HRV analysis

Janice Tjeng

2017-11-21

# Load packages

library(RMySQL)  
library(tidyverse)  
library(lubridate)  
library(robustbase)  
library(stringr)  
library(Rssa)

# Read data from E4, firstbeat, & MSband

db <- dbConnect(MySQL(), user="deepresearcher", password="ctpc177!", dbname="deephealth2", host="deephealthlab.org")  
e4 <- dbSendQuery(db,"SELECT \* FROM viewl\_e4\_rr")  
e4dat <- fetch(e4,n=-1)   
fb <- dbSendQuery(db, "SELECT \* FROM viewl\_firstbeat\_rr")  
fbdat <- fetch(fb, n=-1)  
mb <- dbSendQuery(db, "SELECT \* FROM viewl\_msband\_rr")  
mbdat <- fetch(mb, n=-1)

# Read timing data

timing <- read\_csv("timing\_data.csv")  
df <- timing %>%  
 select (Event, LWP2\_0019, Activity) %>%  
 rename("Time"="LWP2\_0019")  
n <- nrow(df)  
dte <- mdy(as.character(df[1,"Time"]))  
start <- as.character(df[2,"Time"])  
end <- as.character(df[n,"Time"])

# Select rows for user 19 from E4

user19\_l <- e4dat %>%  
 filter(user=="lwp2\_0019", device\_location=="Left") %>%   
 mutate(readable\_timestamp=as.POSIXct(readable\_timestamp),  
 e4\_rr=e4\_rr\*1000)   
filter\_time\_day1\_e4 <- user19\_l %>%  
 filter(strftime(readable\_timestamp, "%H:%M:%S") >= start & strftime(readable\_timestamp, "%H:%M:%S") <= end, date(readable\_timestamp)==dte)

# Select rows for user 19 from ECG

user19\_fb <- fbdat %>%  
 filter(user=="lwp2\_0019") %>%   
 mutate(readable\_timestamp=as.POSIXct(readable\_timestamp))   
filter\_time\_day1\_fb <- user19\_fb %>%  
 filter(strftime(readable\_timestamp, "%H:%M:%S") >= start & strftime(readable\_timestamp, "%H:%M:%S") <= end, date(readable\_timestamp)==dte)

# Select rows for user 19 from MSband

user19\_l\_mb <- mbdat %>%  
 filter(user=="lwp2\_0019", device\_location=="Left") %>%   
 mutate(readable\_timestamp=as.POSIXct(readable\_timestamp),  
 mb\_rr=(mb\_rr\*1000))   
filter\_time\_day1\_mb <- user19\_l\_mb %>%  
 filter(strftime(readable\_timestamp, "%H:%M:%S") >= start & strftime(readable\_timestamp, "%H:%M:%S") <= end, date(readable\_timestamp)==dte)

# Clean data

## Detect outlier, short RR-interval, based on z-score standardization.

# Standardize function, standard deviation based on median absolute deviation, a robust estimate of standard deviation  
standardize <- function(rr){  
 return ((rr-mean(rr))/sqrt(covMcd(rr)$cov[1])) # use robust covariance estimate  
}  
  
# Function to identify outlier  
outlier\_short <- function(rr){  
 z <- standardize(rr)  
 ind <- which(z<(-3)) # -3: limit in sd for outlier detection of short rr interval  
 return (ind)  
}  
  
# Function to remove outlier, then add to subsequent beat  
rm\_short <- function(x,y){ # x is the index of outliers, y is the column of rr-interval  
 for (i in 1:length(x)){  
 if (x[i]<length(y)){  
 y[x[i]+1] <- y[x[i]+1] + y[x[i]]  
 y[x[i]] <- NA  
 }  
 else{ # if rr-interval is at the end of the array, just remove it without adding it to the next beat  
 y[x[i]] <- NA  
 }  
 }  
 return (y)  
}  
  
# Combine all functions  
rm\_short\_comp <- function (rr){  
 indx <- outlier\_short(rr)  
 clean <- rm\_short(indx,rr)  
 return (clean)  
}

## Impute long rr-interval, applied after removing short

impute\_long <- function(rr){   
 dmax <- mean(rr) + (2\*sqrt(covMcd(rr)$cov[1]))  
 dp <- numeric()  
 dp1 <- numeric()  
 ind\_long <- which(rr>dmax) # Index of long outliers  
 for (i in 1:length(ind\_long)){ # imputation if rr>dmax  
 dp <- sum((rr[ind\_long][i]-dmax), dp)  
 }  
 dp <- dp/length(ind\_long) # imputation if rr<=dmax  
 rr[ind\_long] <- rr[ind\_long]-dp  
 for (i in 1:length(rr[-ind\_long])){  
 dp1 <- sum((rr[-ind\_long][i]-dmax), dp1)  
 }  
 dp1 <- dp1/length(rr[-ind\_long])  
 rr[-ind\_long] <- rr[-ind\_long]-dp1  
 return (rr)  
}

<https://www.r-bloggers.com/wheres-the-magic-emd-and-ssa-in-r/>

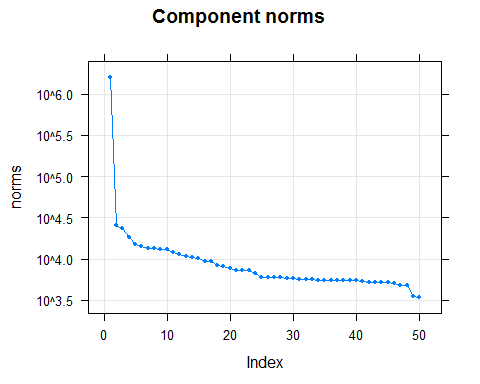
<https://www.researchgate.net/publication/228092069_Basic_Singular_Spectrum_Analysis_and_Forecasting_with_R>

w-correlation matrix **E4**

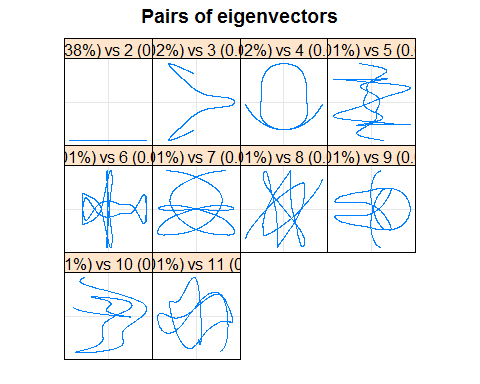
filter\_time\_day1\_e4.clean <- filter\_time\_day1\_e4 %>%  
 mutate(e4\_rr=rm\_short\_comp(e4\_rr)) %>%  
 filter(!is.na(e4\_rr)) %>%  
 mutate(e4\_rr=impute\_long(e4\_rr))  
# 1st stage decomposition  
e4.ssa <- ssa(filter\_time\_day1\_e4.clean$e4\_rr, kind="toeplitz-ssa", svd.method = "svd")  
summary(e4.ssa)

##   
## Call:  
## ssa(x = filter\_time\_day1\_e4.clean$e4\_rr, kind = "toeplitz-ssa", svd.method = "svd")  
##   
## Series length: 3860, Window length: 1930, SVD method: svd  
## Special triples: 0  
##   
## Computed:  
## Eigenvalues: 50, Eigenvectors: 50, Factor vectors: 50  
##   
## Precached: 0 elementary series (0 MiB)  
##   
## Overall memory consumption (estimate): 1.505 MiB

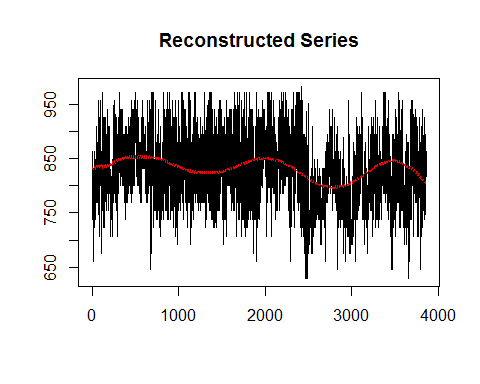
# 1st stage visual information for grouping  
plot(e4.ssa) # scree plot of singular values to identify trends and pairings. Trend at 1&4,



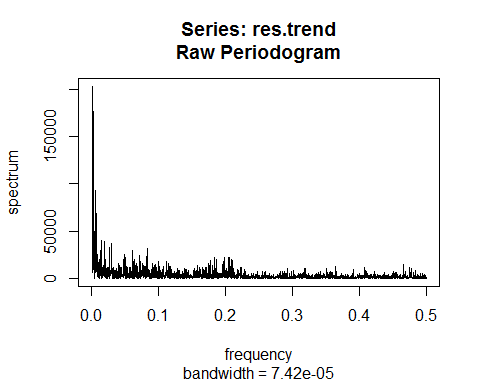
# plot(e4.ssa, "series", groups=as.list(1:5)) # Reconstructed time series for first few components/projections  
# 1st stage reconstruction  
plot(e4.ssa, type="paired") #3&4, 6&7



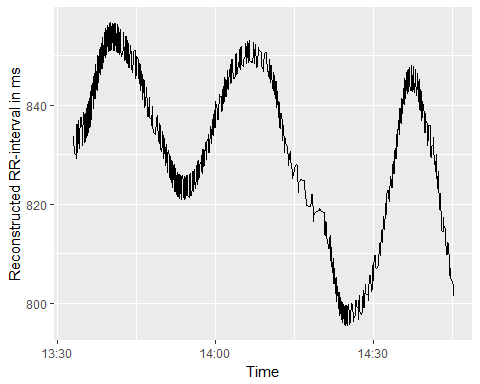
res1 <- reconstruct(e4.ssa, groups=list(Tr=c(1,3,4,48)))  
plot(res1, add.residuals=F) # Reconstructed series of the trend



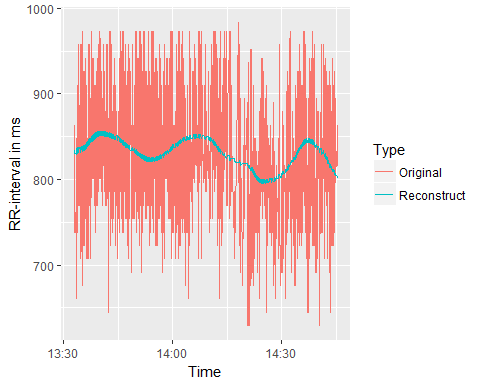
res.trend <- residuals(res1) # Extract seasonality from residuals, original values-predicted values  
spec.pgram(res.trend, detrend=F, log="no") # Periodogram, detect seasonality, should correspond to sine waves in paired plot



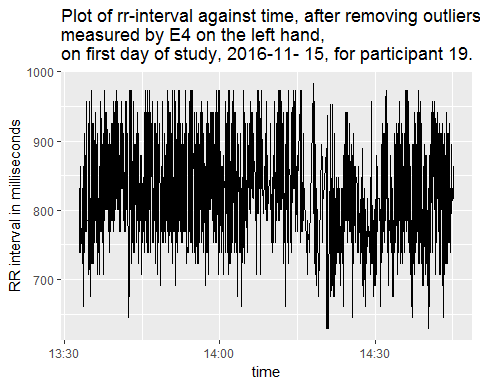
# Reconstructed plot  
df\_e4 <- data.frame(Reconstruct=res1$Tr, Original=filter\_time\_day1\_e4.clean$e4\_rr, Time=filter\_time\_day1\_e4.clean$readable\_timestamp)  
ggplot(df\_e4, aes(x=Time, y=Reconstruct)) + geom\_line() + ylab("Reconstructed RR-interval in ms")



# Reconstructed and original plot combined  
df\_e4 %>%  
 gather(Reconstruct, Original, key="Type", value="RR-interval") %>%  
 ggplot(aes(x=Time, y=`RR-interval`)) +  
 geom\_line(aes(colour=Type)) +  
 ylab("RR-interval in ms")

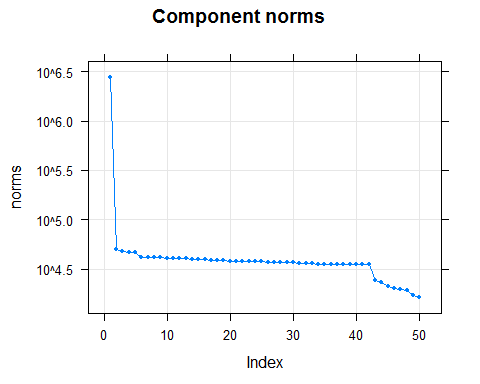


# Original plot  
ggplot(filter\_time\_day1\_e4.clean, aes(x=readable\_timestamp, y=e4\_rr)) + geom\_line() +  
 ggtitle("Plot of rr-interval against time, after removing outliers, \nmeasured by E4 on the left hand, \non first day of study, 2016-11- 15, for participant 19.") +  
 ylab("RR interval in milliseconds") +   
 xlab("time")

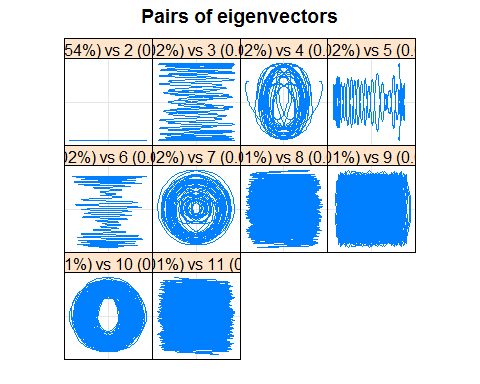


**ECG**

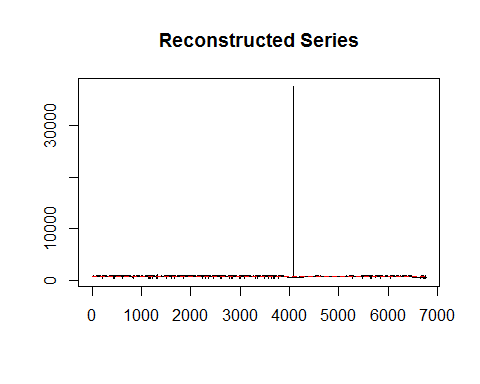
filter\_time\_day1\_fb.clean <- filter\_time\_day1\_fb %>%  
 mutate(fb\_rr=rm\_short\_comp(fb\_rr)) %>%  
 filter(!is.na(fb\_rr)) %>%  
 mutate(fb\_rr=impute\_long(fb\_rr))  
fb.ssa <- ssa(filter\_time\_day1\_fb.clean$fb\_rr, kind="toeplitz-ssa", svd.method = "svd")  
plot(fb.ssa) # Trends at 1 & 42



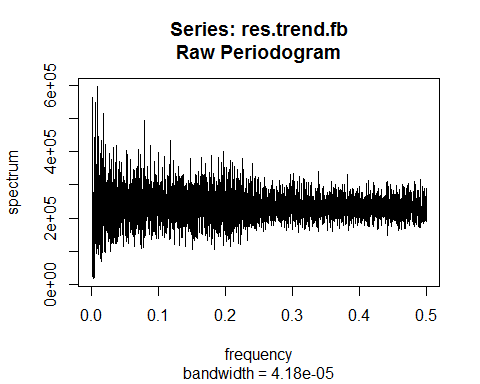
plot(fb.ssa, type="paired") #6&7, 7&8, 8&9, 9&10, 10&11



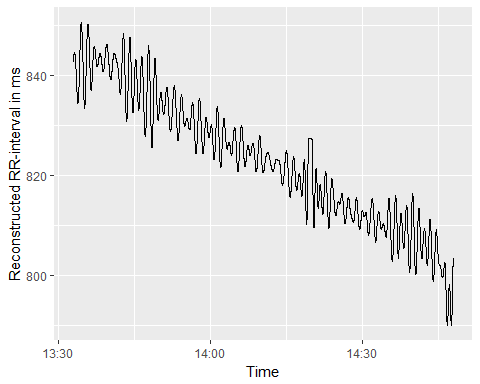
fb\_res <- reconstruct(fb.ssa, group=list(Tr=c(1,42)))  
plot(fb\_res, add.residuals=F)



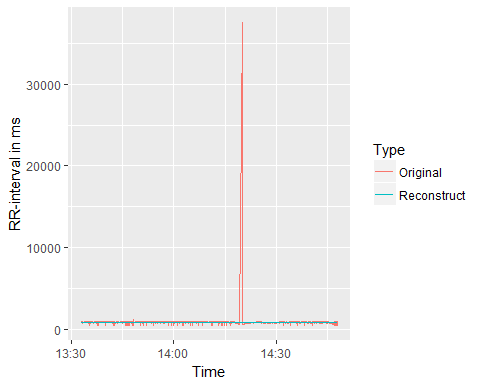
res.trend.fb <- residuals(fb\_res) # Extract seasonality from residuals, original values-predicted values  
spec.pgram(res.trend.fb, detrend=F, log="no") # Periodogram, detect seasonality, should correspond to sine waves in paired plot



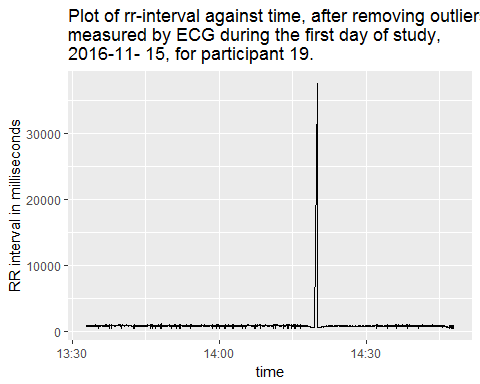
# Reconstructed plot  
df\_fb <- data.frame(Reconstruct=fb\_res$Tr, Original=filter\_time\_day1\_fb.clean$fb\_rr, Time=filter\_time\_day1\_fb.clean$readable\_timestamp)  
ggplot(df\_fb, aes(x=Time, y=Reconstruct)) + geom\_line() + ylab("Reconstructed RR-interval in ms")



# Reconstructed and original plot combined  
df\_fb %>%  
 gather(Reconstruct, Original, key="Type", value="RR-interval") %>%  
 ggplot(aes(x=Time, y=`RR-interval`)) +  
 geom\_line(aes(colour=Type)) +  
 ylab("RR-interval in ms")



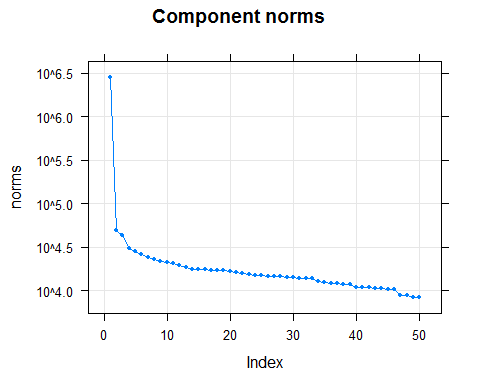
# Original Plot  
ggplot(filter\_time\_day1\_fb.clean, aes(x=readable\_timestamp, y=fb\_rr)) + geom\_line() +  
 ggtitle("Plot of rr-interval against time, after removing outliers, \nmeasured by ECG during the first day of study, \n2016-11- 15, for participant 19.") +  
 ylab("RR interval in milliseconds") +   
 xlab("time")



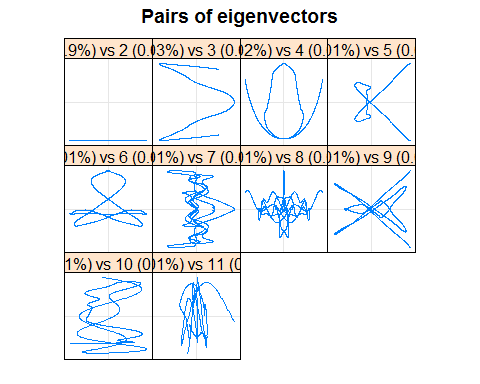
Huge spike of ECG at, 14:19:58, >37,000ms. This is due to removal of previous short rr-intervals (rows 4081-4177) and adding it to the rr-interval at row 4178.

**MSband**

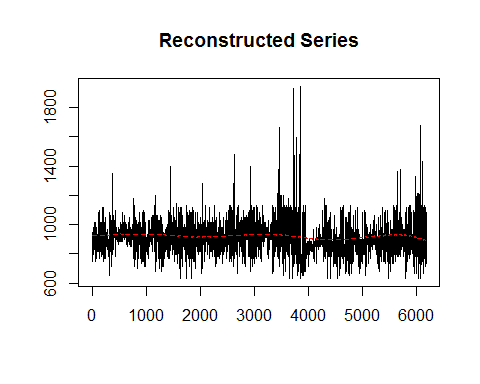
filter\_time\_day1\_mb.clean <- filter\_time\_day1\_mb %>%  
 mutate(mb\_rr=rm\_short\_comp(mb\_rr)) %>%  
 filter(!is.na(mb\_rr)) %>%  
 mutate(mb\_rr=impute\_long(mb\_rr))  
  
mb.ssa <- ssa(filter\_time\_day1\_mb.clean$mb\_rr, kind="toeplitz-ssa", svd.method = "svd")  
plot(mb.ssa) # Trends at 1 & 3



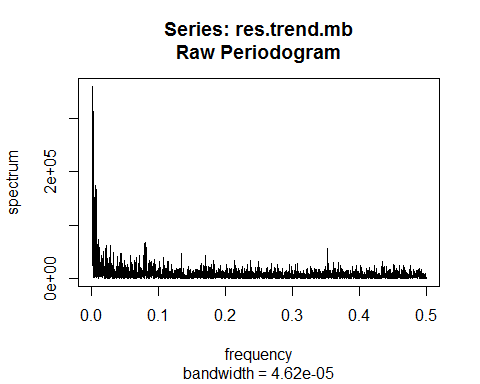
plot(mb.ssa, type="paired") # 3&4



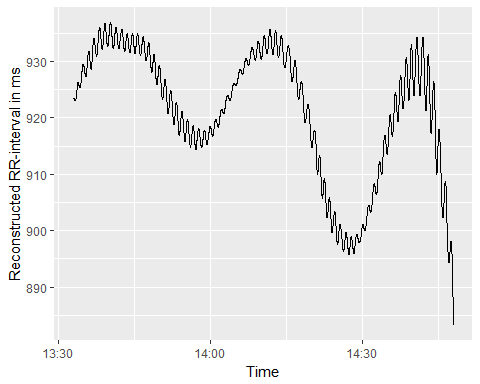
mb\_res <- reconstruct(mb.ssa, group=list(Tr=c(1,3,46)))  
plot(mb\_res, add.residuals=F)



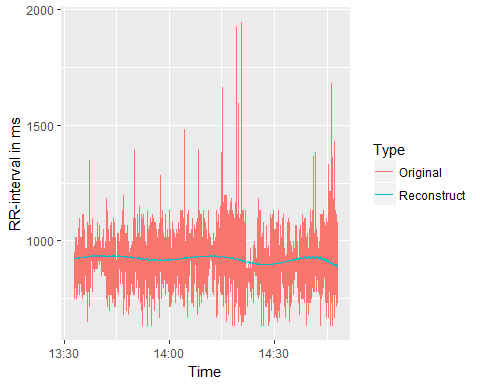
res.trend.mb <- residuals(mb\_res) # Extract seasonality from residuals, original values-predicted values  
spec.pgram(res.trend.mb, detrend=F, log="no") # Periodogram, detect seasonality, should correspond to sine waves in paired plot



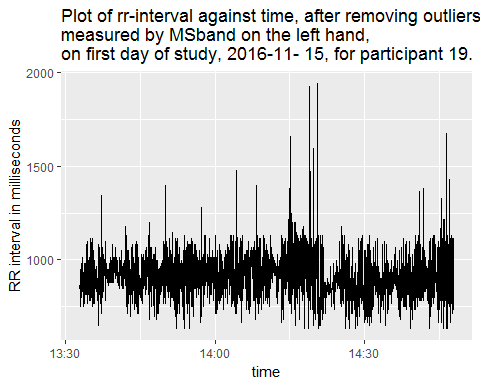
# Reconstructed plot  
df\_mb <- data.frame(Reconstruct=mb\_res$Tr, Original=filter\_time\_day1\_mb.clean$mb\_rr, Time=filter\_time\_day1\_mb.clean$readable\_timestamp)  
ggplot(df\_mb, aes(x=Time, y=Reconstruct)) + geom\_line() + ylab("Reconstructed RR-interval in ms")



# Reconstructed and original plot combined  
df\_mb %>%  
 gather(Reconstruct, Original, key="Type", value="RR-interval") %>%  
 ggplot(aes(x=Time, y=`RR-interval`)) +  
 geom\_line(aes(colour=Type)) +  
 ylab("RR-interval in ms")



# Original Plot  
ggplot(filter\_time\_day1\_mb.clean, aes(x=readable\_timestamp, y=mb\_rr)) + geom\_line() +  
 ggtitle("Plot of rr-interval against time, after removing outliers, \nmeasured by MSband on the left hand, \non first day of study, 2016-11- 15, for participant 19.") +  
 ylab("RR interval in milliseconds") +   
 xlab("time")



# RMSSD equation

# x should be the column containing rr-interval, units in millisecond  
HRV <- function(x){  
 total <- 0  
 for (i in 1:(length(x)-1)){  
 total <- sum(total, (x[i+1] -x[i])^2, na.rm=T)  
 }  
 return (sqrt(total/(length(x)-1)))  
}

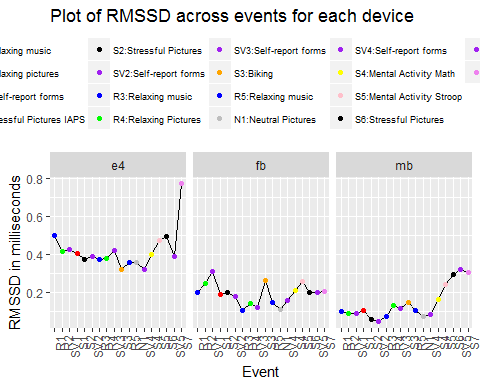
# Select each period and calculate RMSSD for E4, fb, and MSband

for (i in 3:(n-1)){  
 # select time for rr-interval   
 rr\_int\_e4 <- df\_e4 %>%  
 filter(strftime(Time, "%H:%M")>=df$Time[i] & strftime(Time, "%H:%M") <df$Time[i+1])  
 rr\_int\_fb <- df\_fb %>%  
 filter(strftime(Time, "%H:%M")>=df$Time[i] & strftime(Time, "%H:%M") < df$Time[i+1])  
 rr\_int\_mb <- df\_mb %>%  
 filter(strftime(Time, "%H:%M")>=df$Time[i] & strftime(Time, "%H:%M") < df$Time[i+1])  
 # calculate RMSSD for each rr interval according to time  
 df$RMSSD\_e4[i] <- HRV(rr\_int\_e4$Reconstruct)  
 df$RMSSD\_fb[i] <- HRV(rr\_int\_fb$Reconstruct)  
 df$RMSSD\_mb[i] <- HRV(rr\_int\_mb$Reconstruct)  
}  
df$RMSSD\_e4[n] <- NA  
df$RMSSD\_fb[n] <- NA  
df$RMSSD\_mb[n] <- NA  
  
df #output

## # A tibble: 21 x 6  
## Event Time Activity RMSSD\_e4 RMSSD\_fb  
## <chr> <chr> <chr> <dbl> <dbl>  
## 1 Date 11/15/2016 <NA> NA NA  
## 2 Session 1 Start 13:32:52 <NA> NA NA  
## 3 R1 13:35 Relaxing music 0.5024974 0.2025361  
## 4 R2 13:44 Relaxing pictures 0.4173555 0.2467778  
## 5 SV1 13:46 Self-report forms 0.4245374 0.3116567  
## 6 S1 13:48 Stressful Pictures IAPS 0.4061301 0.1891114  
## 7 S2 14:00 Stressful Pictures 0.3765007 0.2004154  
## 8 SV2 14:02 Self-report forms 0.3905433 0.1816611  
## 9 R3 14:03 Relaxing music 0.3750090 0.1076057  
## 10 R4 14:14 Relaxing Pictures 0.3783640 0.1408492  
## # ... with 11 more rows, and 1 more variables: RMSSD\_mb <dbl>

# Plot HRV over different periods for each device

hrv\_df <- df %>%  
 slice(3:20) %>%  
 gather(`RMSSD\_e4`, `RMSSD\_fb`, `RMSSD\_mb`, key="Device", value="RMSSD") %>%  
 mutate(Device=str\_replace(Device, "^.\*\_", ""),  
 Event=factor(Event, levels=df$Event[3:20]))  
  
clr <- c("blue", "green", "purple", "red", "black", "purple", "blue", "green", "purple", "orange", "blue", "gray", "purple", "yellow", "pink", "black", "purple", "violet")  
ggplot(hrv\_df, aes(x=Event, y=RMSSD)) +  
 geom\_path(aes(group=Device)) +  
 geom\_point(aes(colour=Event)) +  
 facet\_wrap(~Device) +  
 scale\_colour\_manual(values=clr,   
 labels=paste(hrv\_df$Event,hrv\_df$Activity, sep=":"),  
 name=NULL) +  
 #annotate(geom="text", x=1, y=50, label="E4") +  
 #annotate(geom="text", x=1, y=30, label="ECG") +  
 #annotate(geom="text", x=1, y=100, label="MSB") +  
 ggtitle("Plot of RMSSD across events for each device") +  
 ylab("RMSSD in milliseconds") +  
 theme(legend.position="top",   
 legend.text=element\_text(size=7),  
 axis.text.x = element\_text(angle = 90, hjust = 1))



# Implement timing, 3 min + 30s

time <- hms(start) + minutes(3)  
timings <- c(as.character(hms(start)),vector())  
while (as.character(time)<(hms(end)-seconds(30))){  
 timings <- c(timings, as.character(seconds\_to\_period(period\_to\_seconds(time)+seconds(30))))  
 time <- seconds\_to\_period(period\_to\_seconds(time)+seconds(30))  
}  
timings <- c(timings, as.character(hms(end)))  
for (i in 1:length(timings)){  
 timings[i]<- ifelse(nchar(timings)[i]<11,  
 paste(substr(timings[i],1,3)," 0", substr(timings[i],5,nchar(timings[i])), sep=""),  
 timings[i])  
 # Add zeros in between minutes for some timings  
}  
timings <- substr(str\_replace\_all(timings, "\\D\\s", ":"),1,8) # Replace HMS with semicolon for comparison of timing  
timings

## [1] "13:32:52" "13:36:22" "13:36:52" "13:37:22" "13:37:52" "13:38:22"  
## [7] "13:38:52" "13:39:22" "13:39:52" "13:40:22" "13:40:52" "13:41:22"  
## [13] "13:41:52" "13:42:22" "13:42:52" "13:43:22" "13:43:52" "13:44:22"  
## [19] "13:44:52" "13:45:22" "13:45:52" "13:46:22" "13:46:52" "13:47:22"  
## [25] "13:47:52" "13:48:22" "13:48:52" "13:49:22" "13:49:52" "13:50:22"  
## [31] "13:50:52" "13:51:22" "13:51:52" "13:52:22" "13:52:52" "13:53:22"  
## [37] "13:53:52" "13:54:22" "13:54:52" "13:55:22" "13:55:52" "13:56:22"  
## [43] "13:56:52" "13:57:22" "13:57:52" "13:58:22" "13:58:52" "13:59:22"  
## [49] "13:59:52" "14:00:22" "14:00:52" "14:01:22" "14:01:52" "14:02:22"  
## [55] "14:02:52" "14:03:22" "14:03:52" "14:04:22" "14:04:52" "14:05:22"  
## [61] "14:05:52" "14:06:22" "14:06:52" "14:07:22" "14:07:52" "14:08:22"  
## [67] "14:08:52" "14:09:22" "14:09:52" "14:10:22" "14:10:52" "14:11:22"  
## [73] "14:11:52" "14:12:22" "14:12:52" "14:13:22" "14:13:52" "14:14:22"  
## [79] "14:14:52" "14:15:22" "14:15:52" "14:16:22" "14:16:52" "14:17:22"  
## [85] "14:17:52" "14:18:22" "14:18:52" "14:19:22" "14:19:52" "14:20:22"  
## [91] "14:20:52" "14:21:22" "14:21:52" "14:22:22" "14:22:52" "14:23:22"  
## [97] "14:23:52" "14:24:22" "14:24:52" "14:25:22" "14:25:52" "14:26:22"  
## [103] "14:26:52" "14:27:22" "14:27:52" "14:28:22" "14:28:52" "14:29:22"  
## [109] "14:29:52" "14:30:22" "14:30:52" "14:31:22" "14:31:52" "14:32:22"  
## [115] "14:32:52" "14:33:22" "14:33:52" "14:34:22" "14:34:52" "14:35:22"  
## [121] "14:35:52" "14:36:22" "14:36:52" "14:37:22" "14:37:52" "14:38:22"  
## [127] "14:38:52" "14:39:22" "14:39:52" "14:40:22" "14:40:52" "14:41:22"  
## [133] "14:41:52" "14:42:22" "14:42:52" "14:43:22" "14:43:52" "14:44:22"  
## [139] "14:44:52" "14:45:22" "14:45:52" "14:46:22" "14:46:52" "14:47:22"  
## [145] "14:47:52" "14:47:57"

df\_window <- data.frame(time=timings) %>%  
 mutate(time=as.character(time))  
for (i in 1:(nrow(df\_window)-1)){  
 # select time  
 e4 <- df\_e4 %>%  
 filter(strftime(Time, "%H:%M:%S")>=df\_window$time[i] & strftime(Time, "%H:%M:%S") <df\_window$time[i+1])  
 fb <- df\_fb %>%  
 filter(strftime(Time, "%H:%M:%S")>=df\_window$time[i] & strftime(Time, "%H:%M:%S") < df\_window$time[i+1])  
 mb <- df\_mb %>%  
 filter(strftime(Time, "%H:%M:%S")>=df\_window$time[i] & strftime(Time, "%H:%M:%S") < df\_window$time[i+1])  
 # calculate RMSSD for each time interval  
 df\_window$RMSSD\_e4[i] <- HRV(e4$Reconstruct)  
 df\_window$RMSSD\_fb[i] <- HRV(fb$Reconstruct)  
 df\_window$RMSSD\_mb[i] <- HRV(mb$Reconstruct)  
}  
  
df\_window$RMSSD\_e4[146] <- 0  
df\_window$RMSSD\_e4[df\_window$RMSSD\_e4==0] <- NA  
head(df\_window)

## time RMSSD\_e4 RMSSD\_fb RMSSD\_mb  
## 1 13:32:52 0.5454991 0.27434334 0.07536895  
## 2 13:36:22 0.5842009 0.18369369 0.12796948  
## 3 13:36:52 0.5175099 0.10388016 0.08954365  
## 4 13:37:22 0.5505367 0.07202889 0.11552760  
## 5 13:37:52 0.4923728 0.05903996 0.10238063  
## 6 13:38:22 0.4922153 0.08167374 0.10367363

tail(df\_window)

## time RMSSD\_e4 RMSSD\_fb RMSSD\_mb  
## 141 14:45:52 NA 0.1087998 0.19678659  
## 142 14:46:22 NA 0.2318197 0.35618434  
## 143 14:46:52 NA 0.2074573 0.14538765  
## 144 14:47:22 NA 0.3401026 0.35798154  
## 145 14:47:52 NA 0.1718368 0.56857480  
## 146 14:47:57 NA 0.2743433 0.07536895

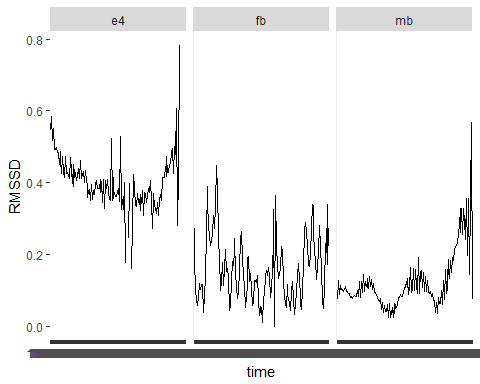
Zeros or NaN in between indicate that there isn't any value for certain timings

# Plot HRV across timing

df\_window1 <- df\_window %>%  
 gather(`RMSSD\_e4`, `RMSSD\_fb`, `RMSSD\_mb`, key="Device", value="RMSSD") %>%  
 mutate(Device=str\_replace(Device, "^.\*\_", ""))  
head(df\_window1)

## time Device RMSSD  
## 1 13:32:52 e4 0.5454991  
## 2 13:36:22 e4 0.5842009  
## 3 13:36:52 e4 0.5175099  
## 4 13:37:22 e4 0.5505367  
## 5 13:37:52 e4 0.4923728  
## 6 13:38:22 e4 0.4922153

ggplot(df\_window1, aes(x=time,y=RMSSD)) +  
 geom\_path(aes(group=Device)) +  
 facet\_wrap(~Device)

 e4 disconnected due to NAs.