Data Preprocessing

Zhen Trinh

Load in required packages

```
library(plyr)
library(rcompanion)
library(forecast)
library(caret)
library(corrplot)
```

Load in data

```
# Define a function to load and preprocessing data
prep.fun <- function(path){

  # Read in dataset
  df <- read.table(path, header = F)

  # Remove transition activity (denoted as 0 in ACTIVITY column) from analysis
  return(df[!(df$V24 == 0),])
}

# Apply prep.fun to the file then combine results into a data frame
subject1 <- ldply(.data = "Data/mHealth_subject1.log", .fun = prep.fun)
```

Detect NAs

```
apply(subject1, 2, function(x) any(is.na(x)))
```

```
   V1    V2    V3    V4    V5    V6    V7    V8    V9   V10   V11   V12
FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
  V13   V14   V15   V16   V17   V18   V19   V20   V21   V22   V23   V24
FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

There is no missing values across the dataset, so we need not to worry about imputing missing data.

Normalize features

Since the scale used for the values of each variable is different, we are going to transform all the values to a common scale using min-max transformation so that all data has a minimum of 0 and maximum of 1.

```
# Define the min-max normalization function
norm.fun <- function(x){
  z <- (x-min(x))/(max(x)-min(x))
  return(z)
}
```

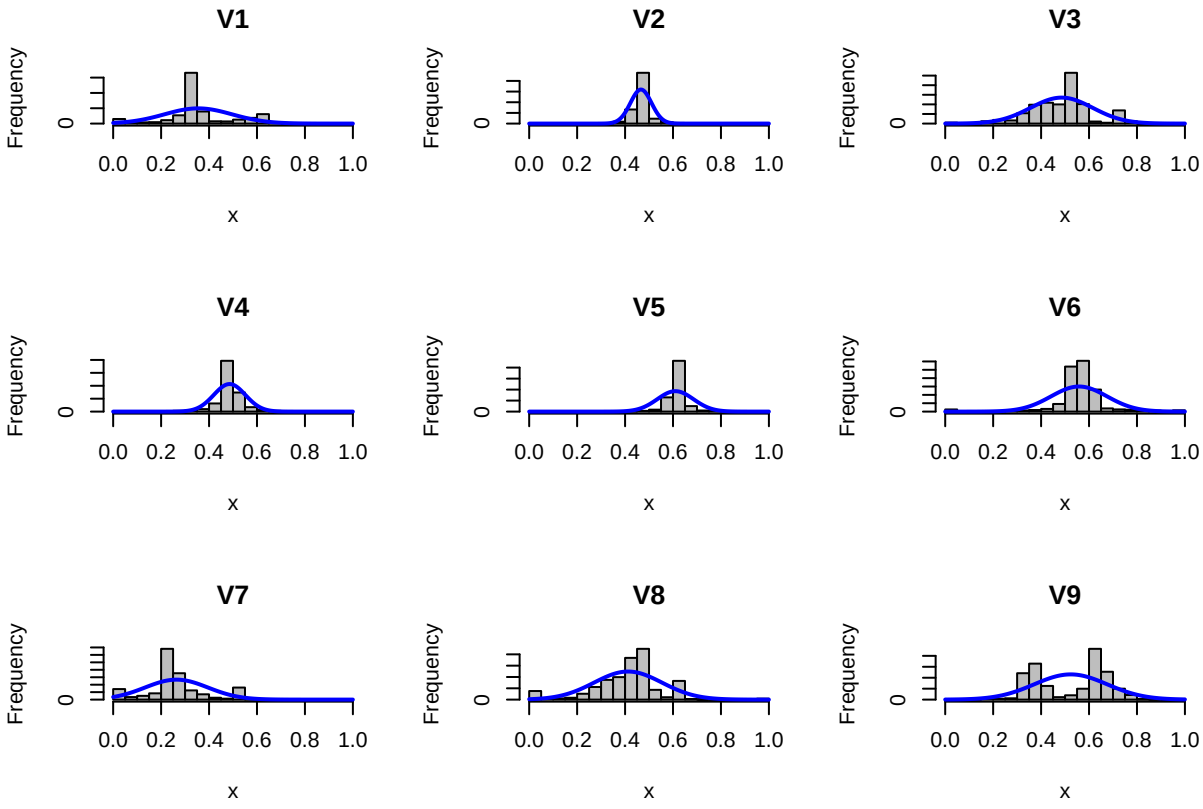
```r
# Apply the normalization function to all numeric features
subject1[1:23] <- as.data.frame(lapply(subject1[1:23], norm.fun))
```

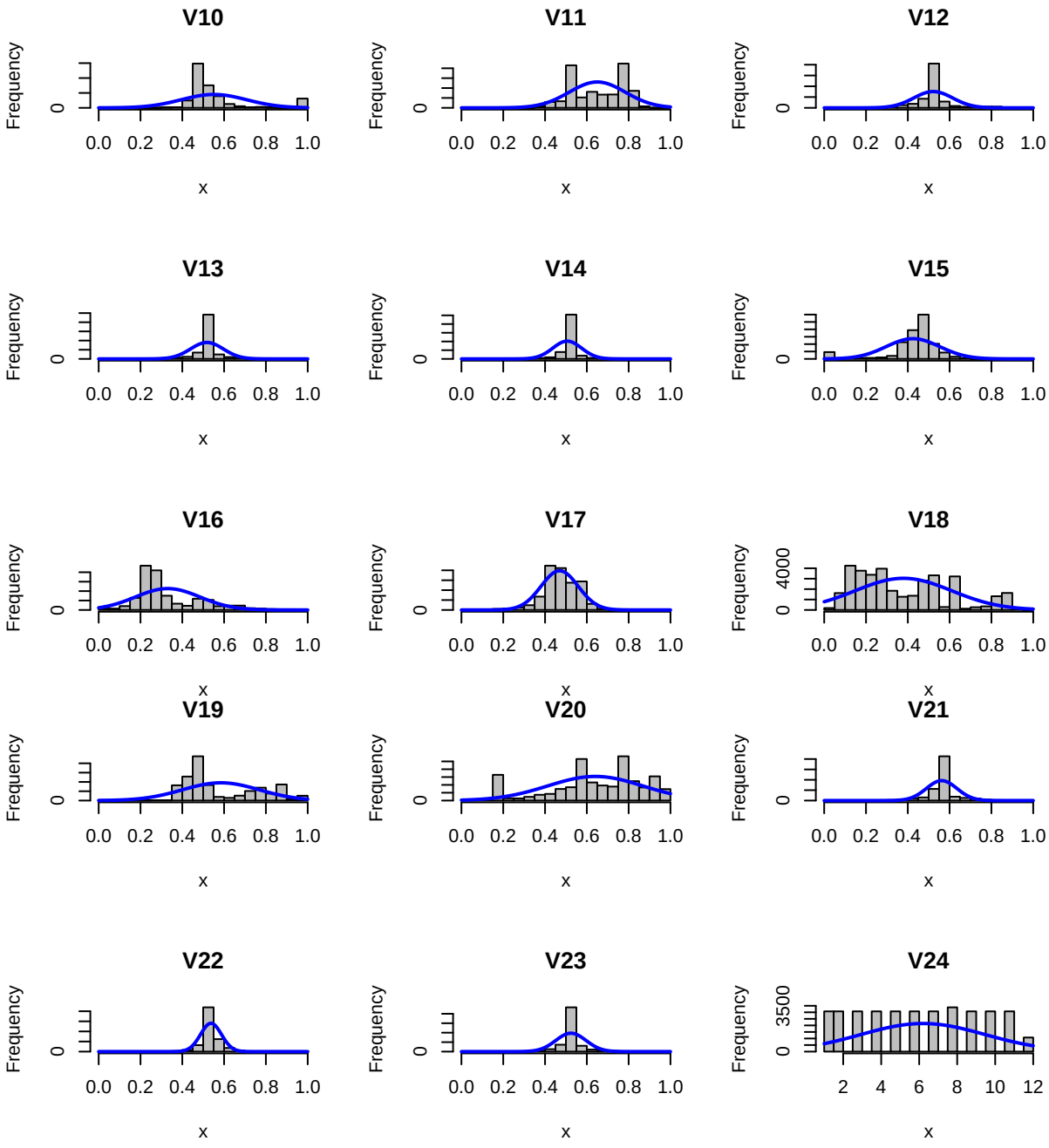
Check for skewness

```r
# Define a function to plot histogram of each feature
norm.plot.fun <- function (index) {
  plotNormalHistogram(subject1[,index], main = names(subject1)[index])
}

# Fit all the graphs into one page
par(mfrow=c(3,3))

# Apply the plot function to all columns of the data
lapply(1:24, FUN = norm.plot.fun)
```

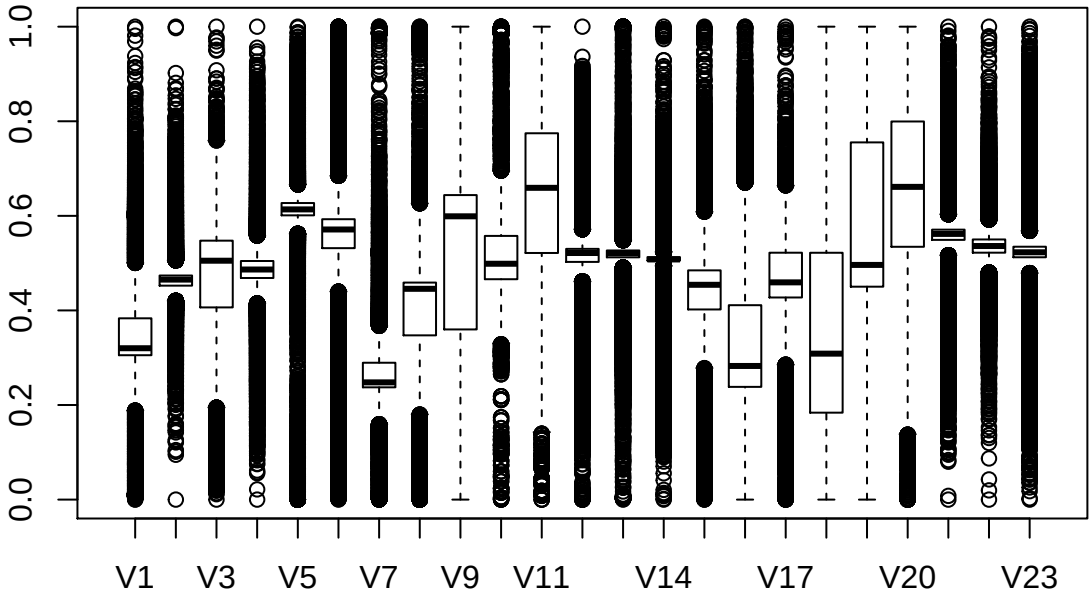




The data seems to be fairly normally distributed, thus we can apply parametric tests on the data. Although there are some skewnesses in some of the features, because they are so small that we need not to transform them. Plus, since we are going to use all the variables in the same analysis, we have to apply the same transformation to all of them. It makes no sense to transform already normally distrubuted data, which is a mojority of them.

Detect outliers

```
# Use boxplots to visualuze outliers in every feature
boxplot(subject1[1:23])
```



Except for V18 and V20, all other variables have a significant amount of outliers. While V9,V11, and V19 only have outliers on the lower bound, V16 only has outliers on the upper bound.

We are going to treat the outliers by applying the tsclean() function, which fits a robust trend using loess (for non-seasonal series) in the forecast package. The residuals are computed and the upper bound and lower bounds are computed as: $U = q_{0.9} + 2(q_{0.9} - q_{0.1})$; $L = q_{0.9} - 2(q_{0.9} - q_{0.1})$ **where** $q_{0.1}$ **and** $q_{0.9}$ **are the 10th and 90th percentiles of the residuals respectively.**

```
# Apply tsclean() function to all the features
subject1[1:23] <- as.data.frame(lapply(subject1[1:23], tsclean))
```

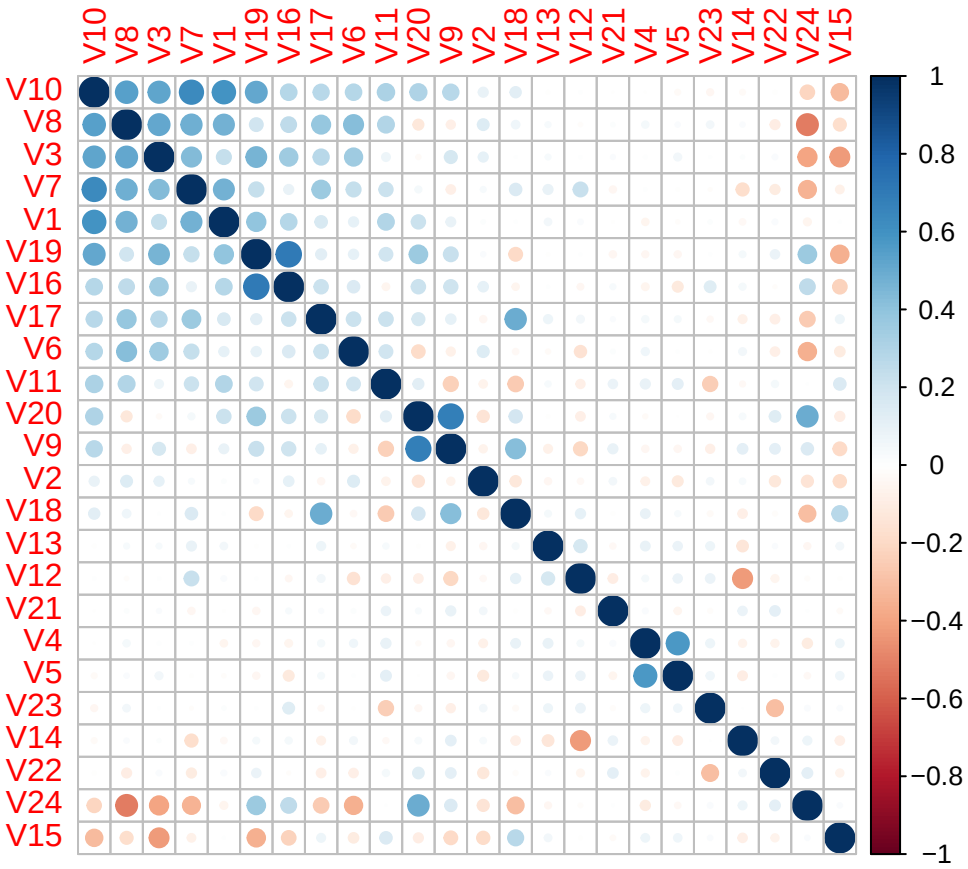
Check correlation

```
# Get a correlation matrix
cor.matrix <- cor(subject1)

# Find attributes that are highly correlated with a cutoff value of 0.75
highlyCorrelated <- findCorrelation(cor.matrix, cutoff = 0.75)
highlyCorrelated
```

```
integer(0)
```

```
# Plot the correlation and order the matrix according to first component order
corrplot(cor.matrix,order = "FPC")
```



We can be sure that none of the attribute is highly correlated with another, i.e. there is no redundancy in the data. V24 is the response variable we are interested in. It seems that compared to all other variables, V20 and V19, which are readings from gyroscope, have more correlation with the response variable.

Feature selection with PCA

Principal component analysis (PCA) is a descriptive method to analyze dependencies (correlations) between variables. We will use PCA to determine which features should be included in the machine learning models.

```
# Perform PCA on training set, set scale. = T so that standard deviation = 1
pca <- prcomp(subject1, scale. = T)

# Print out summary of the PCA fit
summary(pca)
```

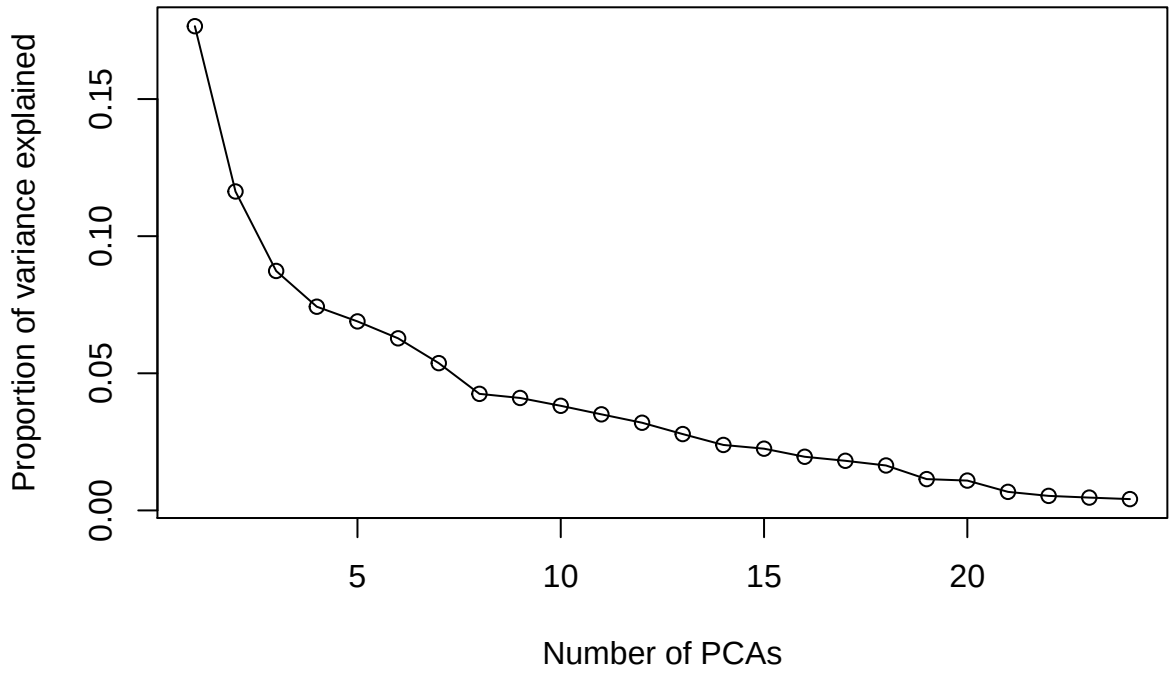
```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6
Standard deviation     2.0585 1.6707 1.44767 1.33521 1.28609 1.22716
Proportion of Variance 0.1766 0.1163 0.08732 0.07428 0.06892 0.06275
Cumulative Proportion  0.1766 0.2929 0.38018 0.45447 0.52338 0.58613
                          PC7     PC8     PC9    PC10    PC11    PC12
Standard deviation     1.13541 1.01001 0.99213 0.95692 0.91702 0.87600
Proportion of Variance 0.05371 0.04251 0.04101 0.03815 0.03504 0.03197
```

```
Cumulative Proportion  0.63985 0.68235 0.72336 0.76152 0.79656 0.82853
                          PC13    PC14    PC15    PC16   PC17    PC18
Standard deviation     0.81747 0.75699 0.73508 0.68533 0.6591 0.62723
Proportion of Variance 0.02784 0.02388 0.02251 0.01957 0.0181 0.01639
Cumulative Proportion  0.85637 0.88025 0.90277 0.92233 0.9404 0.95683
                          PC19    PC20    PC21   PC22    PC23    PC24
Standard deviation     0.52361 0.51107 0.40301 0.3566 0.33501 0.31471
Proportion of Variance 0.01142 0.01088 0.00677 0.0053 0.00468 0.00413
Cumulative Proportion  0.96825 0.97913 0.98590 0.9912 0.99587 1.00000
```

We see that the first 21 components explained 97% of total variation, and the first 23 components explained 99% of total variation. The last component can be ignored without missing much information. We can use a scree plot to aid deciding how many components we want to include in the ML models.

```r
# Save the proportion of variance of each PC (2nd row)
p.var <- summary(pca)$importance[2,]

# Plot PCAs and the proportion of variance explained
plot(1:length(p.var), p.var, xlab = "Number of PCAs",
     ylab = "Proportion of variance explained", type = "o")
```



It appears that the first component explains most of the variance, and there is a sharp drop afterwards. The curve has a significant bend at n=8, so we will only use the first 8th principal components for the analysis.

```r
# Keep only the first 10 principal components for analysis
data.pca <- pca$x[,1:8]

# Response variable with true activity
```

```r
Activity <- subject1$V24

# Combine response variable with PCA data
subjects.new <- data.frame(data.pca, Activity)

# Rename the activity column so it's easier to identify characters from numbers
subjects.new$Activity <- paste("L", subjects.new$Activity, sep="")

# Change the class of the activity column to factor
subjects.new$Activity <- as.factor(subjects.new$Activity)
```

Split data

```r
# set the seed to make the partition reproductible
set.seed(123)

# Sample into 3 sets so that the ratio of train:test:validation = 6:2:2
idx <- sample(seq(1,3), size = nrow(subjects.new), replace = T,
              prob = c(.6, .2, .2))

# Divide data into training, testing, and validation sets
train <- subjects.new[idx == 1, ]
test <- subjects.new[idx ==2, ]
validation <- subjects.new[idx ==3, ]
```

Export data

```r
write.csv(subjects.new, row.names=FALSE, file = "Data/subjects.new.csv")
write.csv(train, row.names=FALSE, file = "Data/train.csv")
write.csv(test, row.names=FALSE, file = "Data/test.csv")
write.csv(validation, row.names=FALSE, file = "Data/validation.csv")
```