main

Huong Tran.

Introduction:

In this assignment, we are going to use data from accelerometers on the belt, forearm, arm and dumbell to quantify how well people do a particular activity. There are 5 classes of activities: sitting-down, standing up, standing, walking and sitting.

Load the packages:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(caret)
```

```
## Loading required package: lattice
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

Data Exploration:

Now, we are going to load the data provide on websites:

```
training.url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testing.url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# create a directory for data:
dir.create("data")
```

```
## Warning in dir.create("data"): 'data' already exists
```

```
download.file(training.url, destfile = "data/training.csv")
download.file(testing.url, destfile = "data/testing.csv")

training.orig <- read.csv("data/training.csv")
testing.orig <- read.csv("data/testing.csv")
```

```r
dim(training.orig)
```

```
## [1] 19622   160
```

```r
dim(testing.orig)
```

```
## [1]  20 160
```

There are 160 variables. While training set contains more than 19,000 observations, the testing set only have 20 observations. However, the first column is index, which is not necessary for our prediction, we are able to drop them.

```r
delete_columns <- grepl("X|user_name|window|timestamp|max|min|var|avg|stddev|skewness|kurtosis|amplitud
training <- training.orig[, !delete_columns]
names(training)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

All varialbes are numeric, while the last variables is our outcome "classe" is categorical variables.

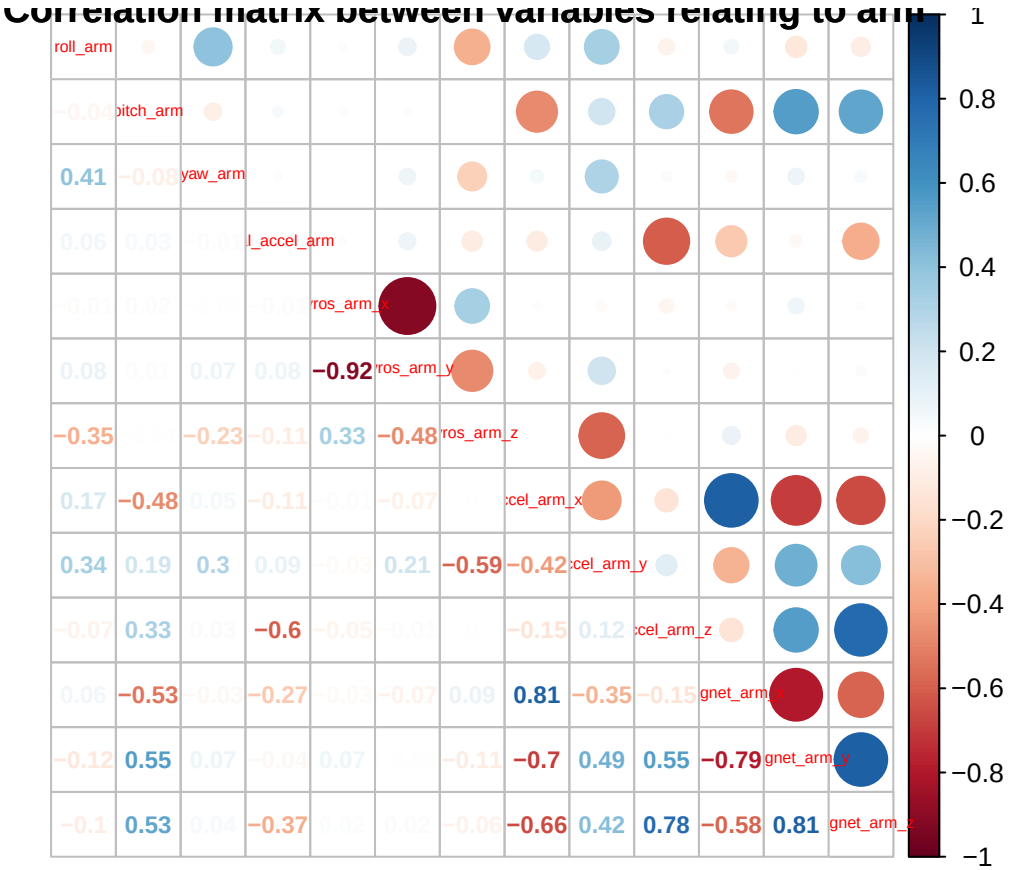
Arms - relating variables:

```r
names(training)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
```

```
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

```
arm.names <- grepl("_arm", names(training))
arm <- subset(training, select = arm.names)
arm.cor <- cor(arm)
corrplot::corrplot.mixed(arm.cor,
                         number.cex = 0.75,
                         tl.cex = 0.5,
                         main = "Correlation matrix between variables relating to arm")
```



As, we can see, the infomation recorded at arm are correlated, but not too high.

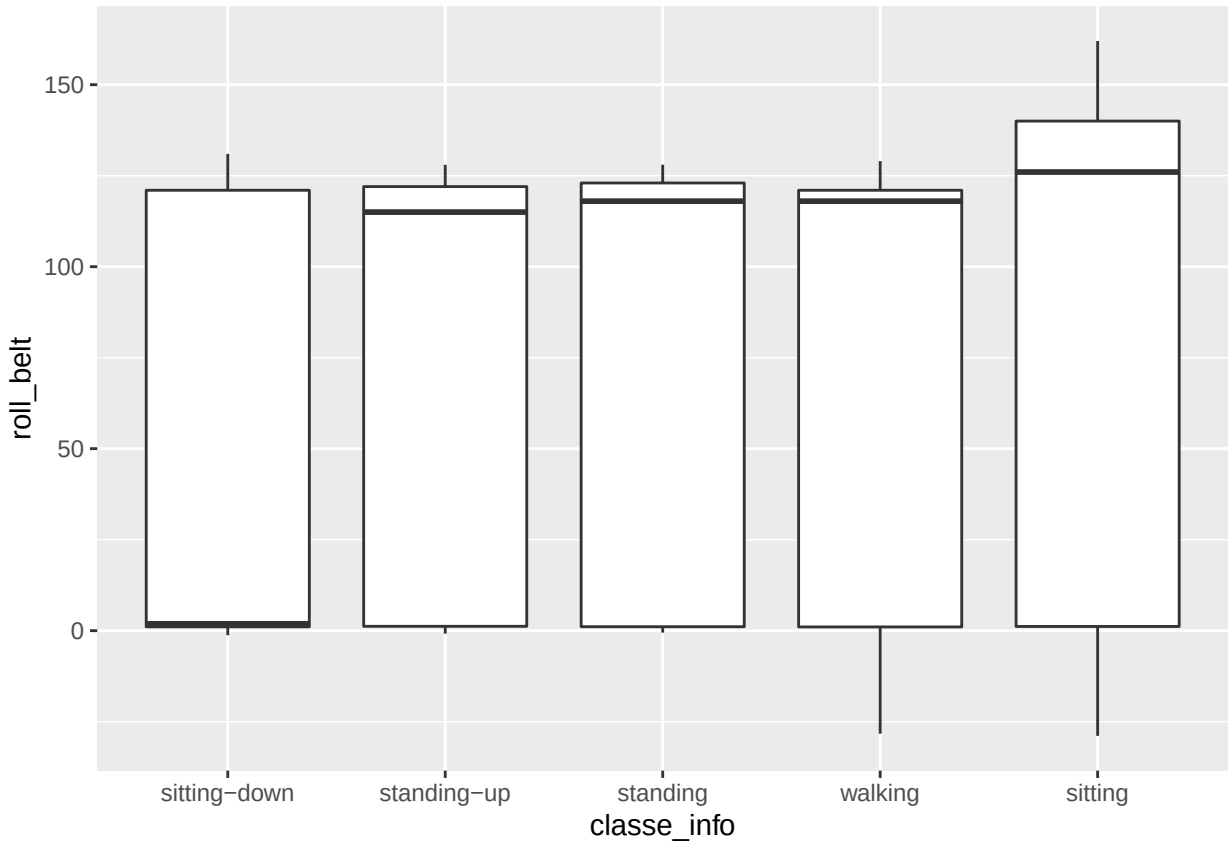
Relation to activities:

Now, we will use informative labels for the "classe":

```
training$classe <- as.factor(training$classe)

training$classe_info <- factor(training$classe, labels = c("sitting-down",
                                                           "standing-up",
                                                           "standing",
                                                           "walking",
                                                           "sitting"))
```

```
g <- ggplot(training, aes(classe_info, roll_belt))
g + geom_boxplot()
```



People who is sitting down yeild the value of "roll_belt" lower than other type of activities.

Data preprocess:

Now we will split training set into traning and validation set:

```
training <- training[, -ncol(training)]

set.seed(123)
inTrain <- createDataPartition(y=training$classe,p=0.8,list=FALSE)
validation <- training[-inTrain,]
training <- training[inTrain,]
```

Although we reduced elminate the unnecessary predictors, we still have 54 variables, which is very large. Now we will use PCA (principle component analysis) to reduce the number of predictors as well as highly correlated observations.

```
prProc <- preProcess(training[,-ncol(training)], method = "pca",thresh = 0.8)
trainPC <- predict(prProc, training)

prProc$numComp
```

```
## [1] 12
```

We reduce 53 variables into 13 variables.

Fit model with Random Forest:

Random Forest perform pretty well in classification model, now we try to use Random Forest to predict the activities. Will both model with and without PCA to have a better comparsion.

Model without PCA:

```
# We will use k-fold cross validation to resample data
start.time <- Sys.time()
mod.rf <- train(classe ~., method = "rf", data = training,
                trControl = trainControl(method = "cv"), number = 3)
end.time <- Sys.time()
time.rf <- end.time - start.time
cat("Training time is: ", time.rf)
```

```
## Training time is:  22.55036
```

Now, we are going to fit the model in validation set:

```
pred.rf <- predict(mod.rf, validation)
con.Matrix.rf <- confusionMatrix(pred.rf, validation$classe)

rf.acc <- append(con.Matrix.rf$overall, time.rf)
rf.acc <- data.frame(rf.acc)
row.names(rf.acc)[8] <- "time"
names(rf.acc) <- "Random forest"
```

Model without PCA

```
start.time <- Sys.time()
mod.rf.pca <- train(classe ~., method = "rf", data = trainPC,
                trControl = trainControl(method = "cv"), number = 3)
end.time <- Sys.time()
time.pca <- end.time - start.time
cat("Training time is: ", time.pca)
```

```
## Training time is:  5.842776
```

Fit model in validation set:

```
validationPC <- predict(prProc, validation)
pred.pca <- predict(mod.rf.pca, validationPC)
con.Matrix.pca <-confusionMatrix(pred.pca, validation$classe)

pca.acc <- append(con.Matrix.pca$overall, time.pca)
pca.acc <- data.frame(pca.acc)
row.names(pca.acc)[8] <- "time"
names(pca.acc) <- "Random forest with PCA"
```

Make comparsion between the two models:

```
cbind.data.frame(rf.acc, pca.acc)
```

```
##                Random forest Random forest with PCA
## Accuracy           0.9956666            0.966607188
```

```
## Kappa             0.9945181           0.957757507
## AccuracyLower     0.9930708           0.960498964
## AccuracyUpper     0.9974737           0.972006733
## AccuracyNull      0.2844762           0.284476166
## AccuracyPValue    0.0000000           0.000000000
## McnemarPValue           NaN           0.001707649
## time             22.5503569           5.842775599
```

Comment: We use random forest with and withoug Principle Components. While using PCA yeilds lower accuracy on the validation set, but total for trainin model is much more shorter. Howerver, the tradeoff is not too expensive, I recommend to use PCA before training random forest model to reduce the price.

Predict on testing set:

```
validationPC <- predict(prProc, validation)
testingPC <- predict(prProc, testing.orig)
test.rf <- predict(mod.rf, testing.orig)
test.rf <- as.character(test.rf)
test.rf.pca <- as.character(predict(mod.rf.pca, testingPC))
data.frame(rbind(test.rf, test.rf.pca))
```

```
##             X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14 X15 X16 X17 X18 X19
## test.rf      B  A  B  A  A  E  D  B  A   A   B   C   B   A   E   E   A   B   B
## test.rf.pca  B  A  A  A  A  E  D  B  A   A   A   C   B   A   E   E   A   B   B
##             X20
## test.rf       B
## test.rf.pca   B
```

They two model work well, but they are sensitive of classifying class A and B, which is sitting-down and standing-up.