Daily Activity Classification

C.Zhou

27/07/2020

Overview

Machine learning can be used to predict both continuous and categorical outcomes. The previous MovieLens project is a regression task to predict continuous outcomes (ratings of movies). Different from the MovieLens project, this project is a classification task to predict categorical outcomes (daily activity type). A more detailed description of this project is as follow.

Goal

The object of this project is to distinguish different daily activity types based on biometric indicators. This project's goal is slightly different from the target in the description of the dataset because the dataset downloaded did not contain all the data mentioned in the data description (e.g., it mentioned 16 activities of daily living, but there are only six types in the dataset). The original goal mentioned in the data description was to classify different movements measured by wearable sensors to detect the time when someone is about to fall and then help prevent falling over. As the dataset available only contains biometric indicators and monitoring time as predictors, it is impossible to classify the activity types based on the physical data measured by tri-axial devices (accelerometer, gyroscope, and magnetometer/compass). Therefore, we will make use of the available predictors to distinguish the activity type instead. ## Dataset and Variables The raw data (falldeteciton dataset) used in this project can be obtained from: https://www.kaggle.com/pitasr/falldata/download

This falldeteciton dataset contains 16382 observations of patients of 65-year old and above in Chinese hospitals.

The dependent variable (DV) is:

ACTIVITY: six types of daily activities coded as "0 = Standing", "1 = Walking", "2 = Sitting", "3 = Falling", "4 = Cramps", "5 = Running"

The independent variables (IVs) are:

TIME: monitoring time

SL: sugar level

EEG: EEG monitoring rate

BP: Blood pressure

HR: Heart beat rate

CIRCLUATION: Blood circulation.

We will then use the 6 IVs mentioned above to predict rating (the only DV).

Key Steps

The following steps were carried out to determine the model that predicts the types of activity most accurately:

Step 1. Load the data and split data sets.
Step 2. Inspect the data to select the useful predictors, including visualizing the distribution of variables and exploring the correlation between variables.
Step 3. Several models were trained on the test data set to find out the model that maximizes accuracy, including k-nearest neighbors (kNN), decision trees, and random forest. Techniques, such as tuning parameter and cross-validation, were also applied to improve the models.
Step 4. The final algorithm was tested on the validation set. The overall accuracy and ROC curve were reported as indicators of how well the predicted activity type match the real activity type based on the final model.

Analysis

Step 1. Create the Train and the Validation Datasets

1.1 Load packages

```
# data wrangling
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
#dataframe manipulation
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
#machine learning
if(!require(caret)) install.packages("caret", repos = "httpggthemes://cran.us.r-project.org")
#inspect missing data
if(!require(varhandle)) install.packages("varhandle", repos = "http://cran.us.r-project.org")
#data description
if(!require(dslabs)) install.packages("dslabs", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
#to ensure labels donâ t fall on top of each other.
if(!require(ggrepel)) install.packages("ggrepel", repos = "http://cran.us.r-project.org")
```

1.2 Import data & create train/valid data set

```
#Importing the data
fall <- read.csv("falldeteciton.csv") %>% mutate(ACTIVITY = as.factor(ACTIVITY))
levels(fall$ACTIVITY) <- c("Standing", "Walking", "Sitting", "Falling", "Cramps", "Running")
summary(fall) #summary of the data
```

```
##      ACTIVITY         TIME             SL                EEG
##  Standing:4608   Min.   : 1954   Min.   :     42.2   Min.   :-12626000
##  Walking : 502   1st Qu.: 7264   1st Qu.:   9941.2   1st Qu.:    -5630
##  Sitting :2502   Median : 9769   Median :  31189.2   Median :    -3361
##  Falling :3588   Mean   :10937   Mean   :  75272.0   Mean   :    -5621
##  Cramps  :3494   3rd Qu.:13482   3rd Qu.:  80761.4   3rd Qu.:    -2150
##  Running :1688   Max.   :50896   Max.   :2426140.0   Max.   :  1410000
##        BP               HR         CIRCLUATION
##  Min.   :  0.00   Min.   : 33.0   Min.   :    5
##  1st Qu.: 25.00   1st Qu.:119.0   1st Qu.:  587
##  Median : 44.00   Median :180.0   Median : 1581
##  Mean   : 58.25   Mean   :211.5   Mean   : 2894
##  3rd Qu.: 78.00   3rd Qu.:271.0   3rd Qu.: 3539
```

```
##  Max.   :533.00   Max.   :986.0   Max.   :52210
```

```
names(fall) # the top 6rows of the data
```

```
## [1] "ACTIVITY"    "TIME"        "SL"          "EEG"         "BP"
## [6] "HR"          "CIRCLUATION"
```

```
#Based on the Pareto principle, a ratio of 80:20 (train: validation) was used to split the raw data set
set.seed(2020)
test_index <- createDataPartition(y = fall$ACTIVITY, times = 1, p = 0.2, list = FALSE)
trainset <- fall[-test_index,]
validset <- fall[test_index,]
summary(validset)
```

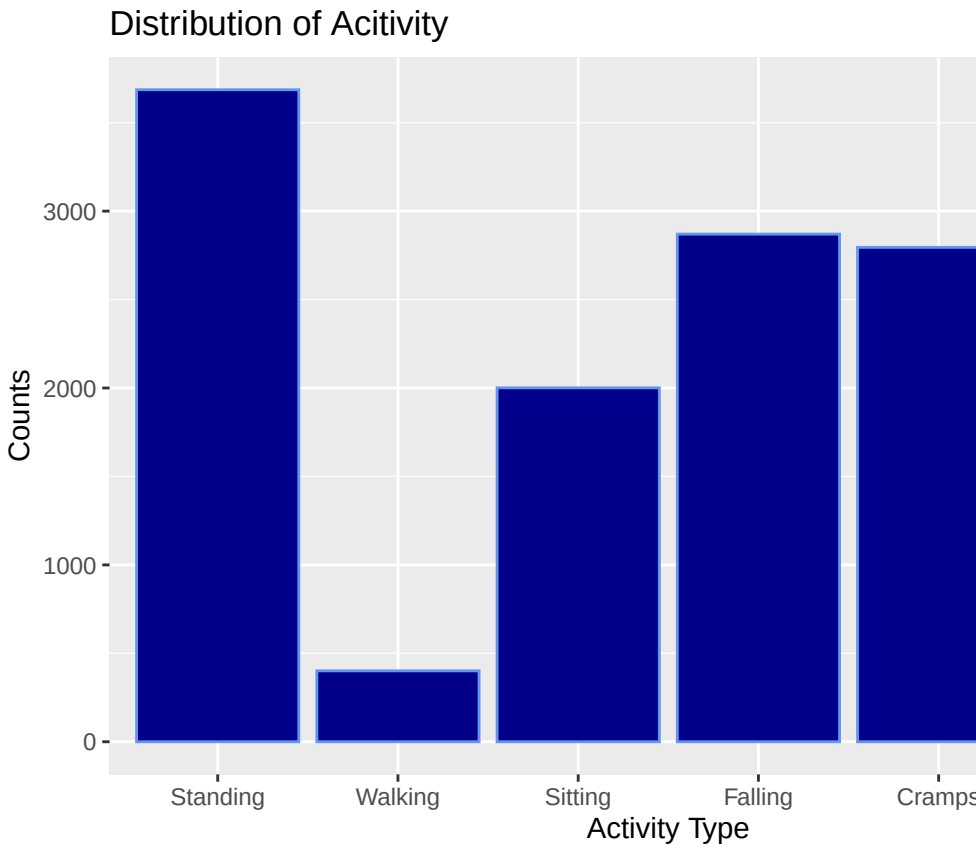
```
##      ACTIVITY        TIME             SL                EEG
##  Standing:922   Min.   : 2080   Min.   :     47.8   Min.   :-2730700
##  Walking :101   1st Qu.: 7303   1st Qu.:  10110.5   1st Qu.:   -5516
##  Sitting :501   Median : 9688   Median :  30363.3   Median :   -3320
##  Falling :718   Mean   :10851   Mean   :  73496.6   Mean   :   -5833
##  Cramps  :699   3rd Qu.:13322   3rd Qu.:  79920.3   3rd Qu.:   -2183
##  Running :338   Max.   :41892   Max.   :1332080.0   Max.   :   24900
##        BP               HR         CIRCLUATION
##  Min.   :  0.00   Min.   : 33.0   Min.   :    5
##  1st Qu.: 25.00   1st Qu.:120.0   1st Qu.:  587
##  Median : 43.00   Median :177.0   Median : 1521
##  Mean   : 56.66   Mean   :209.5   Mean   : 2835
##  3rd Qu.: 77.00   3rd Qu.:262.0   3rd Qu.: 3466
##  Max.   :479.00   Max.   :954.0   Max.   :35169
```

Step 2. Data visualization

```
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(graphics)) install.packages("graphics", repos = "http://cran.us.r-project.org")
```

2.1 distribution of each variable

```
trainset %>%
  ggplot(aes(ACTIVITY)) +
  geom_bar(color="cornflowerblue", fill="darkblue")+
  ggtitle("Distribution of Acitivity") +
  labs(y = "Counts", x = "Activity Type")
```

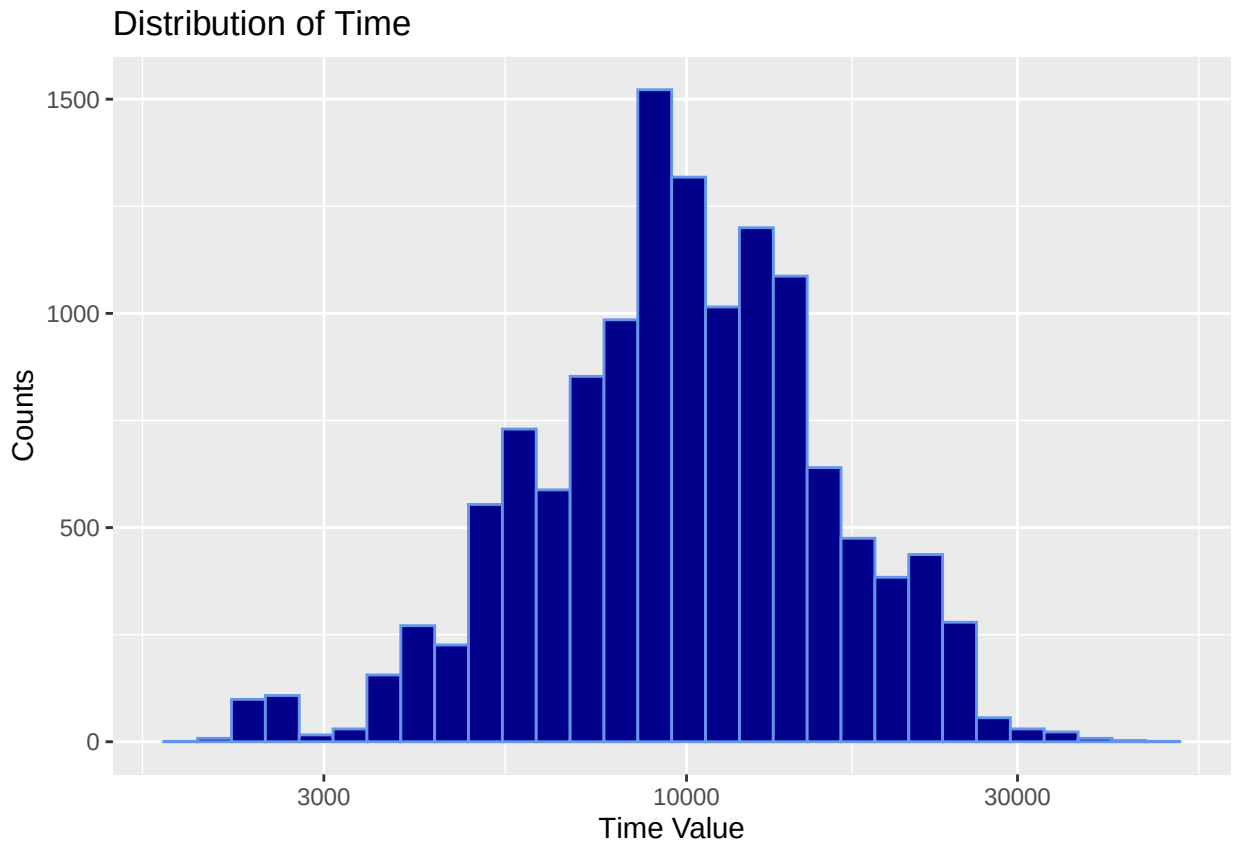


2.1.1 distribution of activity type

2.1.2 histogram of TIME

```
trainset%>%
  ggplot(aes(TIME)) +
  geom_histogram(color="cornflowerblue", fill="darkblue") +
  scale_x_log10() +
  ggtitle("Distribution of Time") +
  labs(y = "Counts", x = "Time Value")
```

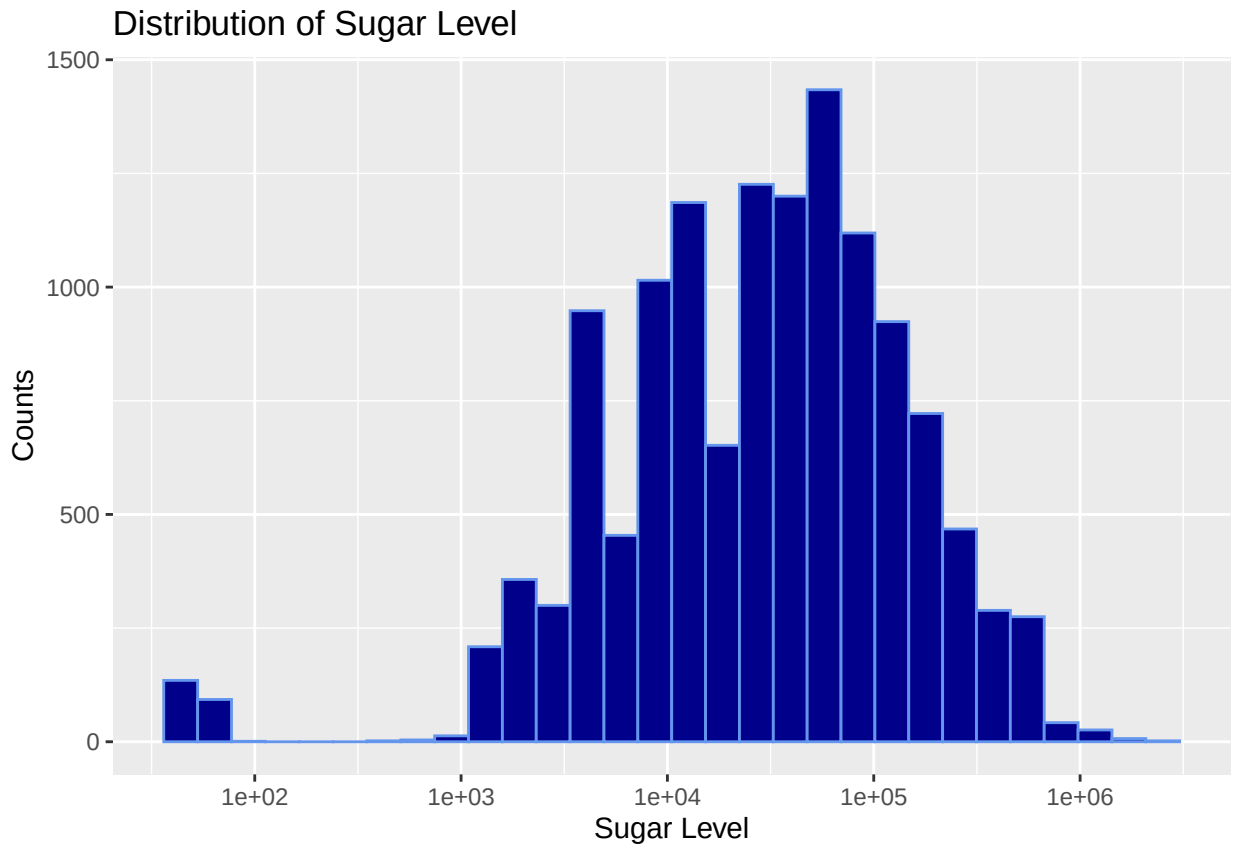
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



2.1.3 histogram of SL

```
trainset%>%
  ggplot(aes(SL)) +
  geom_histogram(color="cornflowerblue", fill="darkblue") +
  scale_x_log10() +
  ggtitle("Distribution of Sugar Level") +
  labs(y = "Counts", x ="Sugar Level")
```

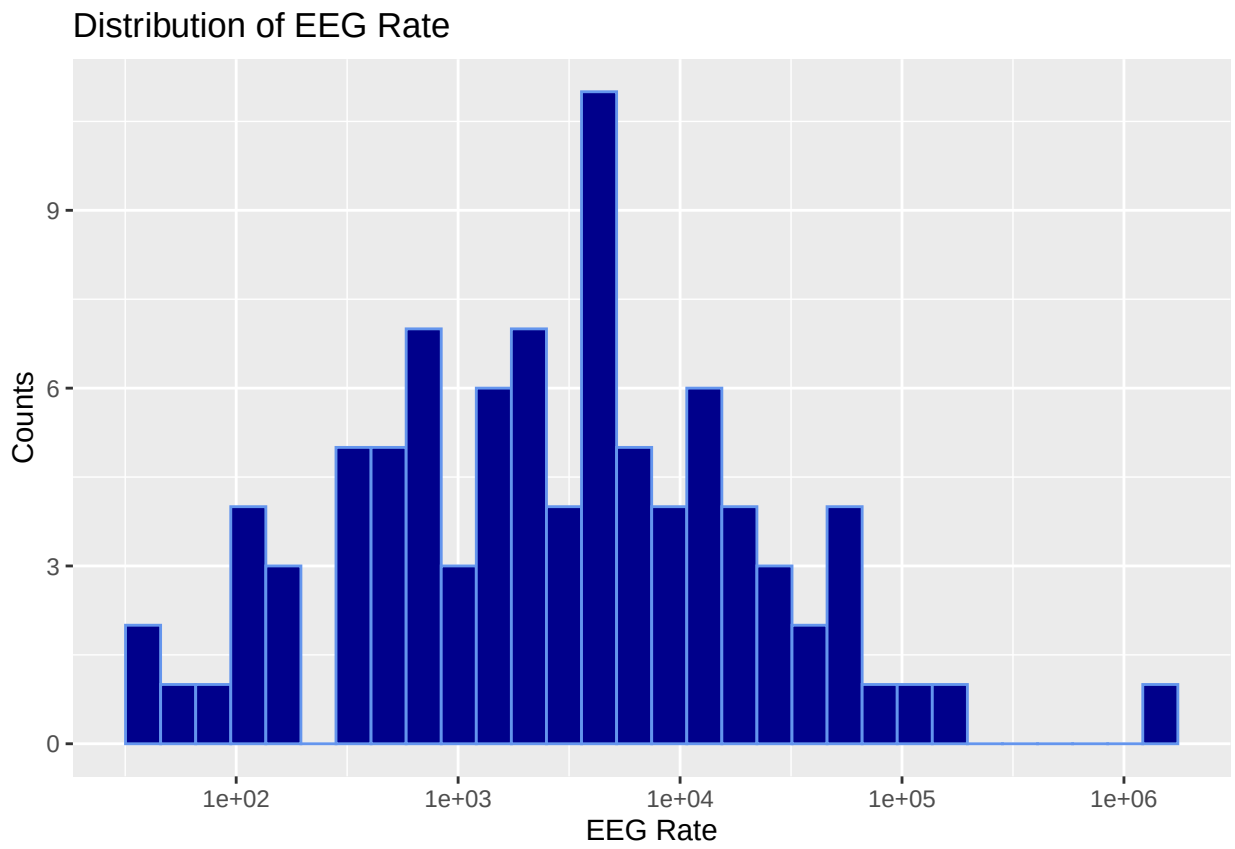
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



2.1.4 histogram of EEG

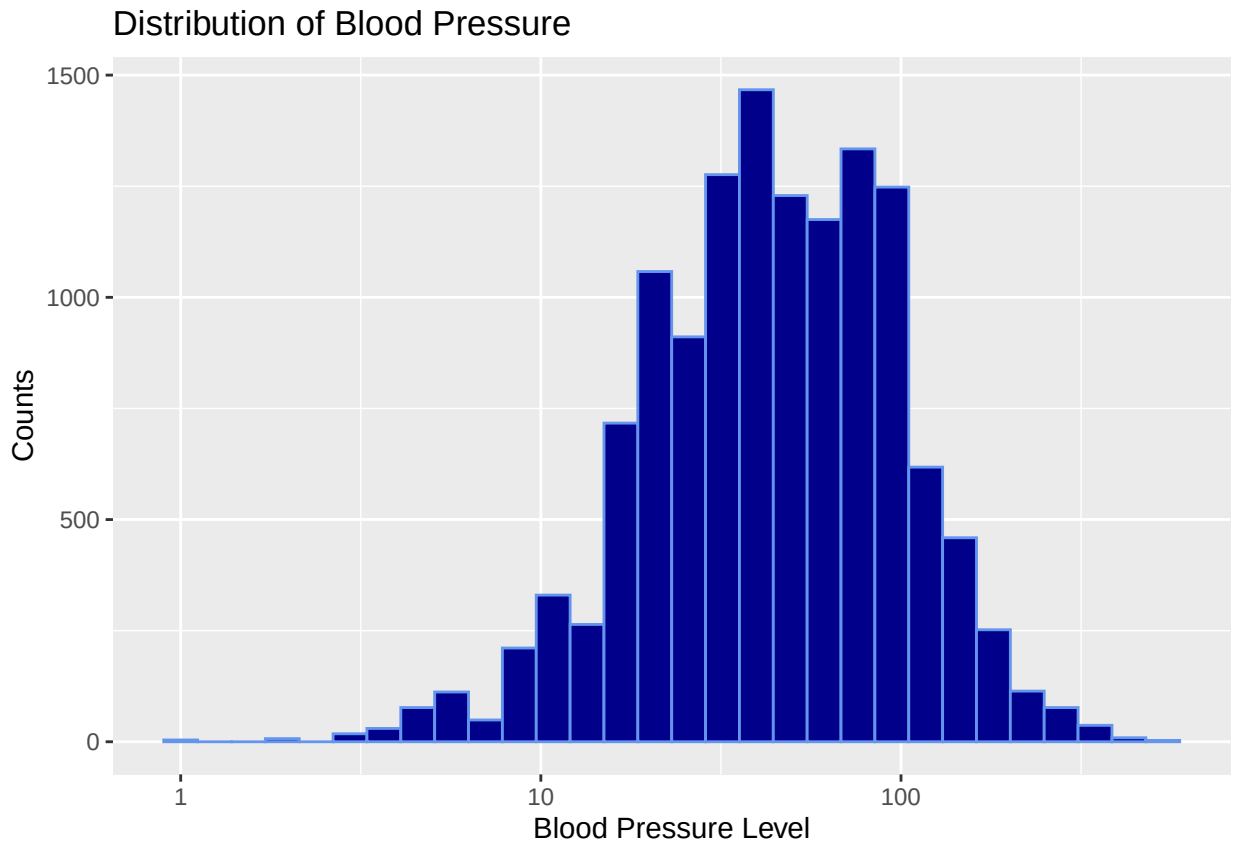
```
trainset%>%
  ggplot(aes(EEG)) +
  geom_histogram(color="cornflowerblue", fill="darkblue") +
  scale_x_log10() +
  ggtitle("Distribution of EEG Rate") +
  labs(y = "Counts", x = "EEG Rate")
```

```
## Warning in self$trans$transform(x): NaNs produced
## Warning: Transformation introduced infinite values in continuous x-axis
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Removed 13012 rows containing non-finite values (stat_bin).
```



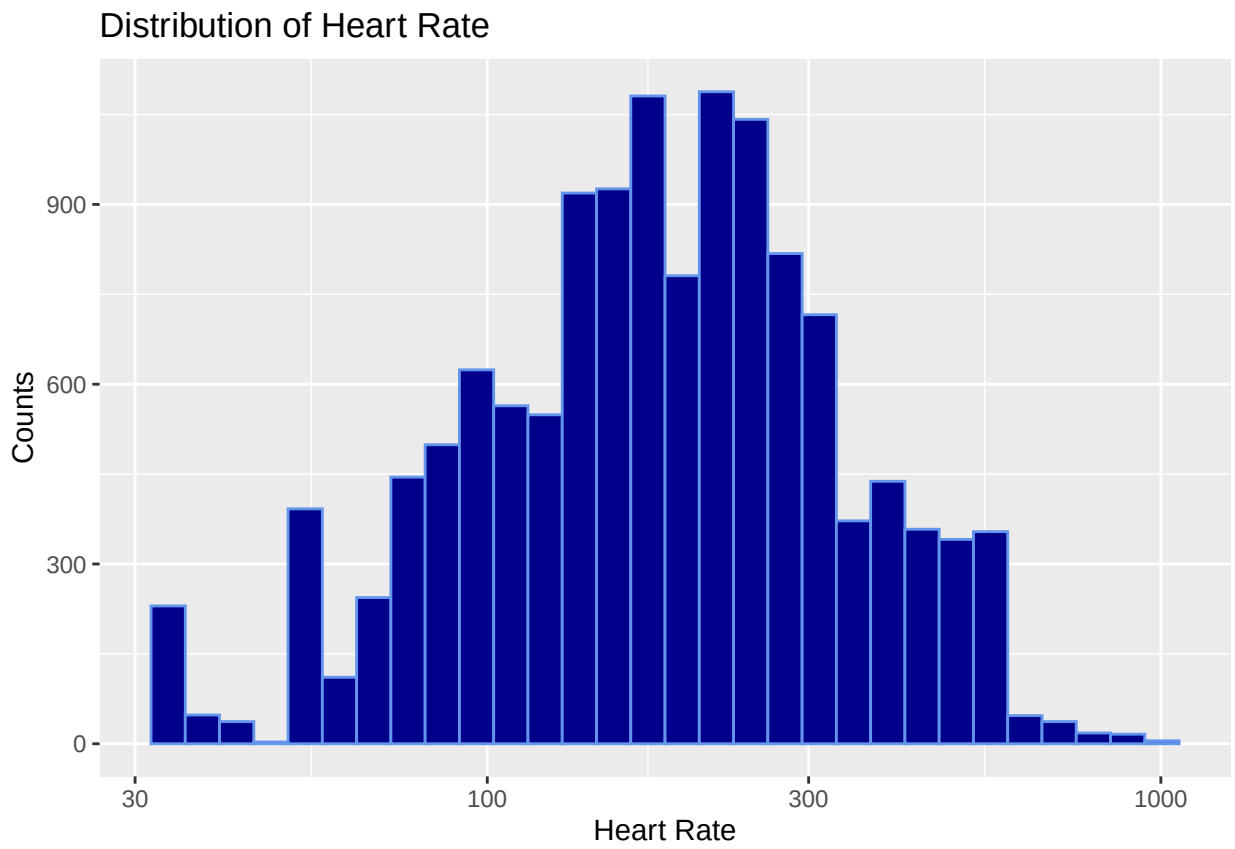
2.1.5 histogram of Blood Pressure

```
trainset%>%
  ggplot(aes(BP)) +
  geom_histogram(color="cornflowerblue", fill="darkblue") +
  scale_x_log10() +
  ggtitle("Distribution of Blood Pressure") +
  labs(y = "Counts", x = "Blood Pressure Level")
```



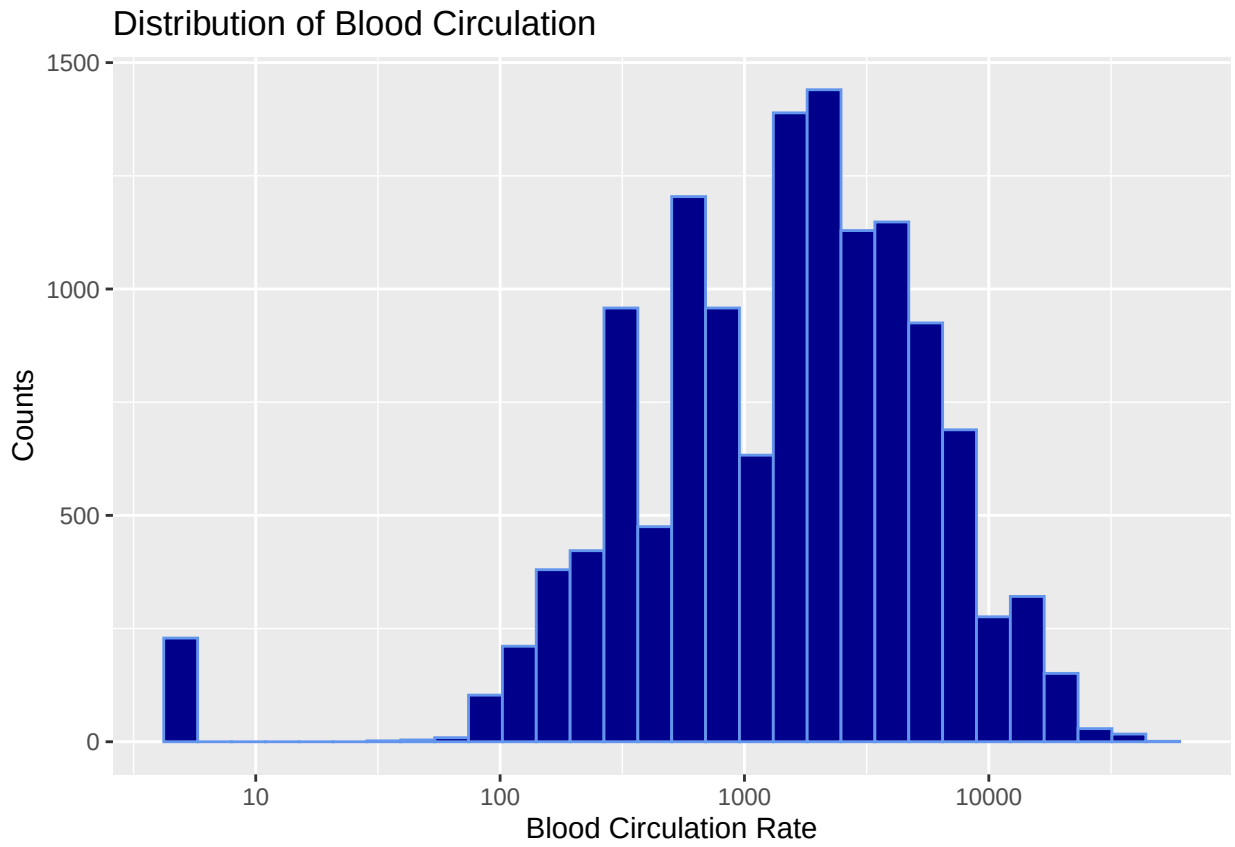
2.1.6 histogram of Heart Rate

```
trainset%>%
  ggplot(aes(HR)) +
  geom_histogram(color="cornflowerblue", fill="darkblue") +
  scale_x_log10() +
  ggtitle("Distribution of Heart Rate") +
  labs(y ="Counts", x =  "Heart Rate")
```



2.1.7 histogram of Blood Circulation

```
trainset%>%
  ggplot(aes(CIRCLUATION)) +
  geom_histogram(color="cornflowerblue", fill="darkblue") +
  scale_x_log10() +
  ggtitle("Distribution of Blood Circulation") +
  labs(y = "Counts", x = "Blood Circulation Rate")
```



2.2 plot the correlation between variables

```
# to make correlation plots
if(!require(corrplot)) install.packages("corrplot", repos = "http://cran.us.r-project.org")
 # to change themes in plot
if(!require(ggthemes)) install.packages("ggthemes", repos = "http://cran.us.r-project.org")
# to change the theme color
if(!require(RColorBrewer)) install.packages("RColorBrewer", repos ="http://cran.us.r-project.org")
```

The MANOVA test showed that different activity types were significantly different on all variable (TIME, SL, EEG, BP, HR, CIRCLUATION). We will then visualize the type difference based on plots.

```
maov <- manova(cbind(TIME, SL, EEG, BP, HR, CIRCLUATION) ~ ACTIVITY, data = trainset)
summary.aov(maov)
```

```
##  Response TIME :
##               Df     Sum Sq   Mean Sq F value    Pr(>F)
## ACTIVITY       5 2.4075e+10 4.815e+09  184.55 < 2.2e-16 ***
## Residuals  13097 3.4170e+11 2.609e+07
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response SL :
##               Df     Sum Sq    Mean Sq F value    Pr(>F)
## ACTIVITY       5 8.9936e+12 1.7987e+12  112.62 < 2.2e-16 ***
## Residuals  13097 2.0918e+14 1.5972e+10
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
##  Response EEG :
##                Df     Sum Sq    Mean Sq F value  Pr(>F)
## ACTIVITY        5 1.6636e+11 3.3271e+10  2.4488 0.03165 *
## Residuals   13097 1.7795e+14 1.3587e+10
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response BP :
##                Df   Sum Sq Mean Sq F value    Pr(>F)
## ACTIVITY        5  1212228  242446  106.12 < 2.2e-16 ***
## Residuals   13097 29922537    2285
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response HR :
##                Df    Sum Sq Mean Sq F value    Pr(>F)
## ACTIVITY        5  15502383 3100477  195.91 < 2.2e-16 ***
## Residuals   13097 207268596   15826
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response CIRCLUATION :
##                Df     Sum Sq    Mean Sq F value    Pr(>F)
## ACTIVITY        5 8.2051e+09 1641020478  115.52 < 2.2e-16 ***
## Residuals   13097 1.8605e+11   14205368
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2.2.1 TIME VS activity Time differs significantly among activity types, except Running-Cramps.

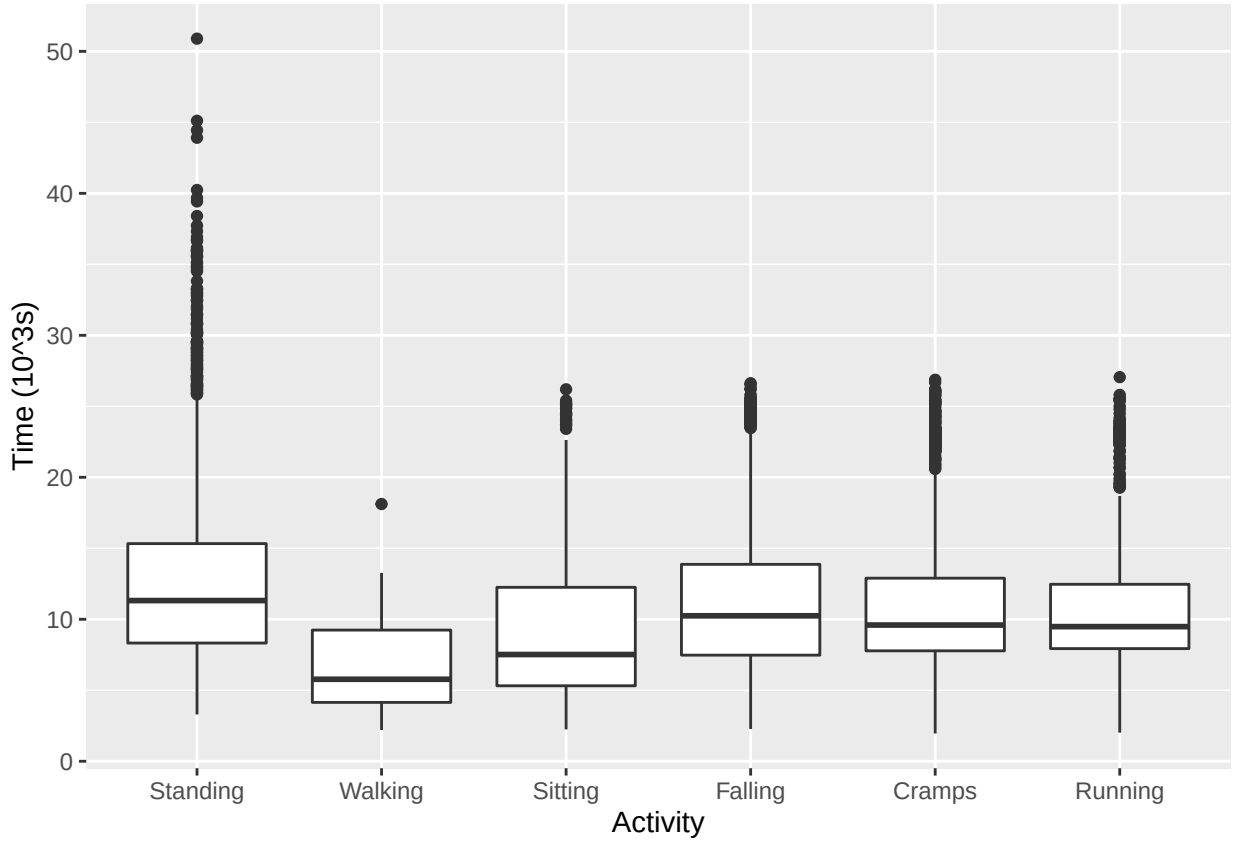
```
Taov <- aov(TIME ~ ACTIVITY, data = trainset)
TukeyHSD(Taov)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = TIME ~ ACTIVITY, data = trainset)
##
## $ACTIVITY
##                         diff        lwr        upr     p adj
## Walking-Standing -5907.3714 -6672.8894 -5141.8534 0.0000000
## Sitting-Standing -3367.4254 -3771.6692 -2963.1816 0.0000000
## Falling-Standing -1230.8399 -1593.2529  -868.4269 0.0000000
## Cramps-Standing  -1847.0300 -2212.1666 -1481.8934 0.0000000
## Running-Standing -2138.9502 -2602.0785 -1675.8220 0.0000000
## Sitting-Walking   2539.9460  1743.4315  3336.4606 0.0000000
## Falling-Walking   4676.5315  3900.4094  5452.6537 0.0000000
## Cramps-Walking    4060.3415  3282.9438  4837.7391 0.0000000
## Running-Walking   3768.4212  2940.4658  4596.3766 0.0000000
## Falling-Sitting   2136.5855  1712.6034  2560.5676 0.0000000
## Cramps-Sitting    1520.3954  1094.0829  1946.7079 0.0000000
## Running-Sitting   1228.4752   715.7322  1741.2182 0.0000000
## Cramps-Falling    -616.1901 -1003.0654  -229.3147 0.0000832
## Running-Falling   -908.1103 -1388.5638  -427.6568 0.0000011
```

```
## Running-Cramps    -291.9202  -774.4315   190.5910 0.5155417
```

2.2.1.1 boxplot

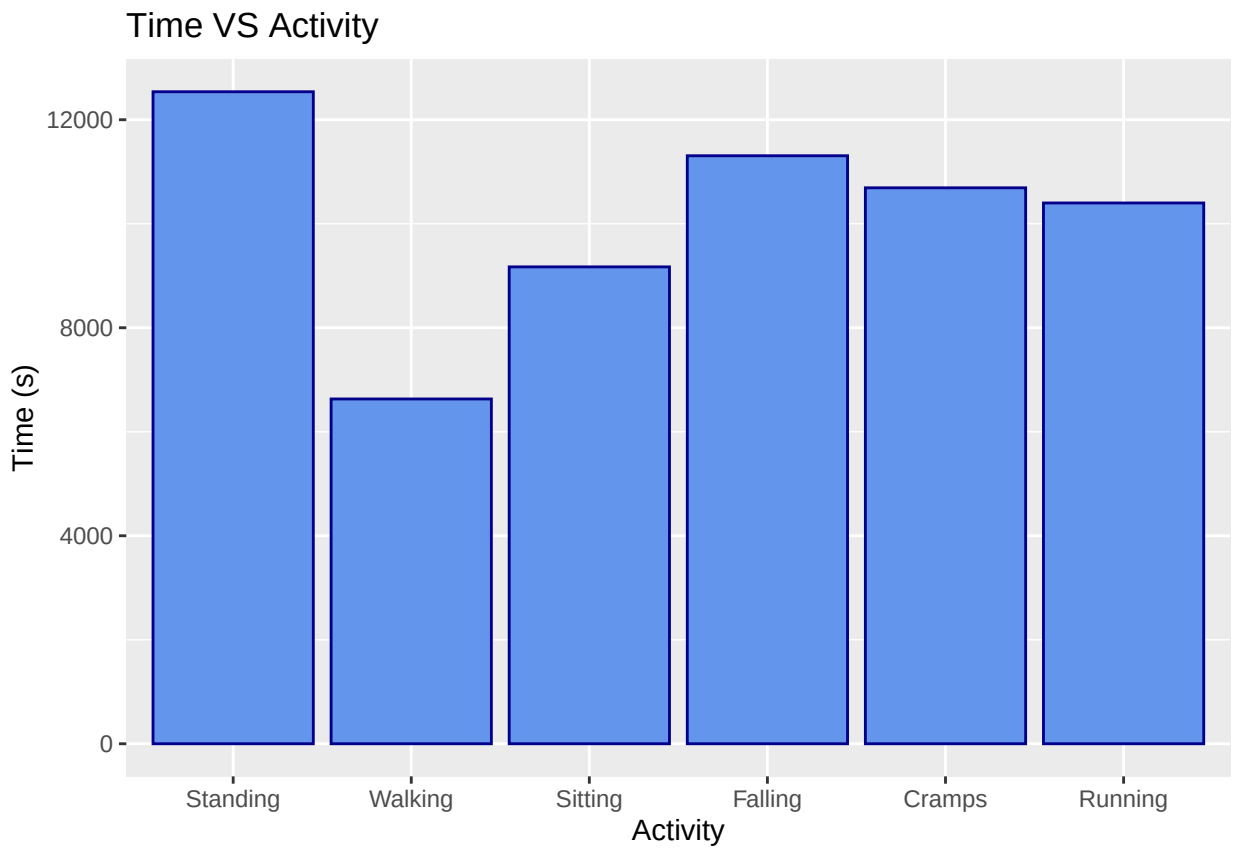
```
trainset%>% mutate(TIME_=TIME/1000) %>%
  ggplot(aes(ACTIVITY, TIME_)) +
  labs(y = "Time (10^3s)", x = "Activity") +
  geom_boxplot()
```



2.2.1.2 bar plot

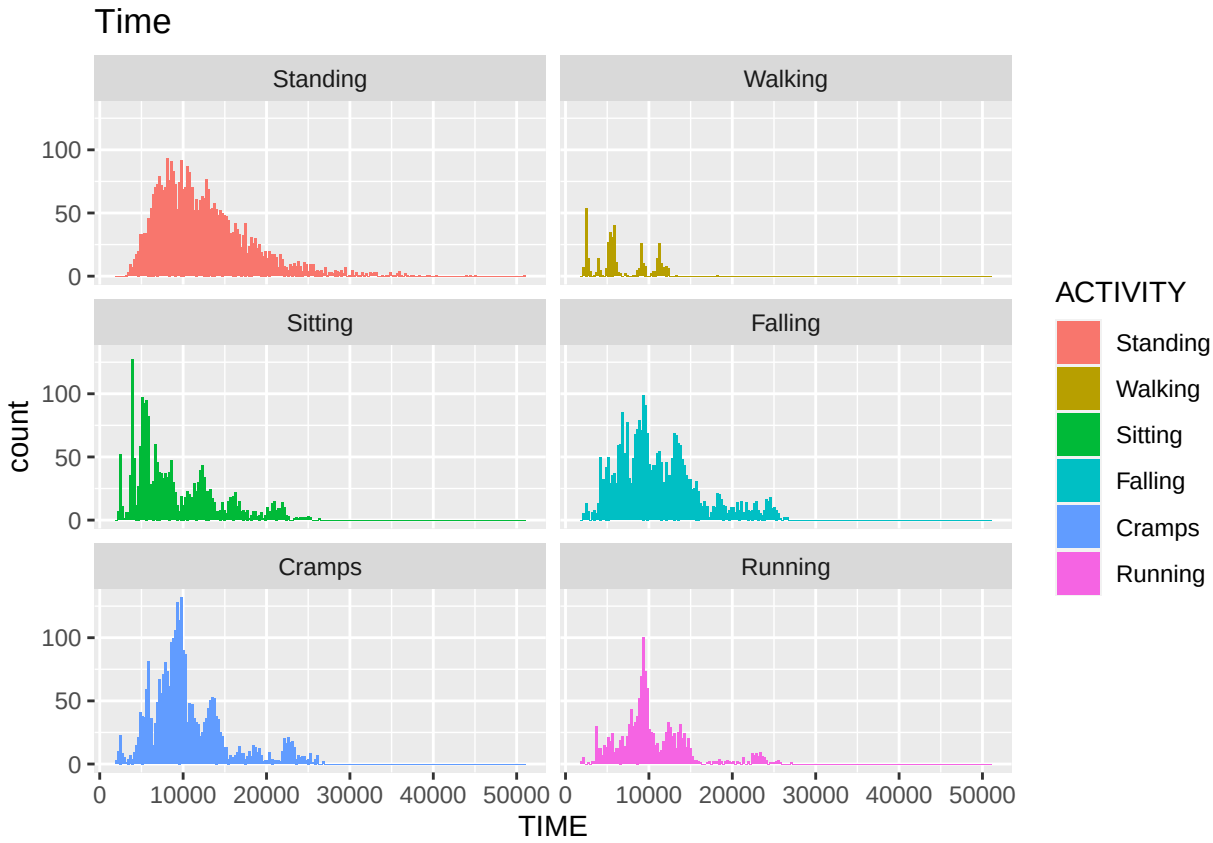
```
trainset %>%
  na.omit() %>%
  group_by(ACTIVITY) %>% # group data by activity
  summarise(M_TIME = mean(TIME)) %>% # mean time
  ggplot(aes(x = ACTIVITY, y = M_TIME)) +
  geom_bar(stat = "identity", fill="cornflowerblue",color="darkblue") +
  labs(y = "Time (s)", x = "Activity") +
  ggtitle("Time VS Activity")
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



2.2.1.3 histogram

```
trainset %>% ggplot(aes(x = TIME, fill = ACTIVITY)) +
  geom_histogram(bins = 200) +
  ggtitle("Time") +
  facet_wrap(~ACTIVITY, ncol = 2)
```



2.2.2 Sugar Level VS Activity Sugar level differs significantly among activity types, except Running-Sitting and Cramps-Falling.

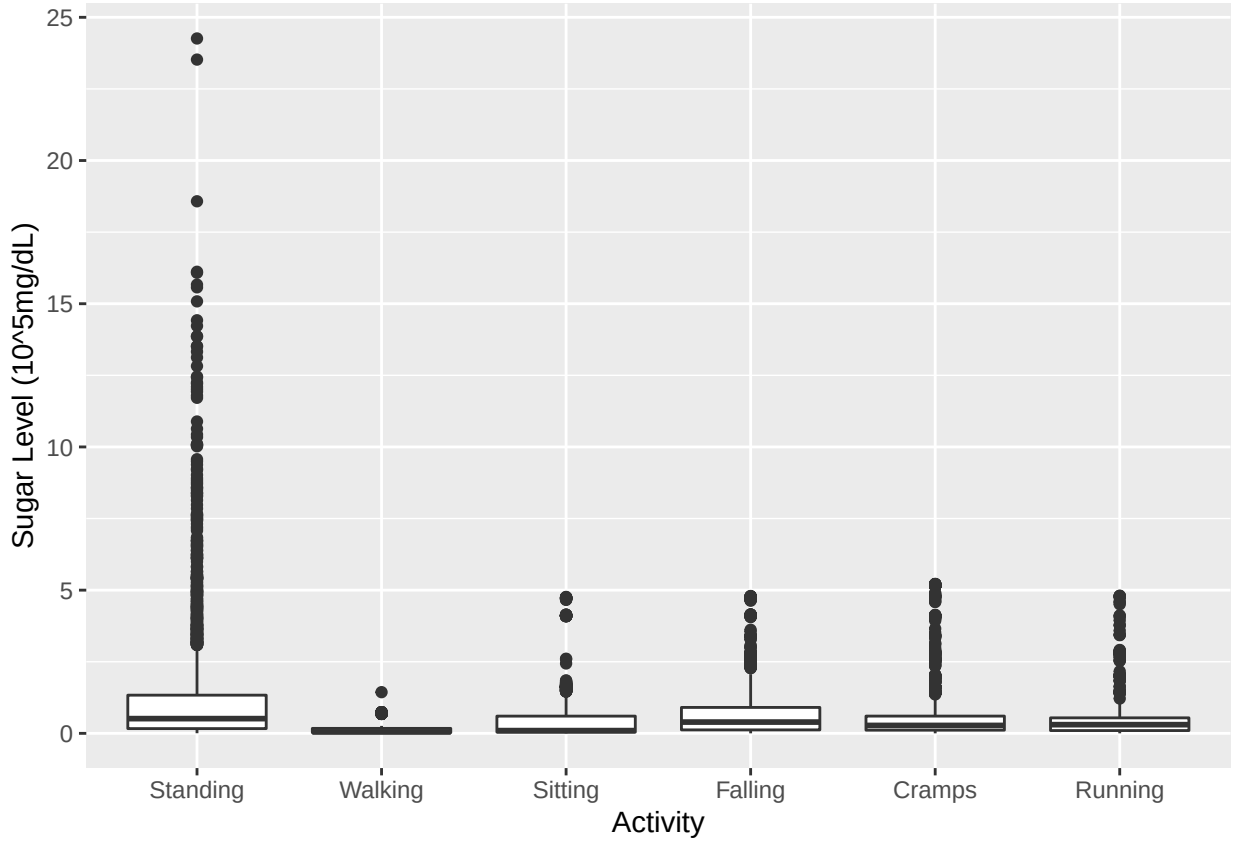
```
SLaov <- aov(SL ~ ACTIVITY, data = trainset)
TukeyHSD(SLaov)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = SL ~ ACTIVITY, data = trainset)
##
## $ACTIVITY
##                          diff         lwr        upr     p adj
## Walking-Standing -95348.3093 -114288.896 -76407.723 0.0000000
## Sitting-Standing -63342.4912  -73344.365 -53340.618 0.0000000
## Falling-Standing -40510.3344  -49477.224 -31543.445 0.0000000
## Cramps-Standing  -47354.1826  -56388.459 -38319.906 0.0000000
## Running-Standing -62410.0136  -73868.817 -50951.210 0.0000000
## Sitting-Walking   32005.8181   12298.309  51713.327 0.0000545
## Falling-Walking   54837.9749   35635.019  74040.931 0.0000000
## Cramps-Walking    47994.1268   28759.611  67228.642 0.0000000
## Running-Walking   32938.2957   12452.872  53423.720 0.0000679
## Falling-Sitting   22832.1568   12341.914  33322.399 0.0000000
## Cramps-Sitting    15988.3086    5440.406  26536.211 0.0002273
## Running-Sitting     932.4776  -11753.903  13618.858 0.9999448
## Cramps-Falling    -6843.8481  -16415.989   2728.292 0.3208419
## Running-Falling  -21899.6792  -33787.147 -10012.212 0.0000023
```

```
## Running-Cramps   -15055.8310  -26994.212  -3117.450 0.0044083
```

2.2.2.1 boxplot

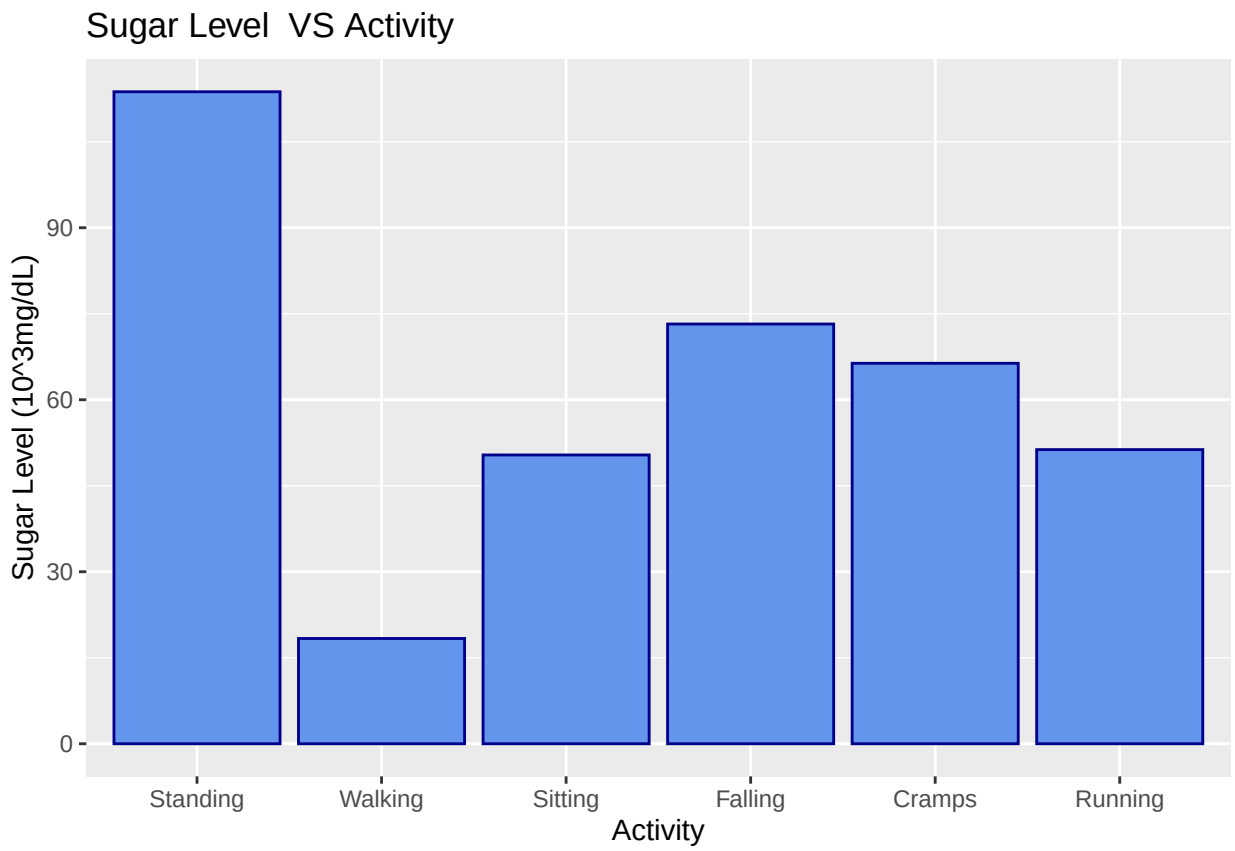
```
trainset%>%
  mutate(SL_=SL/100000)%>%
  ggplot(aes(ACTIVITY, SL_)) +
  labs(y = "Sugar Level (10^5mg/dL)", x = "Activity") +
  geom_boxplot()
```



2.2.2.2 bar plot

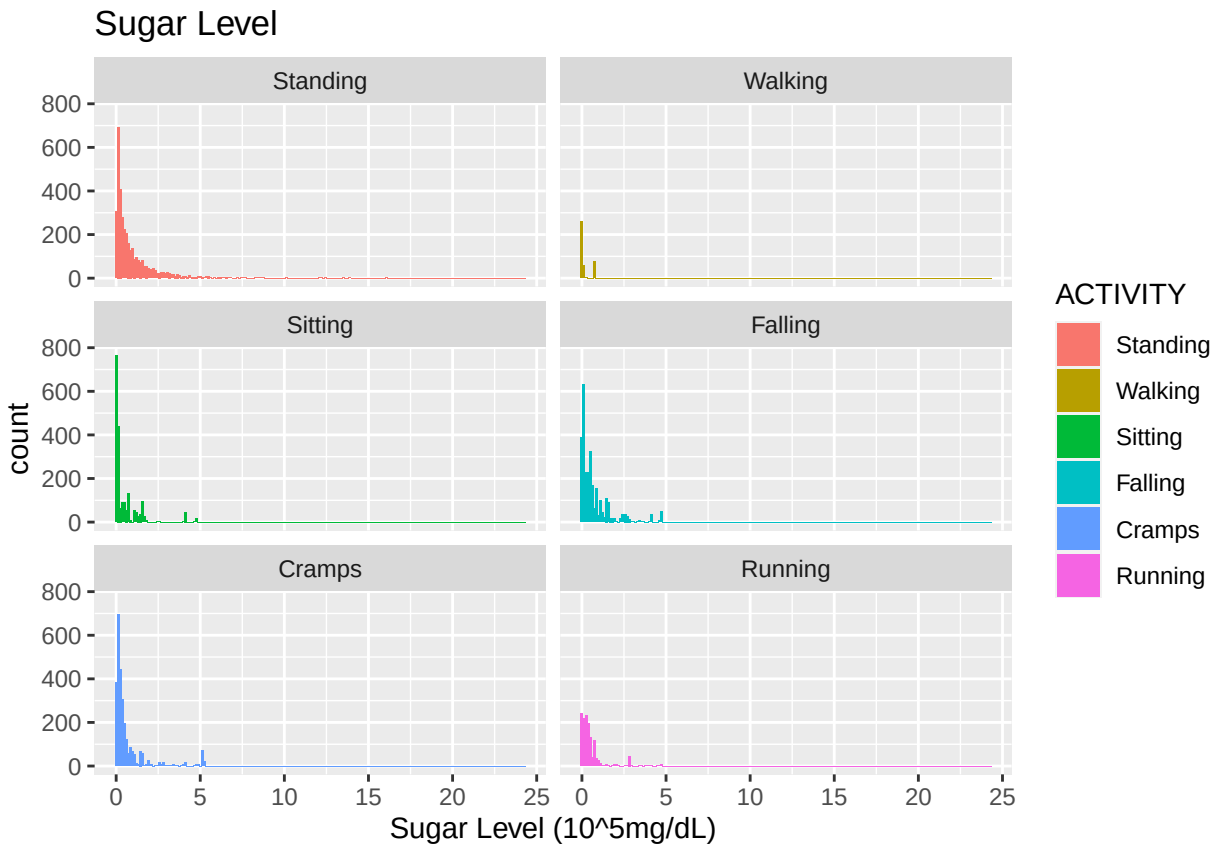
```
trainset %>%
  na.omit() %>%
  group_by(ACTIVITY) %>% # group data by activity
  summarise(M_SL = mean(SL)/1000) %>% # mean SL
  ggplot(aes(x = ACTIVITY, y = M_SL)) +
  geom_bar(stat = "identity", fill="cornflowerblue",color="darkblue") +
  labs(y = "Sugar Level (10^3mg/dL)", x = "Activity") +
  ggtitle("Sugar Level  VS Activity")
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



2.2.2.3 histogram

```
trainset %>%  mutate(SL=SL/100000)%>%
  ggplot(aes(x = SL, fill = ACTIVITY)) +
  geom_histogram(bins = 200) +
  ggtitle("Sugar Level") +
  labs(x = "Sugar Level (10^5mg/dL)")+
  facet_wrap(~ACTIVITY, ncol = 2)
```



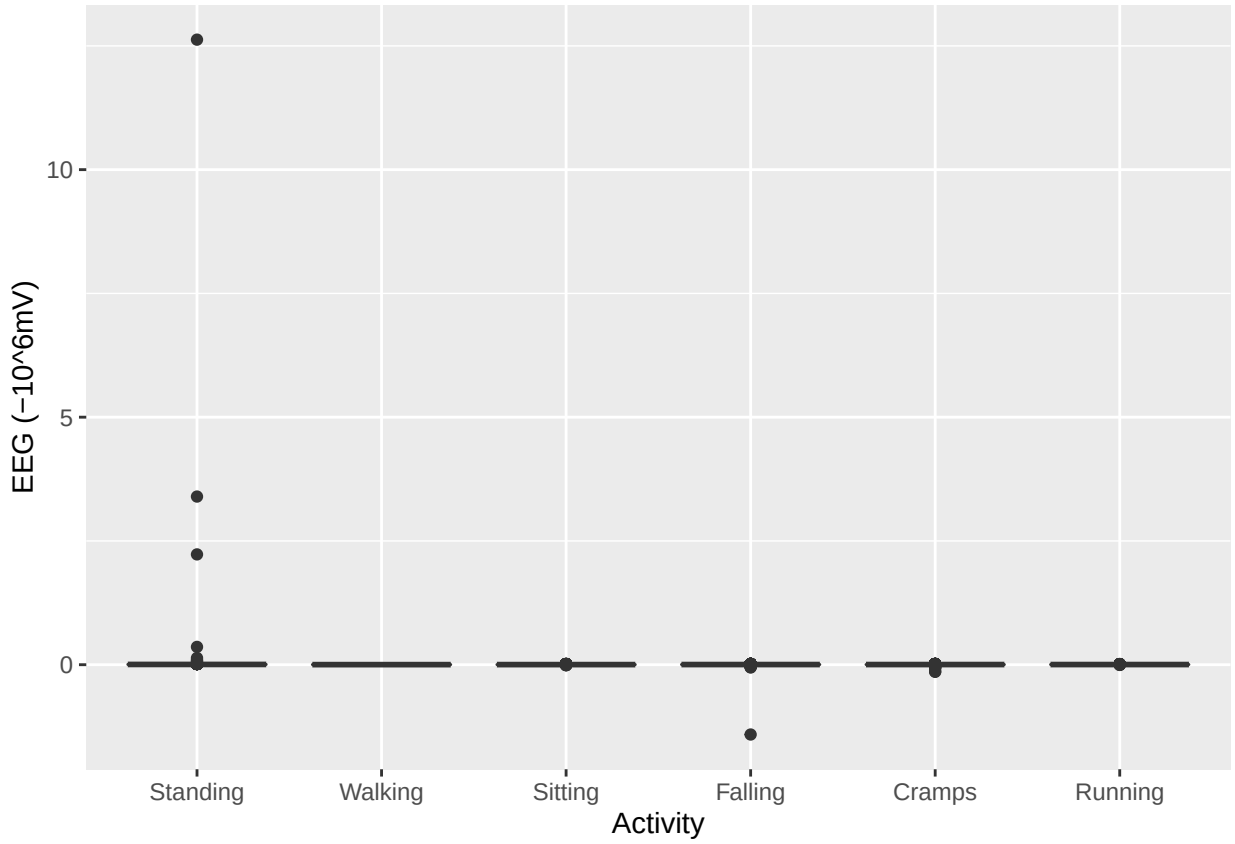
2.2.3 EEG VS activity Heart rate differs significantly among activity types, except Running-Cramps.

```
HRaov <- aov(HR ~ ACTIVITY, data = trainset)
TukeyHSD(HRaov)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = HR ~ ACTIVITY, data = trainset)
##
## $ACTIVITY
##                           diff        lwr         upr     p adj
## Walking-Standing -137.765462 -156.61921 -118.911713 0.0000000
## Sitting-Standing  -88.709432  -98.66545  -78.753414 0.0000000
## Falling-Standing  -41.093196  -50.01897  -32.167417 0.0000000
## Cramps-Standing   -55.085551  -64.07841  -46.092694 0.0000000
## Running-Standing  -62.979579  -74.38585  -51.573311 0.0000000
## Sitting-Walking    49.056031   29.43888   68.673185 0.0000000
## Falling-Walking    96.672266   77.55735  115.787182 0.0000000
## Cramps-Walking     82.679911   63.53358  101.826241 0.0000000
## Running-Walking    74.785883   54.39438   95.177387 0.0000000
## Falling-Sitting    47.616236   37.17409   58.058383 0.0000000
## Cramps-Sitting     33.623881   23.12434   44.123423 0.0000000
## Running-Sitting    25.729853   13.10164   38.358070 0.0000001
## Cramps-Falling    -13.992355  -23.52061   -4.464100 0.0004111
## Running-Falling   -21.886383  -33.71935  -10.053416 0.0000020
## Running-Cramps     -7.894028  -19.77767    3.989619 0.4062594
```

2.2.3.1 boxplot

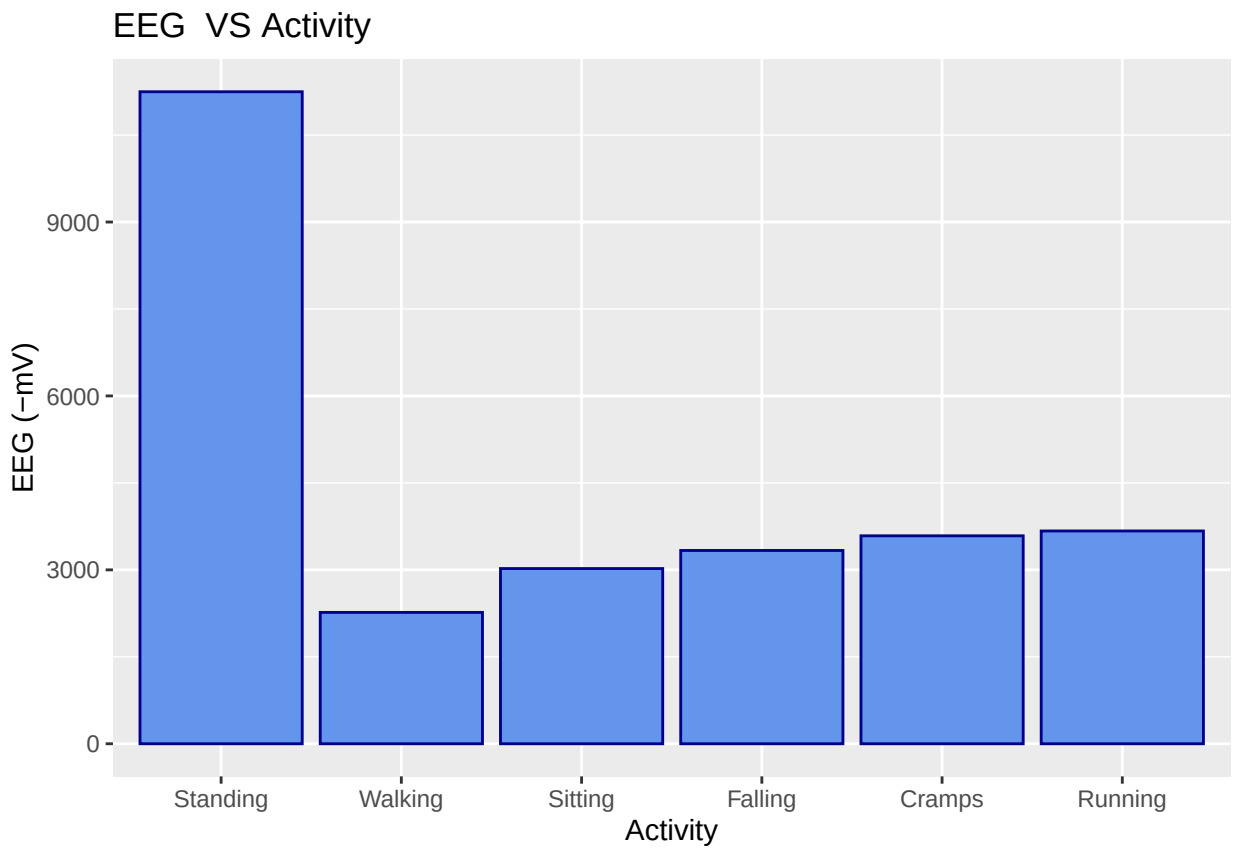
```
trainset%>%
  mutate(EEG_ = EEG*(-10^-6)) %>%
  ggplot(aes(ACTIVITY, EEG_)) +
  labs(y = "EEG (-10^6mV)", x = "Activity") +
  geom_boxplot()
```



2.2.3.2 bar plot

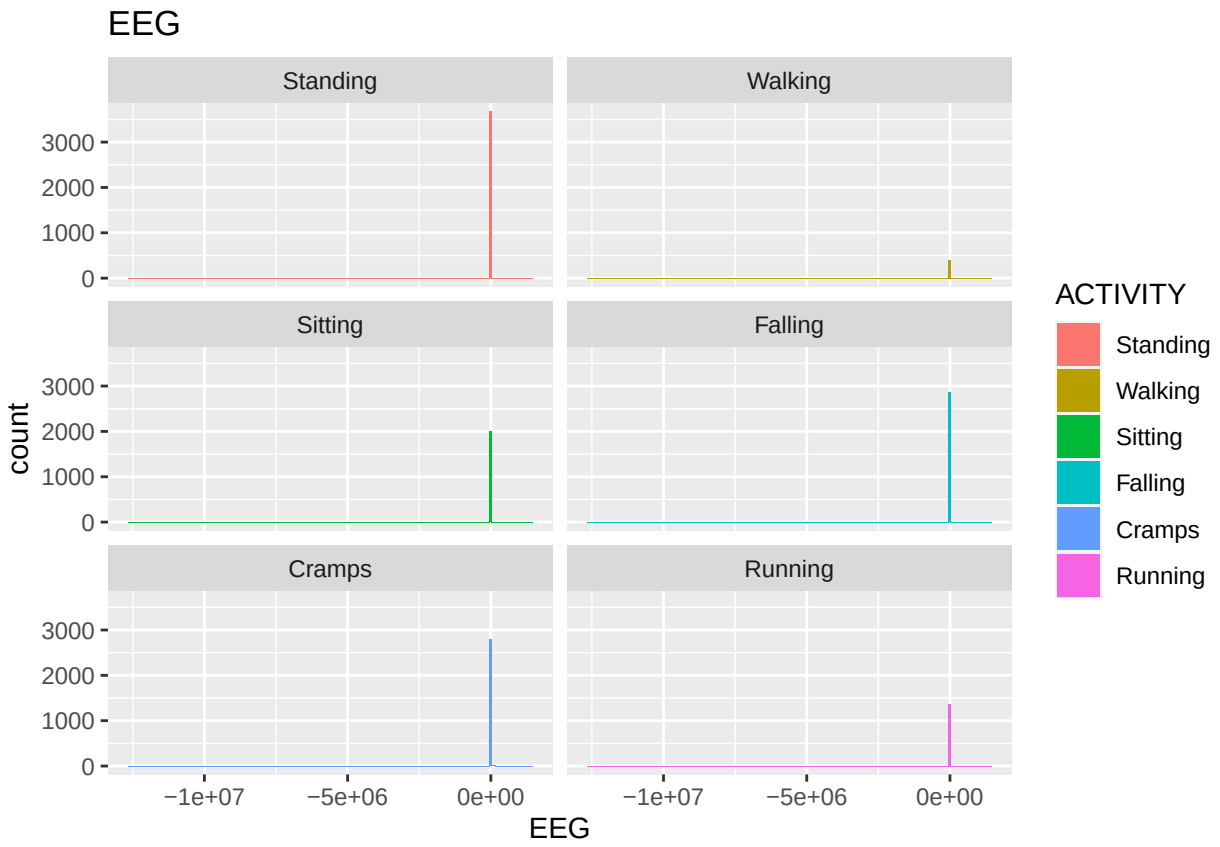
```
trainset %>%
  na.omit() %>%
  group_by(ACTIVITY) %>% # group data by activity
  summarise(M_EEG = mean(EEG*(-1))) %>% # mean EEG
  ggplot(aes(x = ACTIVITY, y = M_EEG)) +
  geom_bar(stat = "identity", fill="cornflowerblue",color="darkblue") +
  labs(y = "EEG (-mV)", x = "Activity") +
  ggtitle("EEG  VS Activity")
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



2.2.3.3 histogram

```
trainset %>% ggplot(aes(x = EEG, fill = ACTIVITY)) +
  geom_histogram(bins = 200) +
  ggtitle("EEG") +
  facet_wrap(~ACTIVITY, ncol = 2)
```



2.2.4 Blood Pressure VS Activity Blood pressure differs significantly among activity types, except Running-Cramps.

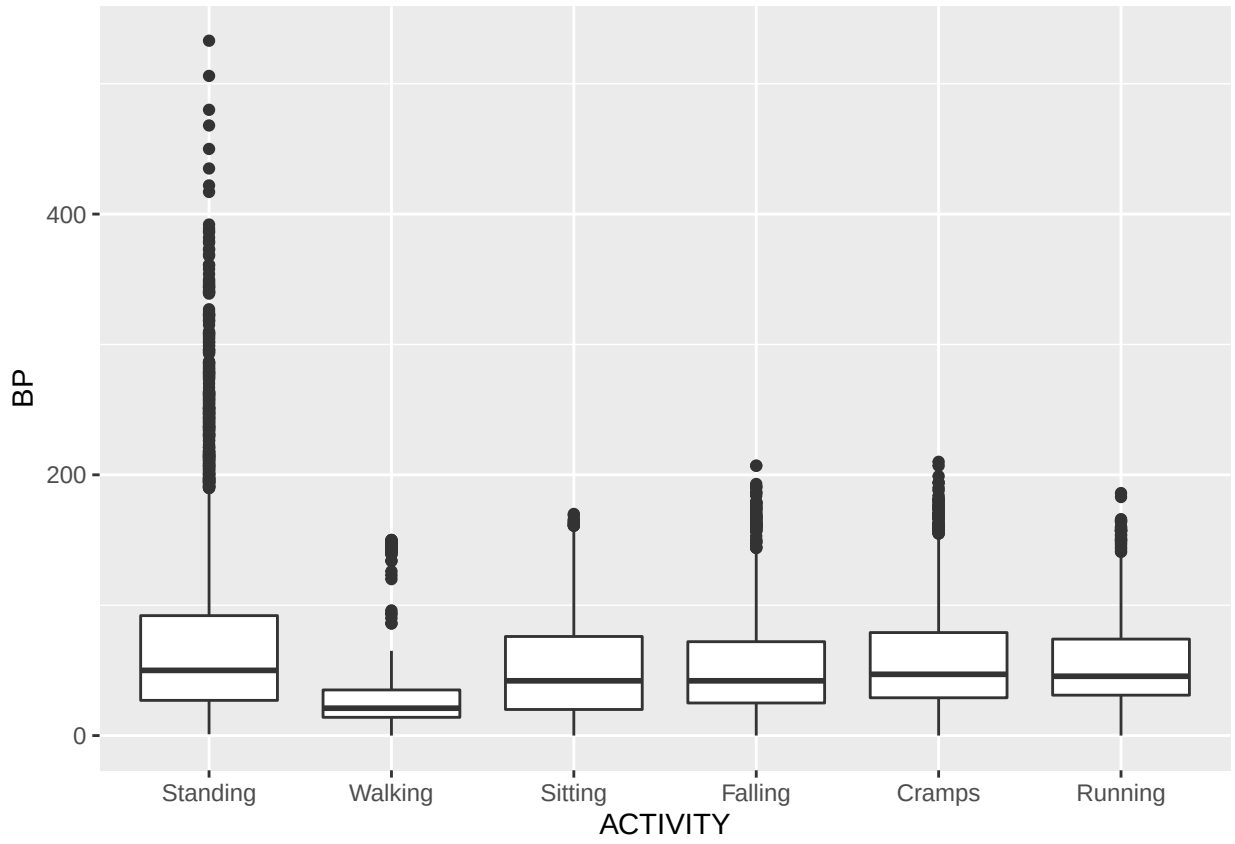
```
BPaov <- aov(BP ~ ACTIVITY, data = trainset)
TukeyHSD(BPaov)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = BP ~ ACTIVITY, data = trainset)
##
## $ACTIVITY
##                           diff          lwr        upr     p adj
## Walking-Standing -33.02732926 -40.19091019 -25.863748 0.0000000
## Sitting-Standing -22.12769429 -25.91053548 -18.344853 0.0000000
## Falling-Standing -22.11467101 -25.50606748 -18.723275 0.0000000
## Cramps-Standing  -15.92804170 -19.34492498 -12.511158 0.0000000
## Running-Standing -17.37342698 -21.70729833 -13.039556 0.0000000
## Sitting-Walking   10.89963497   3.44599390  18.353276 0.0004437
## Falling-Walking   10.91265825   3.64984559  18.175471 0.0002689
## Cramps-Walking    17.09928756   9.82453883  24.374036 0.0000000
## Running-Walking   15.65390228   7.90604331  23.401761 0.0000001
## Falling-Sitting    0.01302328  -3.95452552   3.980572 1.0000000
## Cramps-Sitting     6.19965259   2.21029615  10.189009 0.0001388
## Running-Sitting    4.75426731  -0.04389008   9.552425 0.0537885
## Cramps-Falling     6.18662931   2.56631891   9.806940 0.0000167
## Running-Falling    4.74124403   0.24524615   9.237242 0.0317866
```

```
## Running-Cramps    -1.44538528  -5.96063923   3.069869 0.9434998
```

2.2.4.1 boxplot

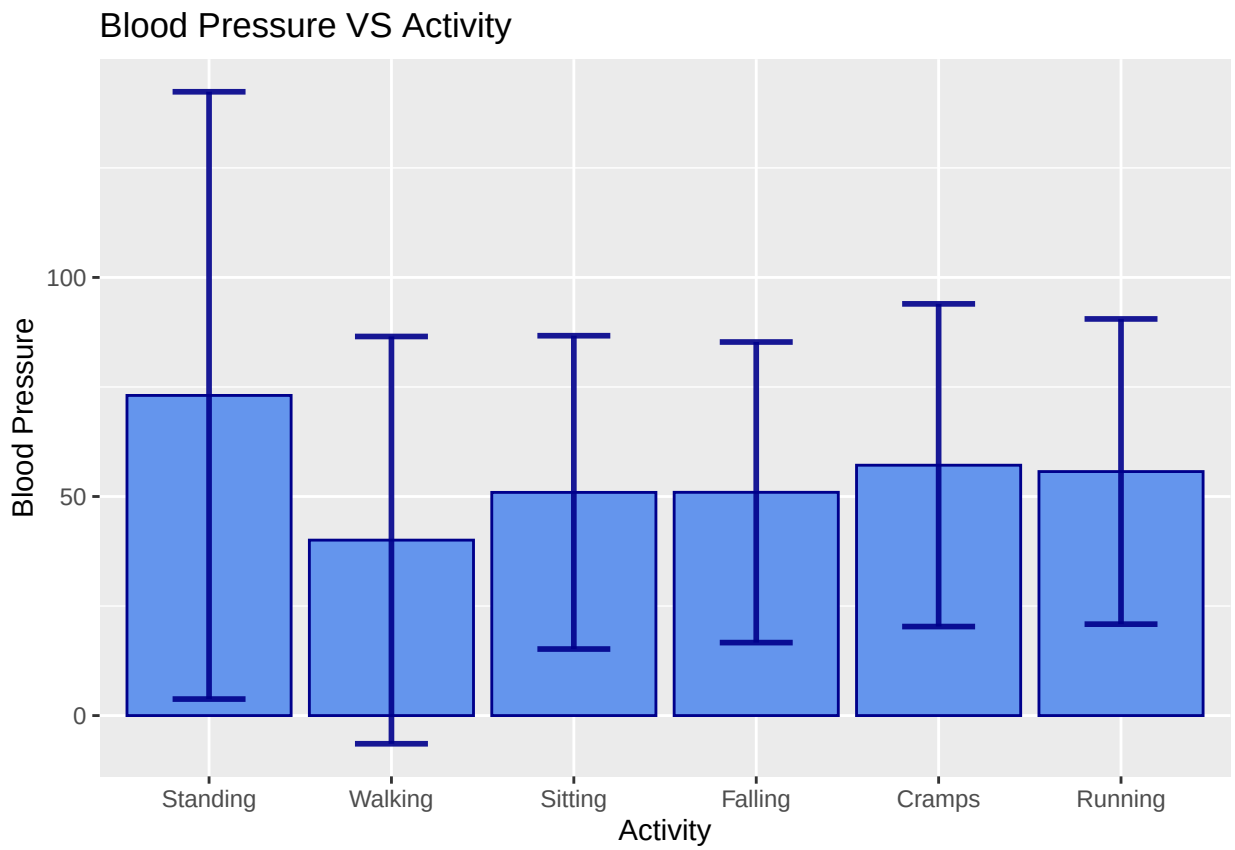
```
trainset%>%
  ggplot(aes(ACTIVITY, BP)) +
  geom_boxplot()
```



2.2.4.2 barplot (with error bar)

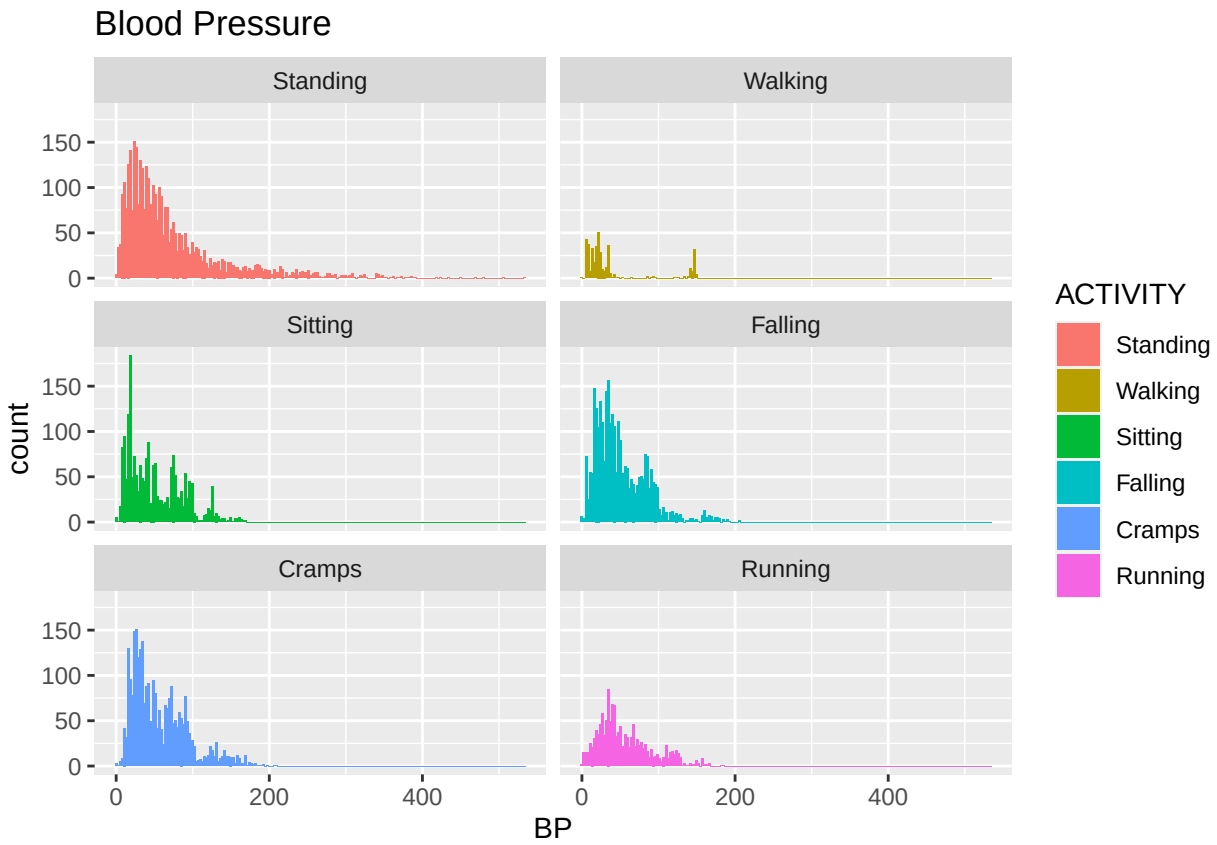
```
trainset %>%
  na.omit() %>%
  group_by(ACTIVITY) %>% # group data by activity
  summarise(M_BP = mean(BP), sd = sd(BP) ) %>% # count
  ggplot(aes(x = ACTIVITY, y = M_BP)) +
  geom_bar(stat = "identity", fill="cornflowerblue",color="darkblue") +
  geom_errorbar(aes(ymin=M_BP-sd, ymax=M_BP+sd),
                width=0.4, colour="darkblue", alpha=0.9, size=1)+
  labs(y = "Blood Pressure", x = "Activity") +
  ggtitle("Blood Pressure VS Activity")
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



2.2.4.3 histogram

```
trainset %>% ggplot(aes(x = BP, fill = ACTIVITY)) +
  geom_histogram(bins = 200) +
  ggtitle("Blood Pressure") +
  facet_wrap(~ACTIVITY, ncol = 2)
```



2.2.5 Heart Rate VS Activity Heart rate differs significantly among activity types, except Running-Cramps.

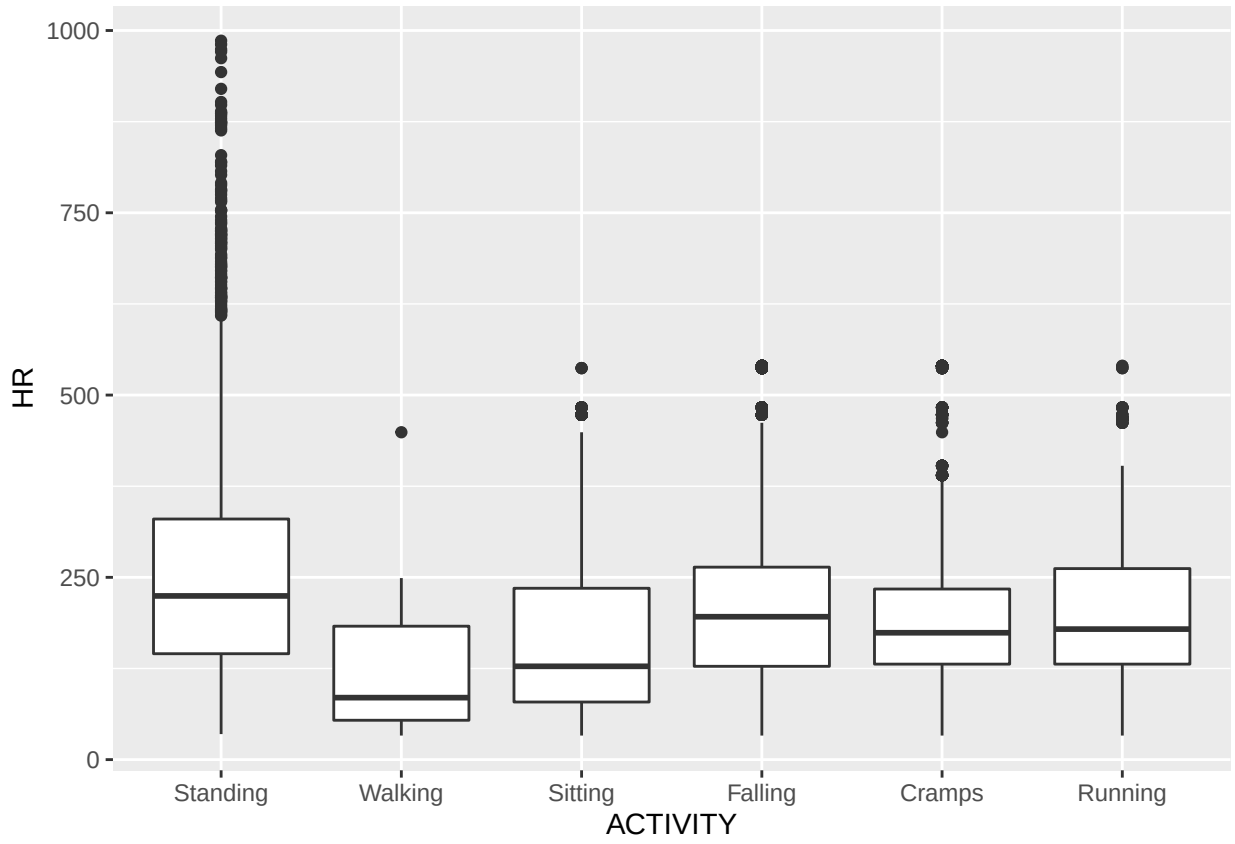
```
HRaov <- aov(HR ~ ACTIVITY, data = trainset)
TukeyHSD(HRaov)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = HR ~ ACTIVITY, data = trainset)
##
## $ACTIVITY
##                          diff        lwr         upr     p adj
## Walking-Standing -137.765462 -156.61921 -118.911713 0.0000000
## Sitting-Standing  -88.709432  -98.66545  -78.753414 0.0000000
## Falling-Standing  -41.093196  -50.01897  -32.167417 0.0000000
## Cramps-Standing   -55.085551  -64.07841  -46.092694 0.0000000
## Running-Standing  -62.979579  -74.38585  -51.573311 0.0000000
## Sitting-Walking    49.056031   29.43888   68.673185 0.0000000
## Falling-Walking    96.672266   77.55735  115.787182 0.0000000
## Cramps-Walking     82.679911   63.53358  101.826241 0.0000000
## Running-Walking    74.785883   54.39438   95.177387 0.0000000
## Falling-Sitting    47.616236   37.17409   58.058383 0.0000000
## Cramps-Sitting     33.623881   23.12434   44.123423 0.0000000
## Running-Sitting    25.729853   13.10164   38.358070 0.0000001
## Cramps-Falling    -13.992355  -23.52061   -4.464100 0.0004111
## Running-Falling   -21.886383  -33.71935  -10.053416 0.0000020
```

```
## Running-Cramps     -7.894028  -19.77767    3.989619 0.4062594
```

2.2.5.1 boxplot

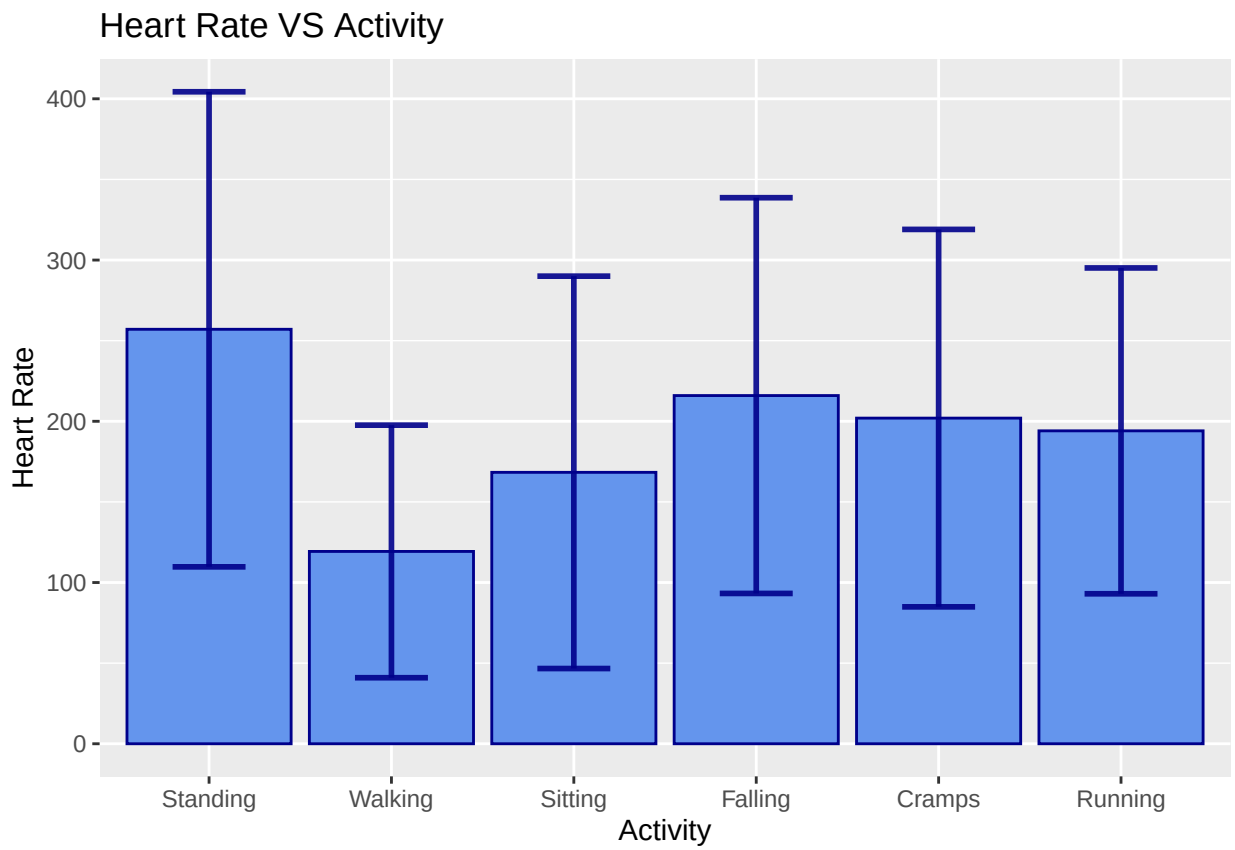
```
trainset%>%
  ggplot(aes(ACTIVITY, HR)) +
  geom_boxplot()
```



2.2.5.2 barplot (with error bar)

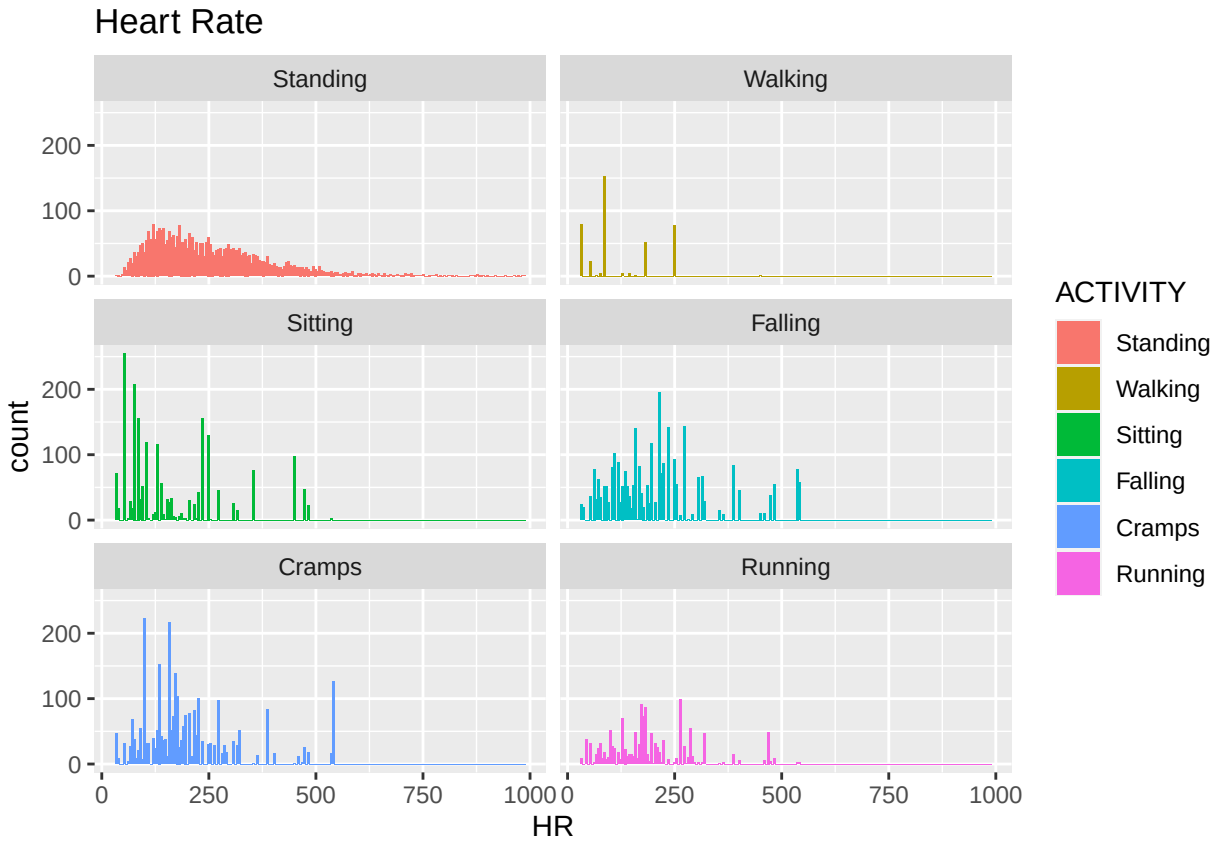
```
trainset %>%
  na.omit() %>%
  group_by(ACTIVITY) %>% # gHRroup data by activity
  summarise(M_HR = mean(HR), sd = sd(HR)) %>% # count
  ggplot(aes(x = ACTIVITY, y = M_HR)) +
  geom_bar(stat = "identity", fill="cornflowerblue",color="darkblue") +
  geom_errorbar(aes(ymin=M_HR-sd, ymax=M_HR+sd),
                width=0.4, colour="darkblue", alpha=0.9, size=1)+
  labs(y = "Heart Rate", x = "Activity") +
  ggtitle("Heart Rate VS Activity")
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



2.2.5.3 histogram

```
trainset %>% ggplot(aes(x = HR, fill = ACTIVITY)) +
  geom_histogram(bins = 200) +
  ggtitle("Heart Rate") +
  facet_wrap(~ACTIVITY, ncol = 2)
```



2.2.6 Circulation VS Activity Blood circulation level differs significantly among activity types, except Running-Sitting and Cramps-Falling.

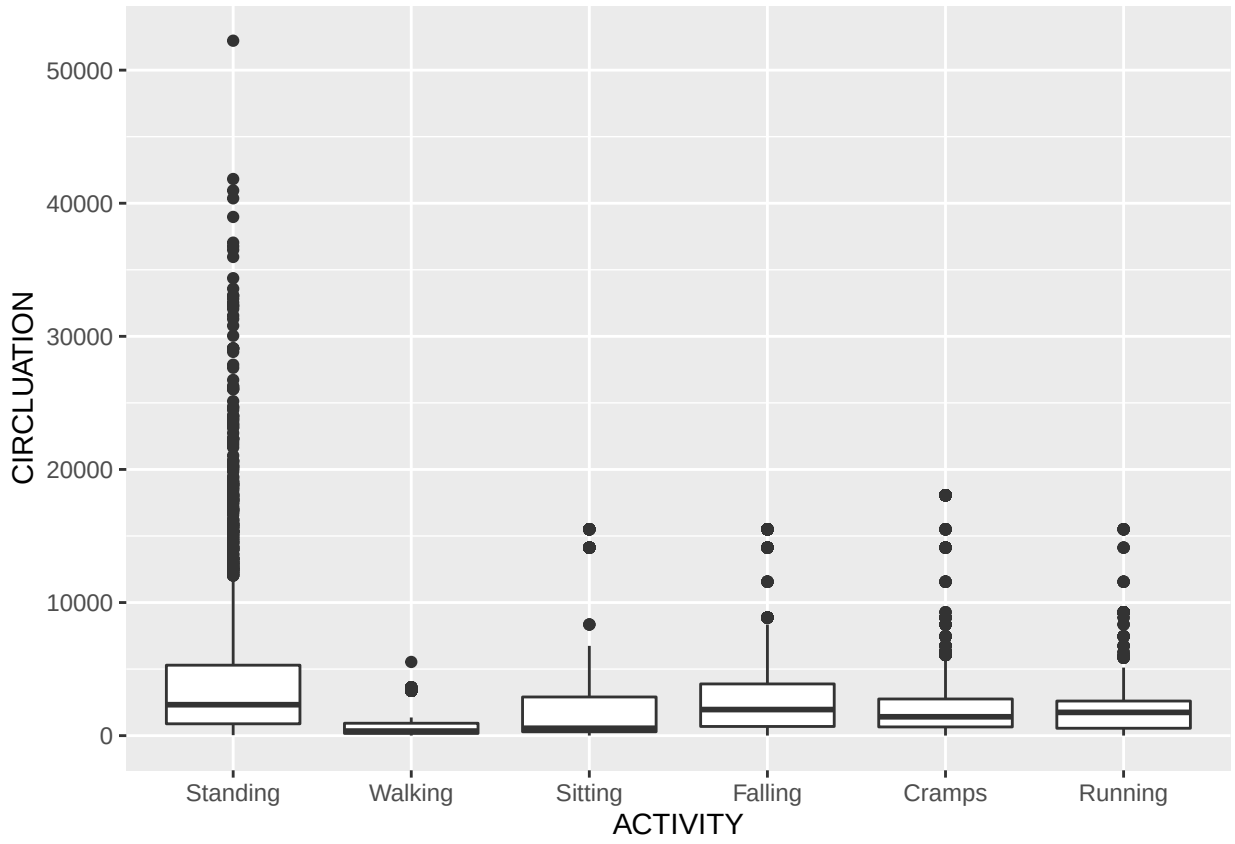
```
CIRaov <- aov(CIRCLUATION ~ ACTIVITY, data = trainset)
TukeyHSD(CIRaov)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = CIRCLUATION ~ ACTIVITY, data = trainset)
##
## $ACTIVITY
##                          diff        lwr         upr     p adj
## Walking-Standing -3053.1906 -3618.0536 -2488.32758 0.0000000
## Sitting-Standing -1941.3238 -2239.6086 -1643.03909 0.0000000
## Falling-Standing -1076.2526 -1343.6711  -808.83402 0.0000000
## Cramps-Standing  -1304.0743 -1573.5025 -1034.64608 0.0000000
## Running-Standing -1810.8052 -2152.5398 -1469.07060 0.0000000
## Sitting-Walking   1111.8667   524.1319  1699.60156 0.0000011
## Falling-Walking   1976.9380  1404.2504  2549.62563 0.0000000
## Cramps-Walking    1749.1163  1175.4875  2322.74507 0.0000000
## Running-Walking   1242.3854   631.4509  1853.31988 0.0000001
## Falling-Sitting    865.0713   552.2220  1177.92061 0.0000000
## Cramps-Sitting     637.2495   322.6806   951.81845 0.0000001
## Running-Sitting    130.5186  -247.8259   508.86315 0.9234277
## Cramps-Falling    -227.8217  -513.2906    57.64713 0.2045240
## Running-Falling   -734.5526 -1089.0713  -380.03403 0.0000001
```

```
## Running-Cramps    -506.7309  -862.7679  -150.69391 0.0007123
```

2.2.6.1 boxplot

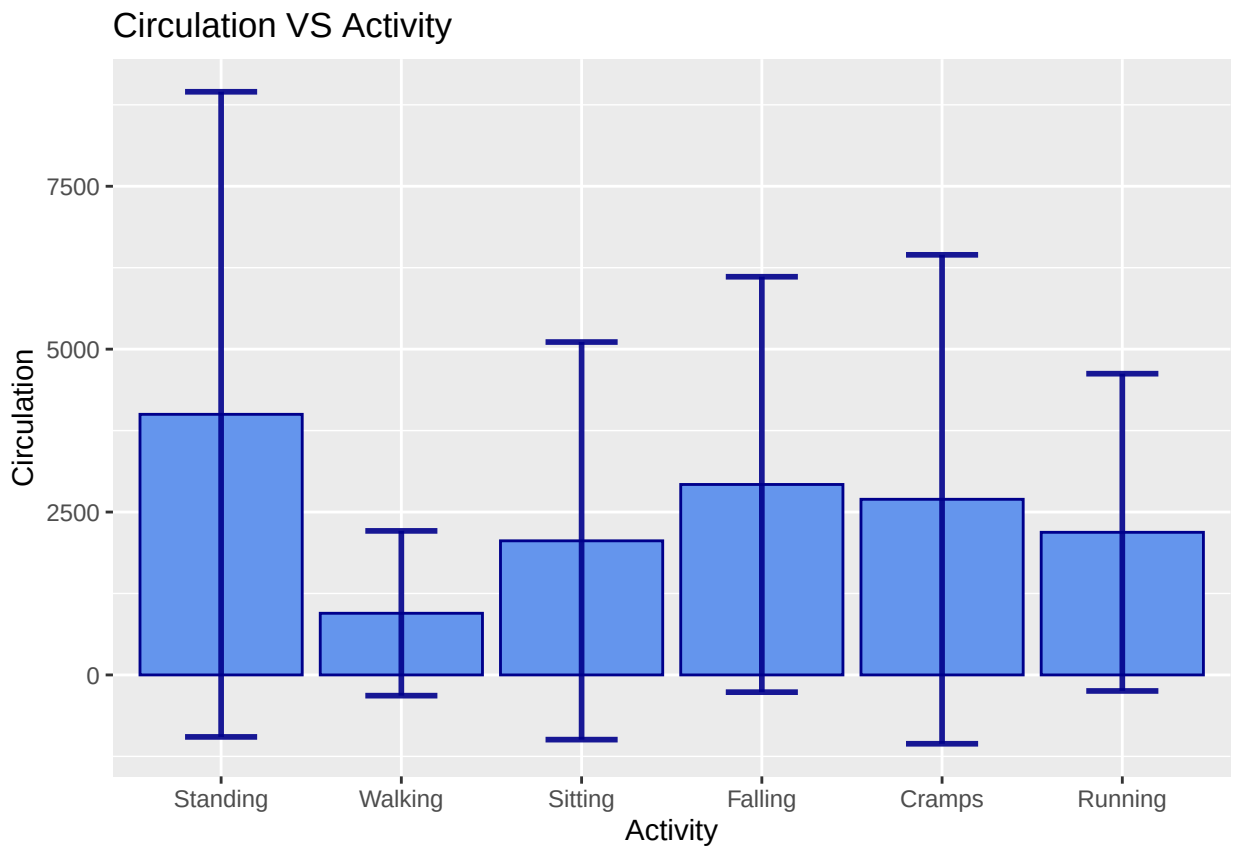
```
trainset%>%
  ggplot(aes(ACTIVITY, CIRCLUATION)) +
  geom_boxplot()
```



2.2.6.2 barplot (with error bar)

```
trainset %>%
  na.omit() %>%
  group_by(ACTIVITY) %>% # group data by activity
  summarise(M_CIRCLUATION = mean(CIRCLUATION), sd = sd(CIRCLUATION) ) %>% # count
  ggplot(aes(x = ACTIVITY, y = M_CIRCLUATION)) +
  geom_bar(stat = "identity", fill="cornflowerblue",color="darkblue") +
  geom_errorbar(aes(ymin=M_CIRCLUATION-sd, ymax=M_CIRCLUATION+sd),
                width=0.4, colour="darkblue", alpha=0.9, size=1)+
  labs(y = "Circulation", x = "Activity") +
  ggtitle("Circulation VS Activity")
```

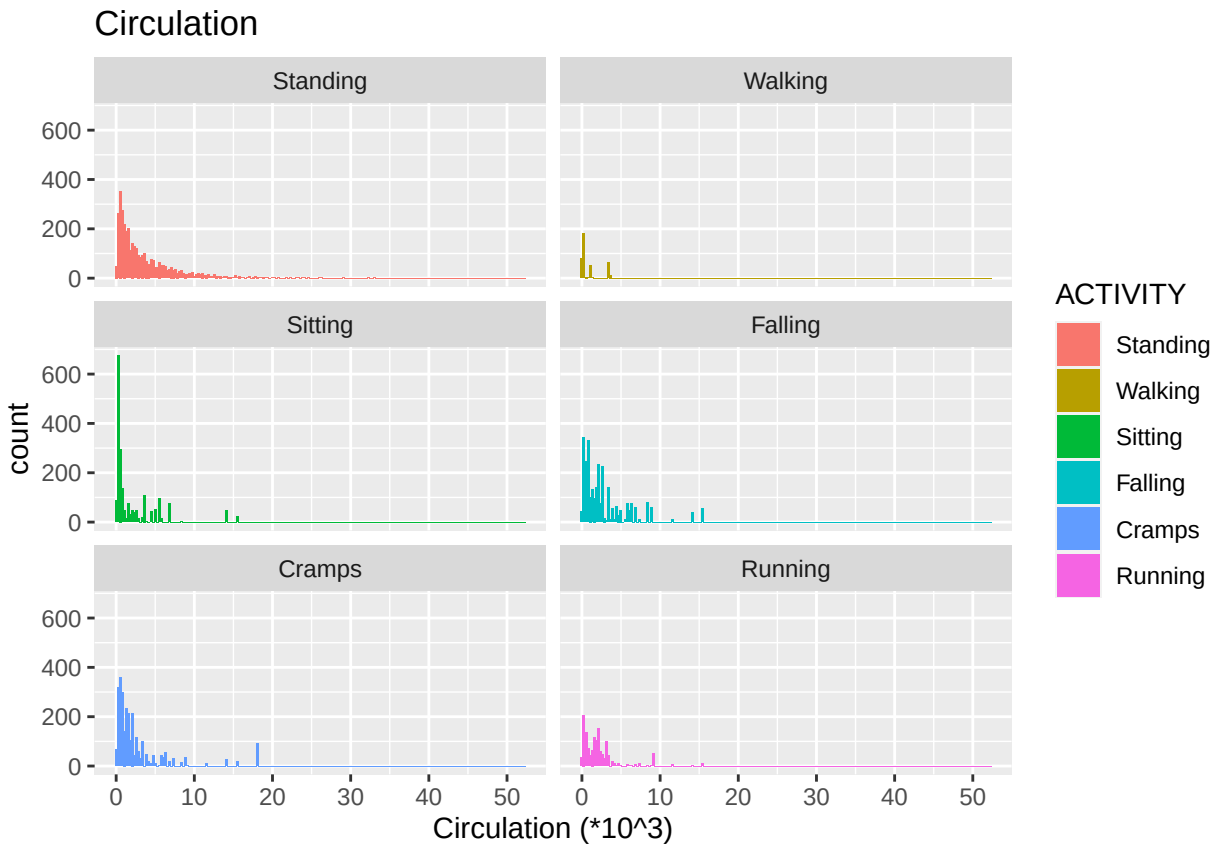
```
## `summarise()` ungrouping output (override with `.groups` argument)
```



/2.2.6.3 barplot (without error bar)/

2.2.6.4 histogram

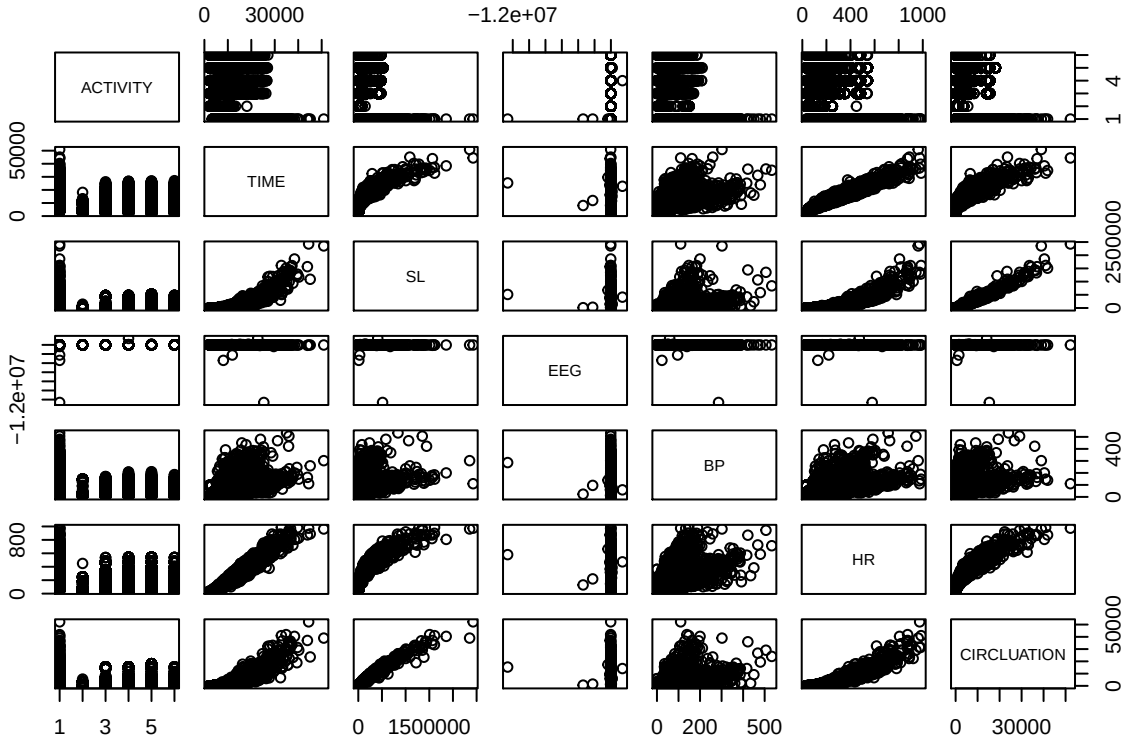
```
trainset %>% mutate(CIRCLUATION = CIRCLUATION/1000) %>%
  ggplot(aes(x = CIRCLUATION, fill = ACTIVITY)) +
  geom_histogram(bins = 200) +
  ggtitle("Circulation ") +
  labs(x = "Circulation (*10^3)") +
  facet_wrap(~ACTIVITY, ncol = 2)
```



2.2.7 Correlation Plots of Variables

Only EEG shows a weak negative correlation with other variables. Time, SL, BP, HR, and Circulation are positively correlated with each other. 2.2.7.1 Correlation plot 1.

```
pairs(trainset)
```

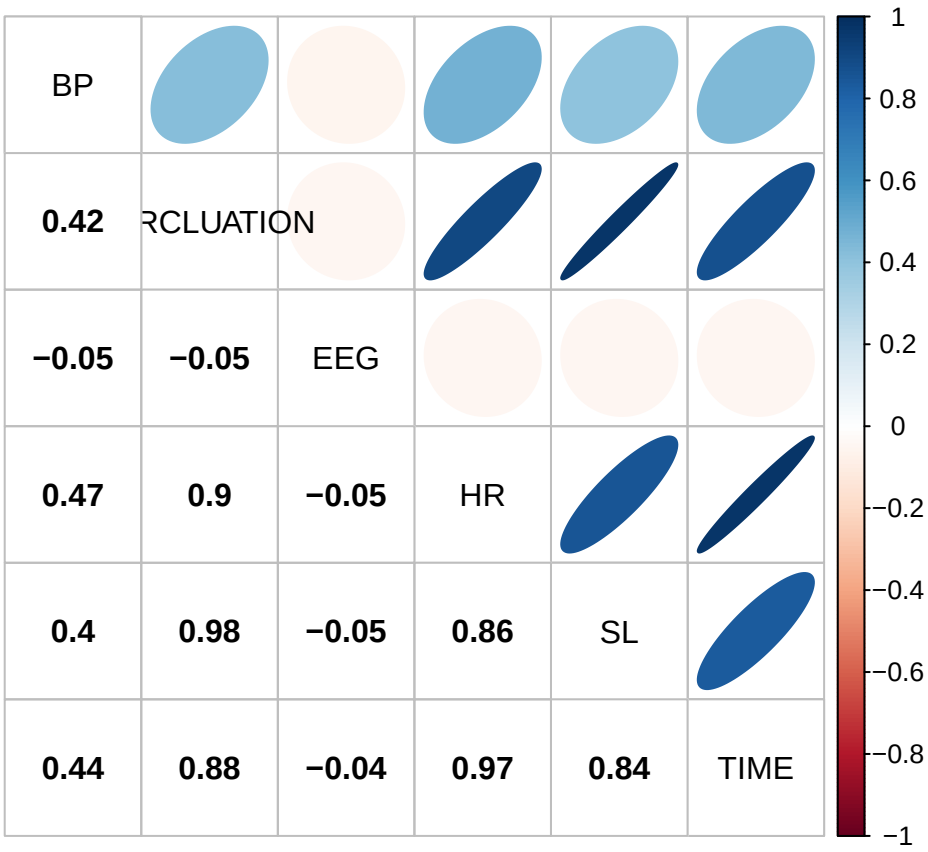



```
correlation <- cor(trainset[,2:7])
correlation
```

```
##                    TIME          SL         EEG          BP          HR
## TIME          1.00000000  0.83959931 -0.04431044  0.44375016  0.97363781
## SL            0.83959931  1.00000000 -0.04781768  0.40239143  0.85569233
## EEG          -0.04431044 -0.04781768  1.00000000 -0.05211849 -0.04585606
## BP            0.44375016  0.40239143 -0.05211849  1.00000000  0.47225802
## HR            0.97363781  0.85569233 -0.04585606  0.47225802  1.00000000
## CIRCLUATION   0.87582806  0.97666268 -0.04647121  0.42217896  0.90326678
##              CIRCLUATION
## TIME          0.87582806
## SL            0.97666268
## EEG          -0.04647121
## BP            0.42217896
## HR            0.90326678
## CIRCLUATION   1.00000000
```

2.2.7.2 Correlation plot2.

```
corrplot.mixed(correlation, lower.col = "black",upper = "ellipse",
               order = "alphabet", number.cex = 1,
               tl.col = "black", sig.level = .05, insig = "blank")
```



Step 3 Model Training

As the previous step showed that when people doing different activities, their biometric indicators were different in most cases. Therefore, we will include all six indicators in the model to predict the activity type.

```
#main package for machine learning algorithms
if(!require(randomForest)) install.packages("randomForest", repos ="http://cran.us.r-project.org")
#working with functions, e.g. using map_df function to pick k in knn
if(!require(purrr)) install.packages("purrr", repos = "http://cran.us.r-project.org")
```

3.0 Preproccessing

```
if(!require(matrixStats)) install.packages("matrixStats", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: matrixStats
```

```
## Warning: package 'matrixStats' was built under R version 4.0.2
```

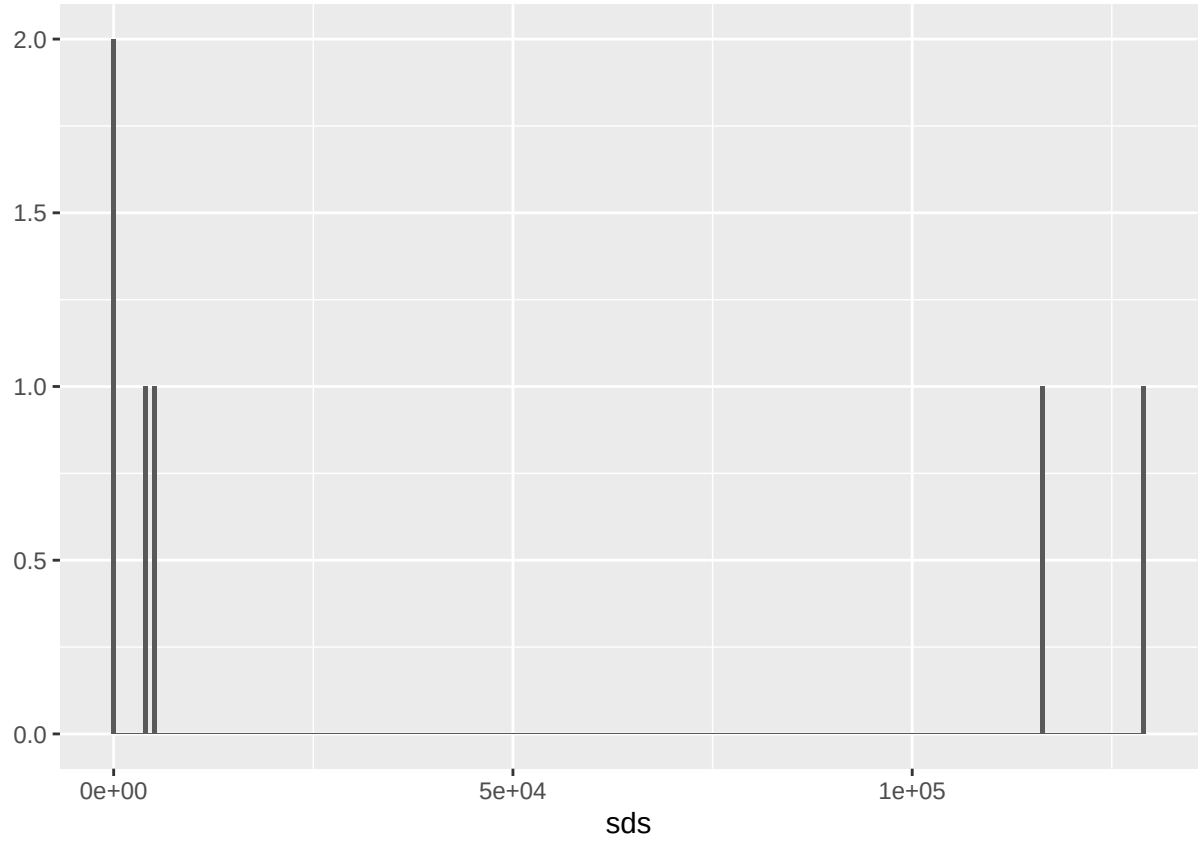
```
##
## Attaching package: 'matrixStats'
```

```
## The following object is masked from 'package:dplyr':
##
##     count
```

```
y<-as.factor(trainset$ACTIVITY)
x<-as.matrix(trainset[,2:7])
y.v<- as.factor(validset$ACTIVITY)
```

```
x.v<-validset[,2:7]

sds <- colSds(x)
qplot(sds, bins = 256) #one feature with 0 variability
```




```
nzv <- nearZeroVar(x) #none variables is recommended to be removed
col_index <- setdiff(1:ncol(x), nzv)
length(col_index) #we will keep all six variables in the model training session
```

```
## [1] 6
```

```
# image(matrix(1:6 %in% nzv))
# no need to check the image any more as it is an empty image.
```

Preprocess Visualizing the categorization

```
index_train <- createDataPartition(y, p=0.8, list = FALSE)
```

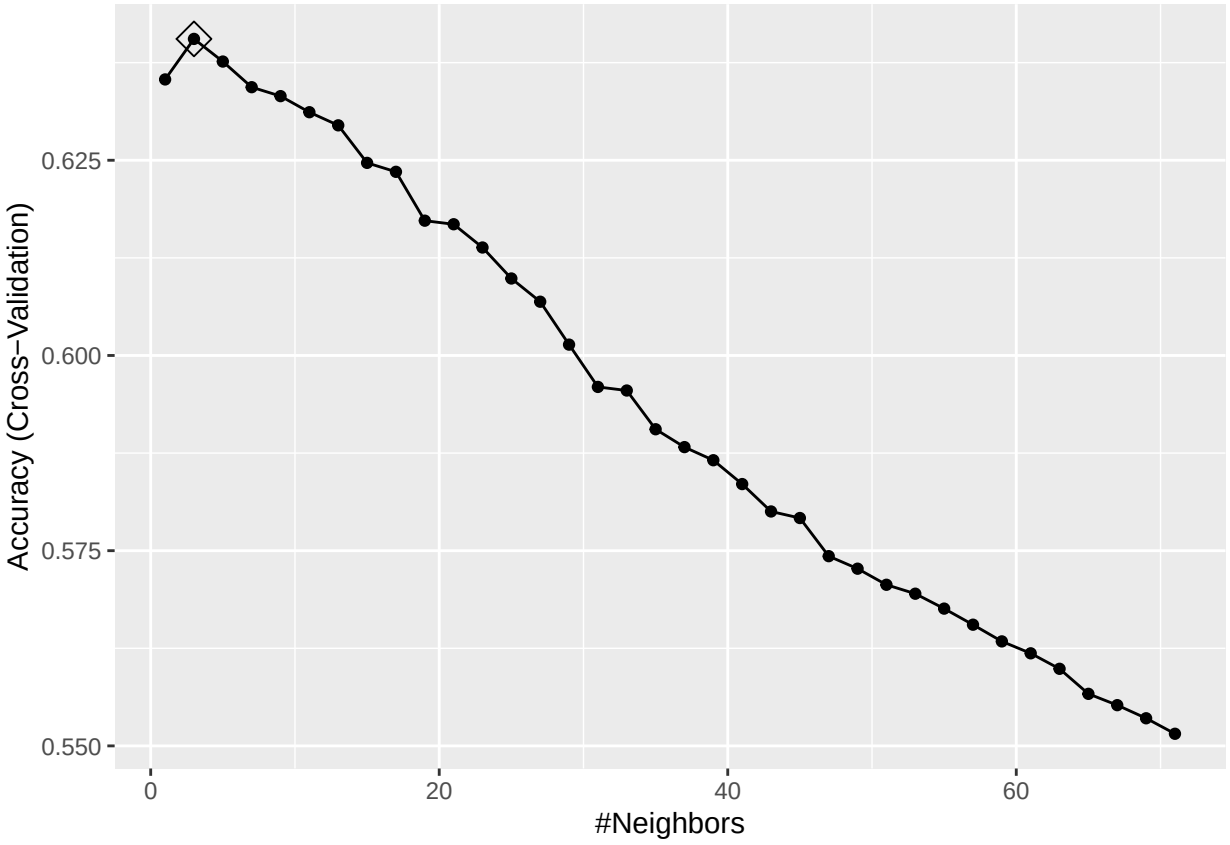
3.1 KNN

3.1.1 Raw KNN Model The KNN model produced an accuracy of .637 on the validation data set.

```
set.seed(2020)

train_knn <- train(ACTIVITY ~ ., method = "knn", data = trainset,
                   tuneGrid = data.frame(k = seq(1, 71, 2)), #Tuning the k value
                     trControl = trainControl(method="cv", number = 5))
```

```
ggplot(train_knn, highlight =  TRUE) #display where the max Overall accuracy is
```




```
y_hat_knn <- predict(train_knn, trainset)
Accuracy1 <- confusionMatrix(y_hat_knn, trainset$ACTIVITY)$overall["Accuracy"]
#test knn model on validation dataset
y_hat_knn <- predict(train_knn, validset)
Accuracy2 <- confusionMatrix(y_hat_knn, validset$ACTIVITY)$overall["Accuracy"]

tb.KNN.M1 <- data.frame(Model = "KNN", Accuracy.train = Accuracy1, Accuracy.test = Accuracy2)
tb.KNN.M1
```

```
##          Model Accuracy.train Accuracy.test
## Accuracy   KNN      0.8019538     0.6364745
```

3.1.2 KNN Model using knn3 function Using the knn3 fuction, we got an accuracy of .649 on the validation dataset.

```
set.seed(2020)
train_knn <- knn3(ACTIVITY ~ ., data = trainset)

y_hat_knn <- predict(train_knn, trainset, type = "class")
Accuracy1 <- confusionMatrix(y_hat_knn, trainset$ACTIVITY)$overall["Accuracy"]

y_hat_knn <- predict(train_knn, validset, type = "class")
Accuracy2 <- confusionMatrix(y_hat_knn, validset$ACTIVITY)$overall["Accuracy"]
```

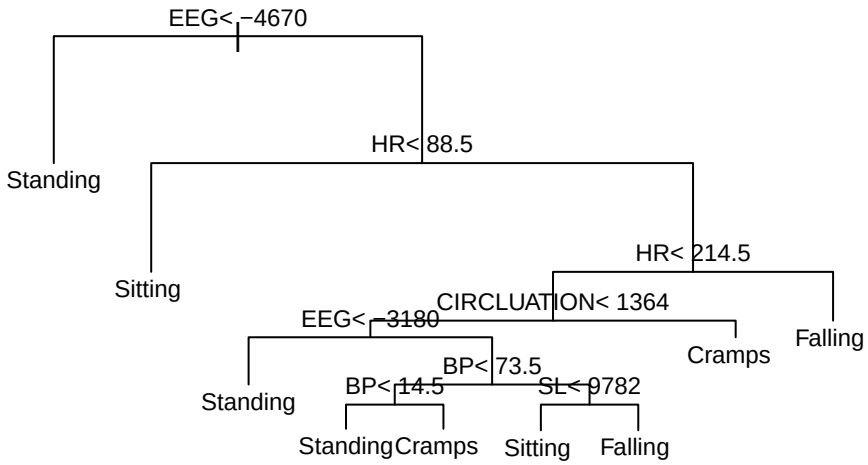
```
tb.KNN.M2 <- data.frame(Model = "KNN3", Accuracy.train = Accuracy1, Accuracy.test = Accuracy2)
tb.KNN.M2
```

```
##          Model Accuracy.train Accuracy.test
## Accuracy  KNN3      0.7588339     0.6508082
```

3.2 Regression Tree

The overall accuracy of regression tree model is .417 on validation dataset.

```
set.seed(2020)
#end up with 9 partition
library(rpart)
trainset$ACTIVITY <- as.factor(trainset$ACTIVITY)
train_rpart <- rpart(ACTIVITY ~ ., data = trainset)
plot(train_rpart, margin = 0.1)
text(train_rpart, cex = 0.75)
```




```
y_hat <- predict(train_rpart, trainset, type = "class")
Accuracy1 <- confusionMatrix(y_hat, trainset$ACTIVITY)$overall["Accuracy"]
#test RT model on validation set
y_hat <- predict(train_rpart, validset, type = "class")
Accuracy2 <- confusionMatrix(y_hat, validset$ACTIVITY)$overall["Accuracy"]

tb.RT <- data.frame(Model = "Regression Tree", Accuracy.train = Accuracy1, Accuracy.test = Accuracy2)
tb.RT
```

```
##                    Model Accuracy.train Accuracy.test
```

```
## Accuracy Regression Tree      0.4271541     0.4168954
```

3.3 Classification Tree

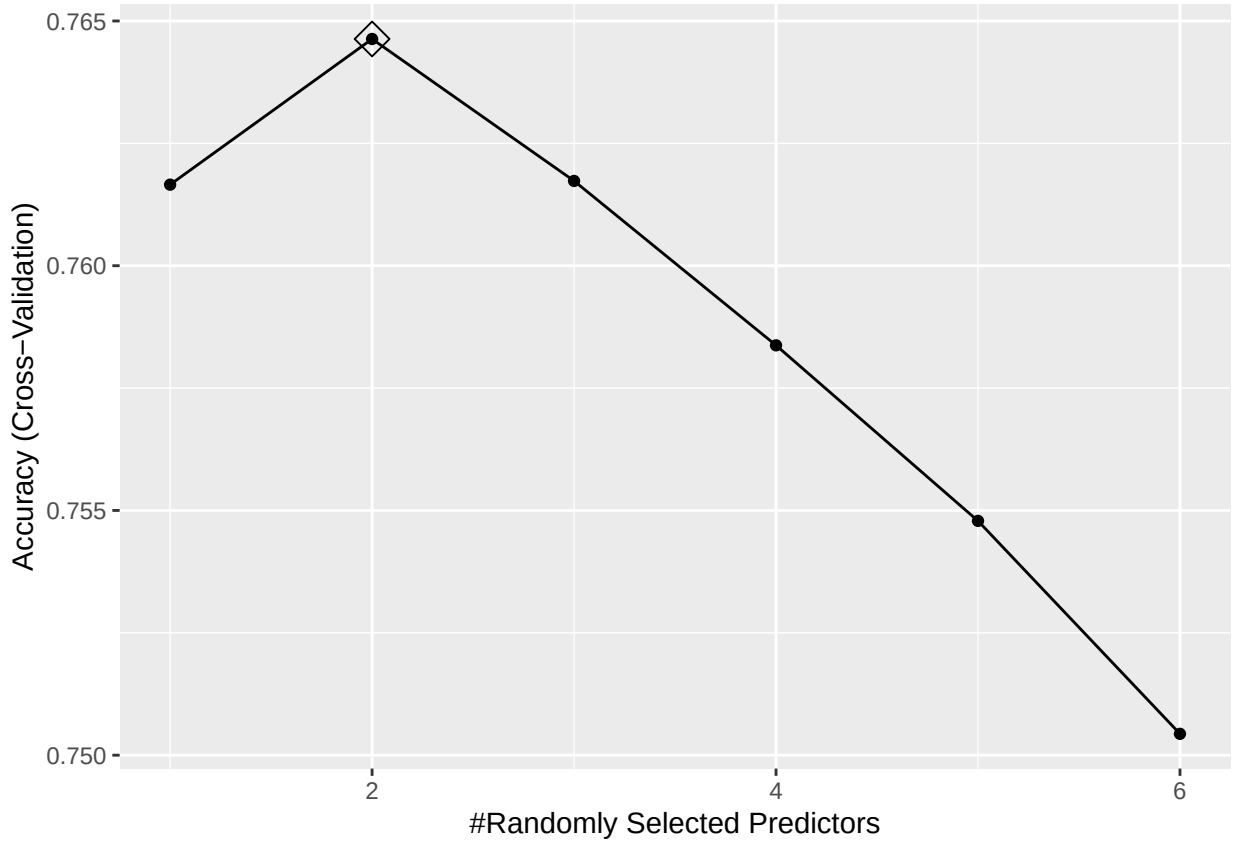
The overall accuracy of classification tree model is .676.

3.4 Rrandom Forest

```
set.seed(2020)

control <- trainControl(method="cv", number = 5) #5 fold validation
grid <- data.frame(mtry = seq(1,6,1)) #try different value of mtry
train_rf <-  train(x, y,
                   method = "rf",
                   ntree = 350,
                   trControl = control,
                   tuneGrid = grid,
                   nSamp = "best")

Pre<-predict(train_rf, x)
rfresults<-confusionMatrix(Pre,y) #Accuracy of Falling = .8162
Accuracy1 <- confusionMatrix(Pre,y)$overall["Accuracy"] #.9950393
#confusionMatrix(Pre,y)$byClass[7] #F1 = .9834535
ggplot(train_rf, highlight=TRUE)
```




```
train_rf$bestTune #mtry=2
```

```
##   mtry
## 2    2
```

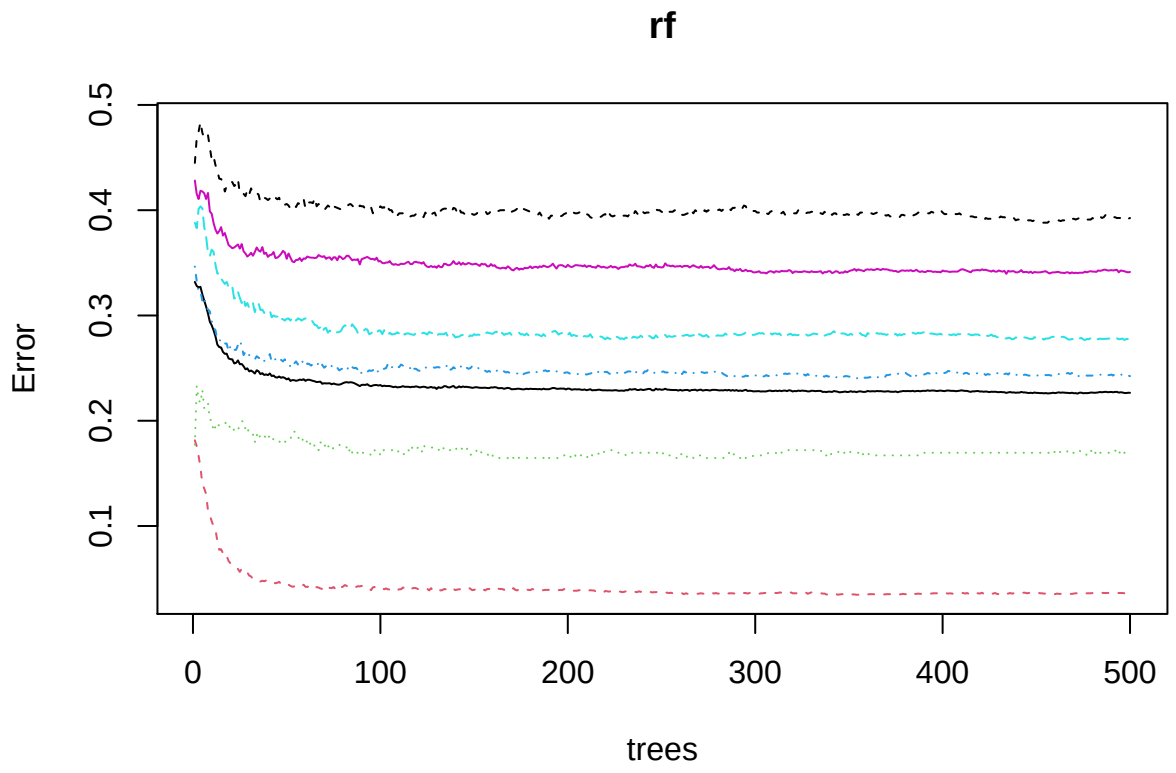
Results

Step 4 Model Test

The final random forest model made a fairly accurate prediction (.772) with a substantial Kappa value of 0.713 (Landis & Koch, 1977).

```
set.seed(2020)
rf <- randomForest(x, y, mtry=2)
# the changes as we add trees.

plot(rf) # increasing the number of trees improves the accuracy (decrease the error rate of the algorit
```




```
pre<- predict(rf, x.v)
confusionMatrix(pre,y.v)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Standing Walking Sitting Falling Cramps Running
##   Standing      888       0       5       5     10      19
##   Walking         0      77      16       3      3       1
##   Sitting         5      22     370      65     18       5
##   Falling        13       1      88     520    120      25
```

```
##   Cramps         11       1      18     117    477      82
##   Running         5       0       4       8     71     206
##
## Overall Statistics
##
##                Accuracy : 0.774
##                  95% CI : (0.7593, 0.7882)
##     No Information Rate : 0.2812
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7142
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Standing Class: Walking Class: Sitting
## Sensitivity                   0.9631        0.76238         0.7385
## Specificity                   0.9835        0.99276         0.9586
## Pos Pred Value                0.9579        0.77000         0.7629
## Neg Pred Value                0.9855        0.99245         0.9531
## Prevalence                    0.2812        0.03080         0.1528
## Detection Rate                0.2708        0.02348         0.1128
## Detection Prevalence          0.2827        0.03050         0.1479
## Balanced Accuracy             0.9733        0.87757         0.8486
##                      Class: Falling Class: Cramps Class: Running
## Sensitivity                  0.7242        0.6824        0.60947
## Specificity                  0.9036        0.9112        0.97008
## Pos Pred Value               0.6780        0.6756        0.70068
## Neg Pred Value               0.9212        0.9137        0.95578
## Prevalence                   0.2190        0.2132        0.10308
## Detection Rate               0.1586        0.1455        0.06282
## Detection Prevalence         0.2339        0.2153        0.08966
## Balanced Accuracy            0.8139        0.7968        0.78977
```

```
Accuracy2 <- confusionMatrix(pre,y.v)$overall["Accuracy"] #Accuracy = .7715767
#confusionMatrix(pre,y.v)$byClass[7] #F1 = .9813322
tb.RF <- data.frame(Model = "Random Forest", Accuracy.train = Accuracy1, Accuracy.test = Accuracy2)
tb.RF
```

```
##                   Model Accuracy.train Accuracy.test
## Accuracy Random Forest      0.9958025     0.7740165
```

Table of the accuracy of models: Although cross-validation was applied in the model training process, overtraining is still a significant problem as the table shows that all accuracy values of the training models are higher than that of test models.

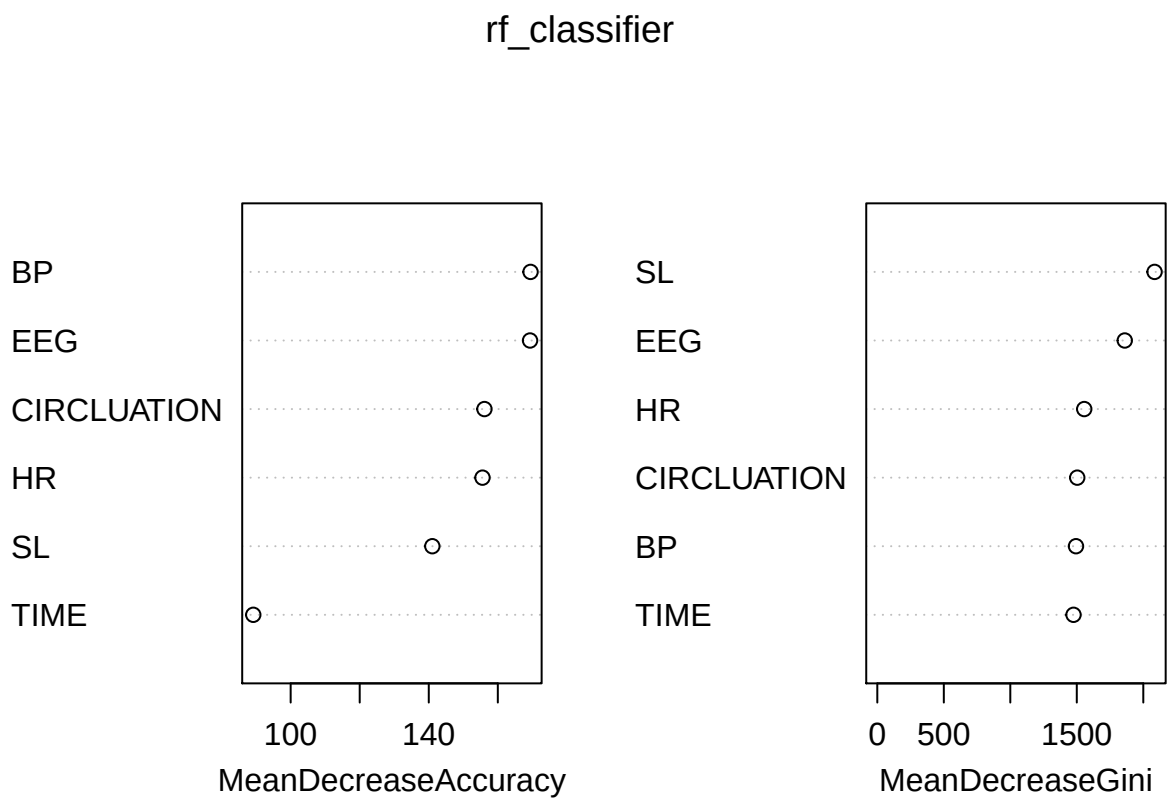
```
rbind(tb.RT, tb.CT, tb.KNN.M1, tb.KNN.M2, tb.RF)
```

```
##                          Model Accuracy.train Accuracy.test
## Accuracy       Regression Tree      0.4271541     0.4168954
## Accuracy1 Classification Tree      0.7985957     0.6758158
## Accuracy2                  KNN      0.8019538     0.6364745
## Accuracy3                KNN3      0.7588339     0.6508082
## Accuracy4        Random Forest      0.9958025     0.7740165
```

The ROC plot shows that the curves are closer to the top-left corner indicating the trade-off between

sensitivity(true positive rate) and specificity (true negative rate) is good.

```
set.seed(2020)
#ROC plot FOR RANDOM FOREST
if(!require(ROCR)) install.packages("ROCR", repos = "http://cran.us.r-project.org")
# Perform training:
rf_classifier = randomForest(ACTIVITY ~ ., data=trainset,
                             mtry=2, importance=TRUE)
varImpPlot(rf_classifier)
```




```
rf_classifier
```

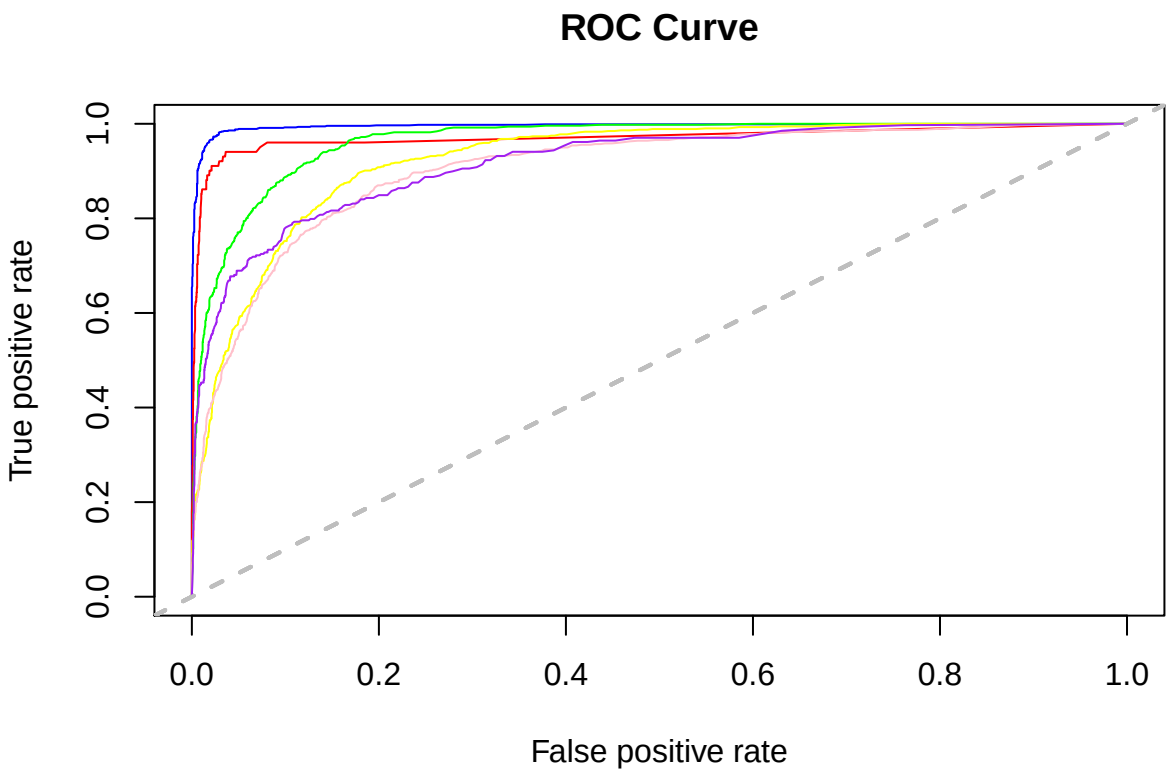
```
##
## Call:
##  randomForest(formula = ACTIVITY ~ ., data = trainset, mtry = 2,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 22.79%
## Confusion matrix:
##          Standing Walking Sitting Falling Cramps Running class.error
## Standing     3550       3      15      45     55      18  0.03689636
## Walking         1     332      64       3      0       1  0.17206983
## Sitting        20      60    1504     355     51      11  0.24837581
## Falling        24       5     327    2073    418      23  0.27770035
## Cramps         51       4      99     558   1838     245  0.34239714
```

```
## Running        47       3      30      92     358     820  0.39259259
```

```
# Validation set assessment #1: looking at confusion matrix
prediction_for_table <- predict(rf_classifier, validset[,-1])
table(validset$ACTIVITY,predicted=prediction_for_table)
```

```
##           predicted
##            Standing Walking Sitting Falling Cramps Running
##   Standing      885       0       5      15     14       3
##   Walking         0      75      24       1      1       0
##   Sitting         5      15     372      89     16       4
##   Falling         8       3      65     522    113       7
##   Cramps         10       3      17     119    477      73
##   Running        16       1       5      25     86     205
```

```
# Calculate the probability of new observations belonging to each class
# prediction_for_roc_curve will be a matrix with dimensions data_set_size x number_of_classes
prediction_for_roc_curve <- predict(rf_classifier,validset[,-1],type="prob")
# Use pretty colours:
pretty_colours <- c("blue","red","green", "yellow","pink", "purple")
# Specify the different classes
classes <- levels(validset[,1])
# For each class
for (i in 1:6){
  # Define which observations belong to class[i]
  true_values <- ifelse(validset[,1]==classes[i],1,0)
  # Assess the performance of classifier for class[i]
  pred <- prediction(prediction_for_roc_curve[,i],true_values)
  perf <- performance(pred, "tpr", "fpr")
  if (i==1)
  {
    plot(perf,main="ROC Curve",col=pretty_colours[i])
  }
  else
  {
    plot(perf,main="ROC Curve",col=pretty_colours[i],add=TRUE)
  }
  # Calculate the AUC and print it to screen (FALLING .9597418)
  auc.perf <- performance(pred, measure = "auc")
  print(auc.perf@y.values)
  abline(a=0,b=1,lwd=2,lty=2,col="gray")
}
```




```
## [[1]]
## [1] 0.9954016
##
## [[1]]
## [1] 0.9707519
##
## [[1]]
## [1] 0.9638495
##
## [[1]]
## [1] 0.9252368
##
## [[1]]
## [1] 0.9054929
##
## [[1]]
## [1] 0.9182015
```

Conclusion

This project aims to apply different machine learning models to predict activity type (especially "falling") using six biometric indicators. I applied decision trees, knn, random forest models in this project. The results showed that the random forest gave the best prediction with an overall accuracy of .77, which means the model correctly predicted 77% of the validation data. The accuracy, sensitivity, and specificity of predicting "falling" is .8138, .726, and .902, respectively. More advanced machine learning algorithms, such as Gradient boosted machines (GBMs), can be applied to predicting the activity type better and to avoiding overtraining.

Reference

Landis, J.R.; Koch, G.G. (1977). "The measurement of observer agreement for categorical data". *Biometrics*, *33* (1): 159–174

Ozdemir, A. T., & Barshan, B. (2014). Detecting falls with wearable sensors using machine learning techniques. *Sensors*, *14*(6), 10691-10708.