HRV analysis

Janice Tjeng

2017-11-05

# Load packages

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

# 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.

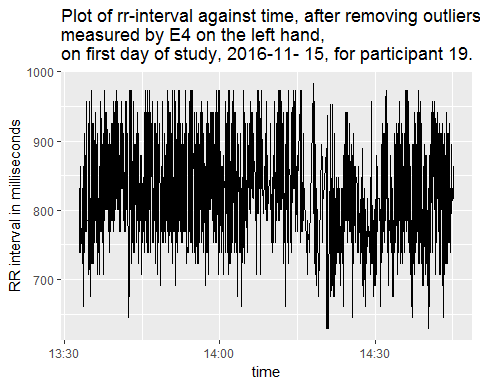
# Standardize function, standard deviation based on median absolute deviation, a robust estimate of standard deviation  
standardize <- function(rr){  
 return ((rr-mean(rr))/sqrt(covMcd(rr)$cov[1])) # use more robust function  
}  
  
# Function to identify outlier  
outlier\_short <- function(rr){  
 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 <- (2\*sqrt(covMcd(rr)$cov[1])) + mean(rr) # For imputation  
 dp <- numeric()  
 dp1 <- numeric()  
 ind\_long <- which(rr>dmax)  
 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)  
}

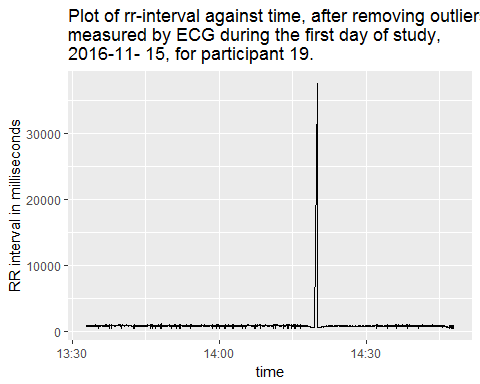
**E4**

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))  
  
# 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**

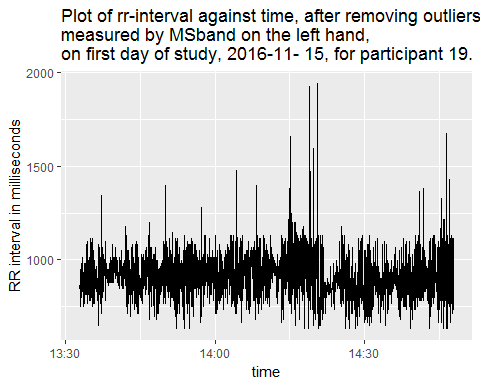
filter\_time\_day1\_fb.clean <- filter\_time\_day1\_fb %>%  
 mutate(fb\_rr=rm\_short\_comp(fb\_rr)) %>%  
 filter(!is.na(fb\_rr)) %>%  
 mutate(fb\_rr=impute\_long(fb\_rr))  
  
# Plot  
ggplot(filter\_time\_day1\_fb.clean, aes(x=readable\_timestamp, y=fb\_rr)) + geom\_line() +  
 ggtitle("Plot of rr-interval 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 remove 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))  
  
# 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)))  
}

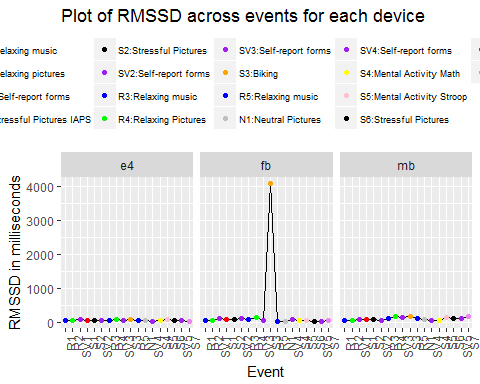
# Select each period and calculate RMSSD for E4, fb, and MSband

for (i in 3:(n-1)){  
 # select rr-interval for E4  
 rr\_int\_e4 <- filter\_time\_day1\_e4.clean %>%  
 filter(strftime(readable\_timestamp, "%H:%M")>=df$Time[i] & strftime(readable\_timestamp, "%H:%M") <df$Time[i+1])  
 # select rr-interval for firstbeat  
 rr\_int\_fb <- filter\_time\_day1\_fb.clean %>%  
 filter(strftime(readable\_timestamp, "%H:%M")>=df$Time[i] & strftime(readable\_timestamp, "%H:%M") < df$Time[i+1])  
 # select rr-interval for MSband  
 rr\_int\_mb <- filter\_time\_day1\_mb.clean %>%  
 filter(strftime(readable\_timestamp, "%H:%M")>=df$Time[i] & strftime(readable\_timestamp, "%H:%M") < df$Time[i+1])  
 # calculate RMSSD for each e4 interval  
 df$RMSSD\_e4[i] <- HRV(rr\_int\_e4$e4\_rr)  
 # calculate RMSSD for each firstbeat interval  
 df$RMSSD\_fb[i] <- HRV(rr\_int\_fb$fb\_rr)  
 # calculate RMSSD for each firstbeat interval  
 df$RMSSD\_mb[i] <- HRV(rr\_int\_mb$mb\_rr)  
}  
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 54.03419 71.18379  
## 4 R2 13:44 Relaxing pictures 66.58650 57.99605  
## 5 SV1 13:46 Self-report forms 80.46968 122.04767  
## 6 S1 13:48 Stressful Pictures IAPS 67.71663 82.44253  
## 7 S2 14:00 Stressful Pictures 70.66291 87.85070  
## 8 SV2 14:02 Self-report forms 73.70199 119.62185  
## 9 R3 14:03 Relaxing music 66.64128 85.76090  
## 10 R4 14:14 Relaxing Pictures 80.43534 140.85046  
## # ... 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 rr-interval for E4  
 e4 <- filter\_time\_day1\_e4.clean %>%  
 filter(strftime(readable\_timestamp, "%H:%M:%S")>=df\_window$time[i] & strftime(readable\_timestamp, "%H:%M:%S") <df\_window$time[i+1])  
 # select rr-interval for firstbeat  
 fb <- filter\_time\_day1\_fb.clean %>%  
 filter(strftime(readable\_timestamp, "%H:%M:%S")>=df\_window$time[i] & strftime(readable\_timestamp, "%H:%M:%S") < df\_window$time[i+1])  
 # select rr-interval for MSband  
 mb <- filter\_time\_day1\_mb.clean %>%  
 filter(strftime(readable\_timestamp, "%H:%M:%S")>=df\_window$time[i] & strftime(readable\_timestamp, "%H:%M:%S") < df\_window$time[i+1])  
 # calculate RMSSD for each e4 interval  
 df\_window$RMSSD\_e4[i] <- HRV(e4$e4\_rr)  
 # calculate RMSSD for each firstbeat interval  
 df\_window$RMSSD\_fb[i] <- HRV(fb$fb\_rr)  
 # calculate RMSSD for each firstbeat interval  
 df\_window$RMSSD\_mb[i] <- HRV(mb$mb\_rr)  
}  
  
df\_window$RMSSD\_e4[146] <- 0  
df\_window

## time RMSSD\_e4 RMSSD\_fb RMSSD\_mb  
## 1 13:32:52 55.48295 50.590055 71.42453  
## 2 13:36:22 47.24755 26.351892 124.90547  
## 3 13:36:52 51.01241 39.844938 153.70169  
## 4 13:37:22 48.38489 115.558034 73.10611  
## 5 13:37:52 49.85306 111.522353 70.20089  
## 6 13:38:22 51.48704 33.309158 27.92850  
## 7 13:38:52 44.79843 30.910571 37.46105  
## 8 13:39:22 54.08047 42.905382 44.21075  
## 9 13:39:52 44.54895 124.927237 31.65998  
## 10 13:40:22 44.78418 29.135970 35.81158  
## 11 13:40:52 40.89655 28.807551 42.09236  
## 12 13:41:22 43.73330 32.123645 34.10453  
## 13 13:41:52 49.84693 22.303039 126.32887  
## 14 13:42:22 96.07860 164.274116 85.06329  
## 15 13:42:52 64.15527 25.956257 88.63978  
## 16 13:43:22 56.33932 51.551961 79.97371  
## 17 13:43:52 57.58016 25.018629 101.82315  
## 18 13:44:22 63.75402 26.699563 67.58105  
## 19 13:44:52 50.72573 34.263683 53.58938  
## 20 13:45:22 76.32539 40.209512 37.90650  
## 21 13:45:52 89.68361 152.382028 47.43977  
## 22 13:46:22 86.16551 109.451300 122.97060  
## 23 13:46:52 77.20078 156.192883 114.40805  
## 24 13:47:22 62.60575 37.554109 69.18707  
## 25 13:47:52 81.75738 189.273913 103.24792  
## 26 13:48:22 60.31593 25.500213 91.91562  
## 27 13:48:52 64.34737 32.511886 43.32445  
## 28 13:49:22 66.79467 34.673305 100.46944  
## 29 13:49:52 80.71309 108.363144 115.53566  
## 30 13:50:22 53.08797 38.683699 83.17249  
## 31 13:50:52 76.57317 121.074495 76.91212  
## 32 13:51:22 56.79102 40.062741 84.66434  
## 33 13:51:52 78.06868 206.812063 144.88605  
## 34 13:52:22 52.54883 40.238533 144.97202  
## 35 13:52:52 57.81739 36.578357 50.40842  
## 36 13:53:22 42.43488 36.039793 148.68567  
## 37 13:53:52 59.23922 103.032997 93.56512  
## 38 13:54:22 95.18495 37.061681 129.48777  
## 39 13:54:52 68.53972 36.349166 128.20723  
## 40 13:55:22 81.32608 32.577992 105.75401  
## 41 13:55:52 54.51654 37.474022 59.27159  
## 42 13:56:22 54.94796 39.450494 53.88600  
## 43 13:56:52 88.71064 31.260403 125.42772  
## 44 13:57:22 72.88469 39.108823 76.53828  
## 45 13:57:52 53.37183 38.100637 86.94750  
## 46 13:58:22 78.40903 119.465590 72.48296  
## 47 13:58:52 67.93127 127.569505 71.12423  
## 48 13:59:22 78.67724 40.124501 68.02798  
## 49 13:59:52 55.81698 116.866432 113.14236  
## 50 14:00:22 56.33923 28.133017 70.92922  
## 51 14:00:52 60.17950 173.752074 74.23334  
## 52 14:01:22 86.81868 40.513564 63.45850  
## 53 14:01:52 60.94902 41.060654 47.39038  
## 54 14:02:22 88.59938 120.680991 81.38764  
## 55 14:02:52 70.25858 115.876591 93.97257  
## 56 14:03:22 54.67211 29.366416 143.63179  
## 57 14:03:52 72.20509 33.111674 132.45453  
## 58 14:04:22 58.13885 103.372815 123.36916  
## 59 14:04:52 70.74867 75.261669 143.64799  
## 60 14:05:22 59.39828 74.529864 67.45075  
## 61 14:05:52 46.11579 29.232928 91.79148  
## 62 14:06:22 39.13535 25.534291 105.32595  
## 63 14:06:52 88.17859 215.756579 77.29122  
## 64 14:07:22 63.60625 83.337813 51.63219  
## 65 14:07:52 98.36820 122.472055 211.11503  
## 66 14:08:22 54.36576 32.208922 180.92063  
## 67 14:08:52 51.09490 37.670564 98.85498  
## 68 14:09:22 47.39511 35.795600 79.37614  
## 69 14:09:52 46.58679 22.931298 103.01330  
## 70 14:10:22 58.81309 28.317197 72.17132  
## 71 14:10:52 66.92709 166.145937 100.15547  
## 72 14:11:22 63.31517 87.490432 30.21289  
## 73 14:11:52 57.12649 40.158224 30.81827  
## 74 14:12:22 82.50594 31.216619 74.51677  
## 75 14:12:52 83.75909 116.424711 35.04621  
## 76 14:13:22 121.03610 34.944206 170.25392  
## 77 14:13:52 57.50078 203.223097 86.21458  
## 78 14:14:22 90.25604 123.864604 160.04220  
## 79 14:14:52 74.52975 43.771580 259.66213  
## 80 14:15:22 85.54236 180.825405 148.81524  
## 81 14:15:52 94.72573 35.496159 191.02380  
## 82 14:16:22 NaN 112.959683 152.48893  
## 83 14:16:52 36.97719 37.786481 113.83599  
## 84 14:17:22 0.00000 34.387828 119.68791  
## 85 14:17:52 103.83990 16.164335 132.99256  
## 86 14:18:22 77.37917 3.851407 138.84338  
## 87 14:18:52 NaN 2.420615 268.36221  
## 88 14:19:22 69.44200 0.000000 204.90531  
## 89 14:19:52 15.62500 5079.290436 144.34726  
## 90 14:20:22 178.33474 7.979813 281.71611  
## 91 14:20:52 91.88259 13.865266 131.04978  
## 92 14:21:22 105.75676 17.651233 146.49573  
## 93 14:21:52 63.80669 26.535306 41.41081  
## 94 14:22:22 39.23370 30.452743 20.75803  
## 95 14:22:52 35.44788 22.274331 29.43335  
## 96 14:23:22 37.29171 26.182885 20.98740  
## 97 14:23:52 53.25046 34.233806 53.31581  
## 98 14:24:22 50.91927 34.777031 26.72929  
## 99 14:24:52 44.02843 28.895973 74.53820  
## 100 14:25:22 86.55615 22.489128 136.83760  
## 101 14:25:52 57.34991 22.789681 144.13821  
## 102 14:26:22 57.33031 18.277377 102.75925  
## 103 14:26:52 30.03912 16.229734 77.98770  
## 104 14:27:22 98.31376 20.321417 77.75582  
## 105 14:27:52 44.96282 13.979151 86.86419  
## 106 14:28:22 21.43848 14.094177 78.29701  
## 107 14:28:52 67.00647 17.416671 111.70601  
## 108 14:29:22 55.68551 15.522242 151.93713  
## 109 14:29:52 49.33035 18.347343 96.89841  
## 110 14:30:22 48.91616 25.545602 128.05532  
## 111 14:30:52 60.51794 17.271302 135.78292  
## 112 14:31:22 54.87560 33.439711 81.34136  
## 113 14:31:52 39.93354 121.743767 51.62308  
## 114 14:32:22 55.10102 20.029765 80.39708  
## 115 14:32:52 44.68670 19.952603 67.49663  
## 116 14:33:22 56.33935 15.997396 82.83360  
## 117 14:33:52 54.21813 119.291405 91.86824  
## 118 14:34:22 58.30098 36.685337 60.52523  
## 119 14:34:52 46.04736 33.749074 39.18586  
## 120 14:35:22 38.70752 24.070446 28.51964  
## 121 14:35:52 71.70595 161.855622 37.65324  
## 122 14:36:22 44.69558 35.156792 31.44133  
## 123 14:36:52 46.62436 24.905780 88.23857  
## 124 14:37:22 56.35457 44.776163 47.81810  
## 125 14:37:52 45.07130 117.641810 95.80842  
## 126 14:38:22 67.86684 48.544285 53.35680  
## 127 14:38:52 60.87134 22.137385 95.71648  
## 128 14:39:22 80.57619 27.583006 78.25338  
## 129 14:39:52 67.46264 25.226887 122.72954  
## 130 14:40:22 102.14350 107.379397 141.03070  
## 131 14:40:52 79.06438 27.218393 209.19100  
## 132 14:41:22 80.01545 41.937785 133.30387  
## 133 14:41:52 56.75903 38.509739 78.35029  
## 134 14:42:22 66.05004 29.469381 37.35879  
## 135 14:42:52 95.84699 35.067237 192.32098  
## 136 14:43:22 88.66827 45.191392 89.30746  
## 137 14:43:52 90.29010 43.831029 110.43642  
## 138 14:44:22 60.27539 39.104703 118.53937  
## 139 14:44:52 27.06445 29.752005 150.27292  
## 140 14:45:22 0.00000 16.494171 177.16402  
## 141 14:45:52 0.00000 16.449747 157.44807  
## 142 14:46:22 0.00000 17.703946 246.69685  
## 143 14:46:52 0.00000 90.828894 202.38873  
## 144 14:47:22 0.00000 141.117024 160.91270  
## 145 14:47:52 0.00000 10.885771 87.71726  
## 146 14:47:57 0.00000 50.590055 71.42453

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 55.48295  
## 2 13:36:22 e4 47.24755  
## 3 13:36:52 e4 51.01241  
## 4 13:37:22 e4 48.38489  
## 5 13:37:52 e4 49.85306  
## 6 13:38:22 e4 51.48704

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

