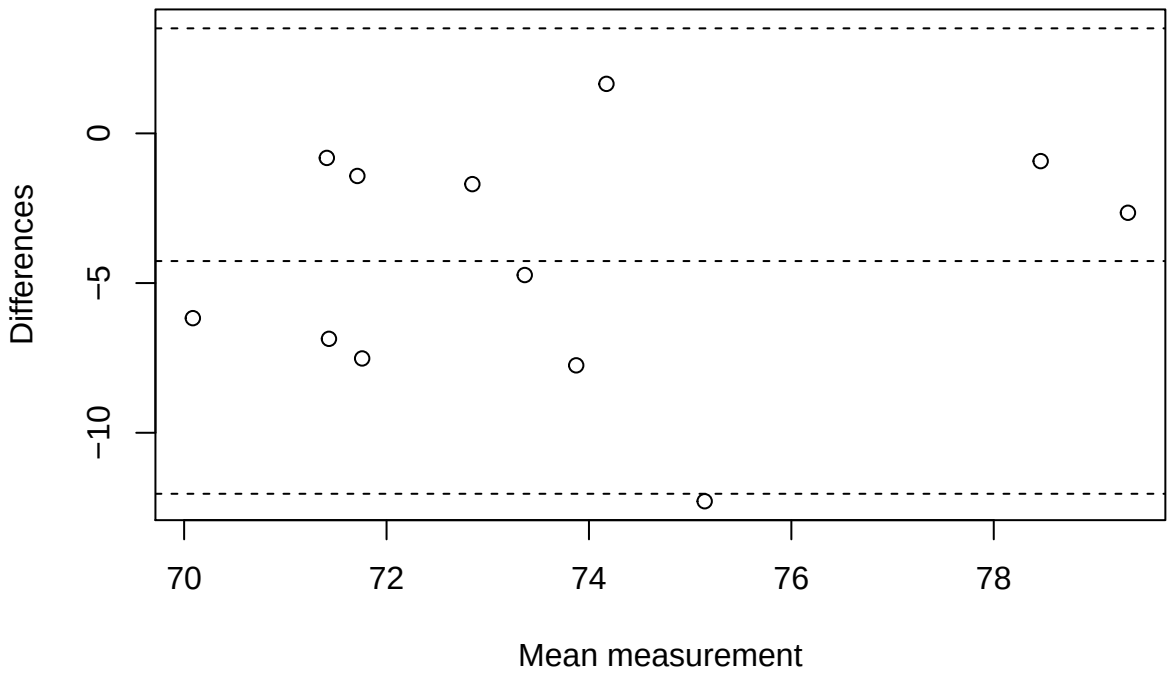
```
total2 <- data.frame(c(fitbit2,bitalino2))
names(total2)<- c("bpm")
total2$time <- c("05","10","15","20","25","30","35","40","45","50","55","60")
total2$device <- c("fitbit","fitbit","fitbit","fitbit","fitbit","fitbit","fitbit","fitbit","fitbit","fi
ggplot(total2, aes(factor(time),bpm, fill = device)) +
  geom_bar(stat="identity", position = "dodge") +
  scale_fill_brewer(palette = "Set2")
```



BlandAltman Plots

The converted results can be analysed by plotting BlandAltman plots to check the accuracy of the measurements.The majority of the points are within the limits of agreement (the 2 dotted lines at the end representing +1.96 SD and -1.96 SD from top to bottom. The middle dotted line is the mean difference between the two methods.)

```
bland.altman.plot(fitbit2, bitalino2, xlab = "Mean measurement", ylab= "Differences")
```




```
## NULL
```

T-test

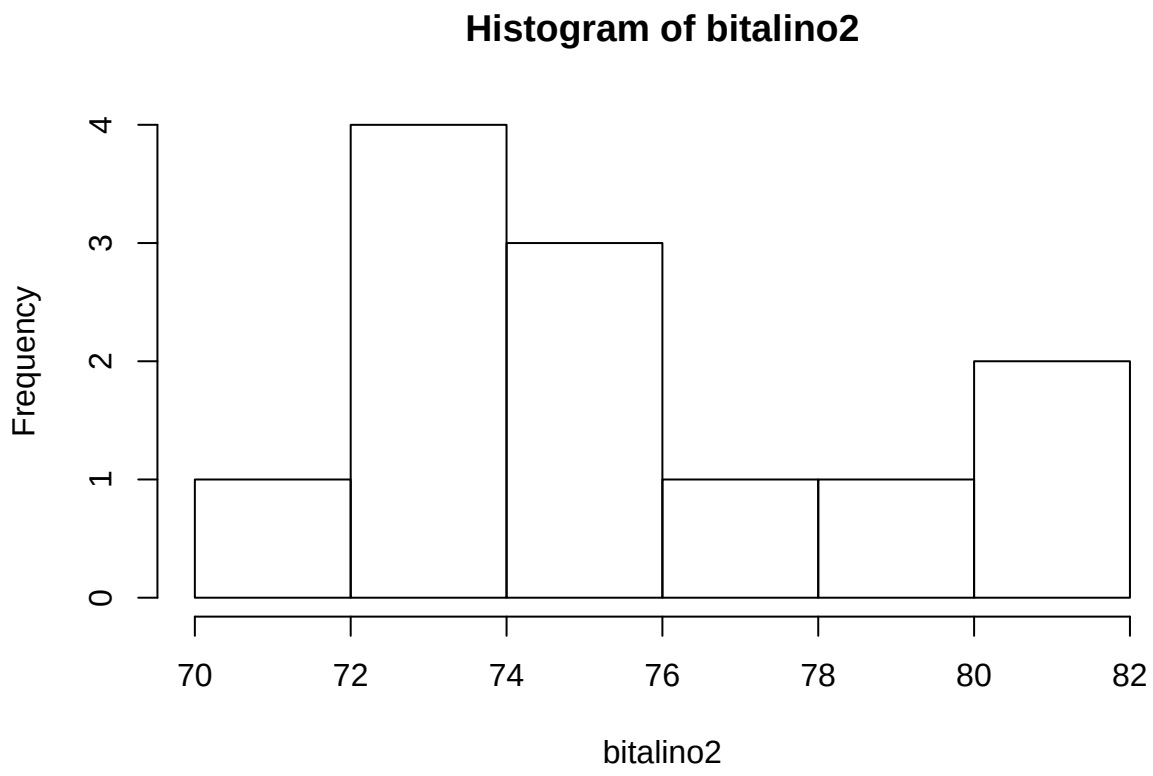
The data can be analysed by performing t-test. The p-value is 0.006337 thus, significant.

```
t.test(bitalino2,fitbit2,var.equal = TRUE)
```

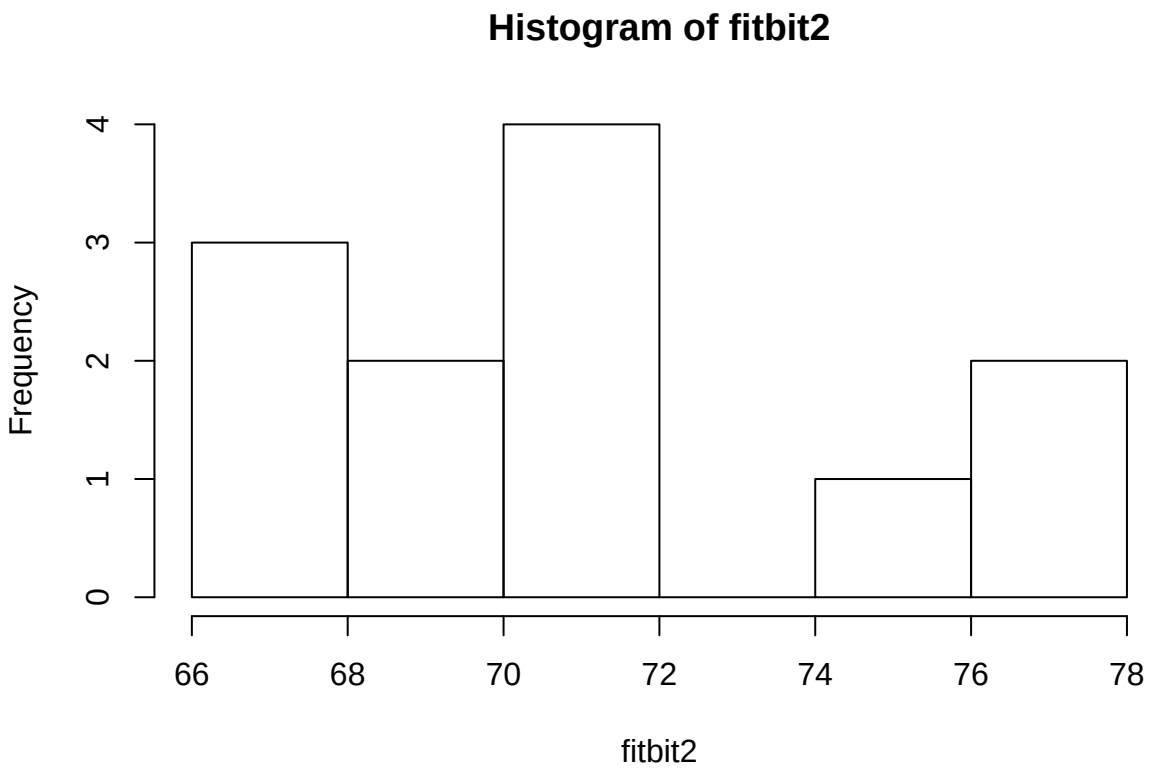
```
##
##  Two Sample t-test
##
## data:  bitalino2 and fitbit2
## t = 3.0172, df = 22, p-value = 0.006337
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  1.333465 7.196704
## sample estimates:
## mean of x mean of y
##  75.76508  71.50000
```

correlation for sample 2 gives a value of 0.1766884

```
hist(bitalino2)
```




```
hist(fitbit2)
```




```
#The data is not normal so I am using Spearman correlation
cor(fitbit2, bitalino2, use="complete.obs", method="spearman")
```

```
## [1] 0.1766884
```

Final Result

Sample 1

By viewing the histogram you can see rise in the heart rate during walking. The first reading for bitalino shows some variation which could be due to the noise in the sample collected. This noise may be due to the position of the electrodes (on palms instead of chest).

Sample 2

The results of the Bland Altman plot shows a proportional error trend with more variability for mid-range heart rate values and less for tail ends.All of the data points fall within the limits of agreement which proves that there is considerable accuracy for heart rate measurement using Fitbit. The t-test gives a p-value of 0.006337. Thus, as the p-value is significant, we reject the null hypothesis. The correlation coefficient gives us a value of 0.1766884.

The HRV can be also used to determine outcome of psychological conditions like Autism and other behavioural conditions. It can also be used to detect the outcome of a bacterial or behavioural abnormality before the patient actually shows indications of it.