TCGA-BRCA

```r
require(tidyverse)
require(limma)
require(TCGAbiolinks)
require(genefilter)
require(reshape2)
require(magrittr)
require(caret)
require(e1071)
require(randomForest)
require(foreach)
require(import)
require(doParallel)
require(caTools)
require(pROC)
require(RColorBrewer)

################### Download expression table and write it to disc #############
# exp_brca_hiseq <- getLinkedOmicsData(
#   project = "TCGA-BRCA",
#   dataset = "RNAseq (HiSeq, Gene level)")
# write.csv(exp_brca_hiseq, "exp_brca_hiseq.csv",row.names = T, quote = F, sep = ",")

################### Download subtype info and write it into disc ################
# pheno <- TCGAquery_subtype("BRCA")
# write.csv(pheno, "pheno.csv", quote = F, sep = ",")

# Reading expression table and subtype information
exp <- read_csv("exp_brca_hiseq.csv")
colnames(exp)[1] <- "ID"
genes <- exp[['ID']]
exp[['ID']] <- NULL

pheno <- read_csv("pheno.csv")
pheno$patient <- gsub('-', '\\.', pheno$patient)
pheno <- pheno[,c(1,12)]
colnames(pheno) <- c('sample', 'subtype')
# Tow missing valuse in pheno data ---> removing them
length(which(is.na(pheno) == TRUE))
```

```
## [1] 2
```

```r
pheno <- na.omit(pheno)

# Making both files ready
s_samples <- which(!names(exp) %in% pheno$sample)
s_samples1 <- which(!pheno$sample %in% names(exp))
exp <- exp[, -s_samples]
```

```r
pheno <- pheno[-s_samples1,]
head(pheno, 10)
```

```
## # A tibble: 10 x 2
##    sample       subtype
##    <chr>        <chr>
##  1 TCGA.3C.AAAU LumA
##  2 TCGA.3C.AALI Her2
##  3 TCGA.3C.AALJ LumB
##  4 TCGA.3C.AALK LumA
##  5 TCGA.4H.AAAK LumA
##  6 TCGA.5L.AAT0 LumA
##  7 TCGA.5L.AAT1 LumA
##  8 TCGA.5T.A9QA LumB
##  9 TCGA.A1.A0SB Normal
## 10 TCGA.A1.A0SD LumA
```

```r
which(!pheno$sample %in% names(exp))
```

```
## integer(0)
```

```r
row_order_exp <- order(pheno$sample)
exp <- exp[, row_order_exp]
identical(pheno$sample, colnames(exp))
```

```
## [1] TRUE
```

```r
exp <- as.matrix(exp)
rownames(exp) <- genes
# Dimension of expression set
dim(exp)
```

```
## [1] 20155  1081
```

```r
# According to resource of expression set, it is normalized by RPKM.
# RNAseq data normalized counts (Illumina HiSeq platform, Gene-level, RPKM)
# Skipping the normalization step
# It is also Log2(Val+1)) transformed.
# source : http://linkedomics.org/data_download/TCGA-BRCA/

# Filtering low counts reads expression set
exp <- varFilter(exp)
# Dimension of expression set after filtering
dim(exp)
```

```
## [1] 10077  1081
```

```r
# Checking for missing valuse
length(which(is.na(exp) == TRUE))
```

```
## [1] 0
```

```r
# No missing values found
# However, bellow are the code for imputing missing values using KNN
# require(DMwR)
# knnOutput <- knnImputation(exp[,-NCOL(exp)])
# anyNA(knnOutput)
# Differential gene expression analysis
groups <- factor(pheno$subtype)
design <- model.matrix(~ 0 + groups)
colnames(design) <- sub("groups","",colnames(design))
head(design, 10)
```

```
##    Basal Her2 LumA LumB Normal
## 1      0    0    1    0      0
## 2      0    1    0    0      0
## 3      0    0    0    1      0
## 4      0    0    1    0      0
## 5      0    0    1    0      0
## 6      0    0    1    0      0
## 7      0    0    1    0      0
## 8      0    0    0    1      0
## 9      0    0    0    0      1
## 10     0    0    1    0      0
```

```
fit <- lmFit(exp, design)
contrast.matrix <- makeContrasts(Basal-Normal,
                                 Her2-Normal,
                                 LumA-Normal,
                                 LumB-Normal,
                                 levels=design)
contrast.matrix
```

```
##         Contrasts
## Levels   Basal - Normal Her2 - Normal LumA - Normal LumB - Normal
##   Basal               1             0             0             0
##   Her2                0             1             0             0
##   LumA                0             0             1             0
##   LumB                0             0             0             1
##   Normal             -1            -1            -1            -1
```

```
fit2 <- contrasts.fit(fit,contrast.matrix)
EB <- eBayes(fit2)

colnames(EB$coefficients)
```

```
## [1] "Basal - Normal" "Her2 - Normal"  "LumA - Normal"  "LumB - Normal"
```

```
Basal <- topTable(EB,1,number=Inf,adjust="fdr")
Her2 <- topTable(EB,2,number=Inf,adjust="fdr")
LumA <- topTable(EB,3,number=Inf,adjust="fdr")
LumB <- topTable(EB,4,number=Inf,adjust="fdr")

DEGs_Basal <- Basal[which(Basal$adj.P.Val < 0.05 & abs(Basal$logFC) > 1),]
DEGs_Her2 <- Her2[which(Her2$adj.P.Val < 0.05 & abs(Her2$logFC) > 1),]
DEGs_LumA <- LumA[which(LumA$adj.P.Val < 0.05 & abs(LumA$logFC) > 1),]
DEGs_LumB <- LumB[which(LumB$adj.P.Val < 0.05 & abs(LumB$logFC) > 1),]
total <- nrow(DEGs_Basal) + NROW(DEGs_Her2) + NROW(DEGs_LumA) + NROW(DEGs_LumB)
sprintf("Total number of significantly expressed genes (DEGs) is %s", total)
```

```
## [1] "Total number of significantly expressed genes (DEGs) is 10881"
```

```
# It seems that we have duplicate genes, but we will take care of it later.
# Checking up and down regulated genes for Basal subtype
Up_regulated <- DEGs_Basal[which(DEGs_Basal$logFC > 0), ]
sprintf("The number of Up-regulated genes is %s", nrow(Up_regulated))
```

```
## [1] "The number of Up-regulated genes is 731"
```

```
# Sorting Fold_change decreasing by order function
Top_10_Up_regulated <- Up_regulated[order(Up_regulated$logFC,
                                          decreasing = TRUE),c(1,5)]
# Top 10 up regulated genes
head(Top_10_Up_regulated, 10)
```

```
##             logFC    adj.P.Val
## HORMAD1  5.107461 1.070322e-49
## PRAME    4.833850 1.291215e-13
## CA9      4.447592 6.509863e-19
## ART3     4.073711 2.079570e-36
## A2ML1    3.724826 7.600709e-20
## MSLN     3.699381 2.958991e-14
## KIF1A    3.671748 8.609794e-11
## LOC84740 3.543368 7.281382e-24
## TLX1     3.460044 8.883799e-18
## ACTL8    3.372360 9.014721e-14
```

```
Down_regulated <- DEGs_Basal[which(DEGs_Basal$logFC < 0), ]
sprintf("The number of Down-regulated genes is %s", nrow(Down_regulated))
```

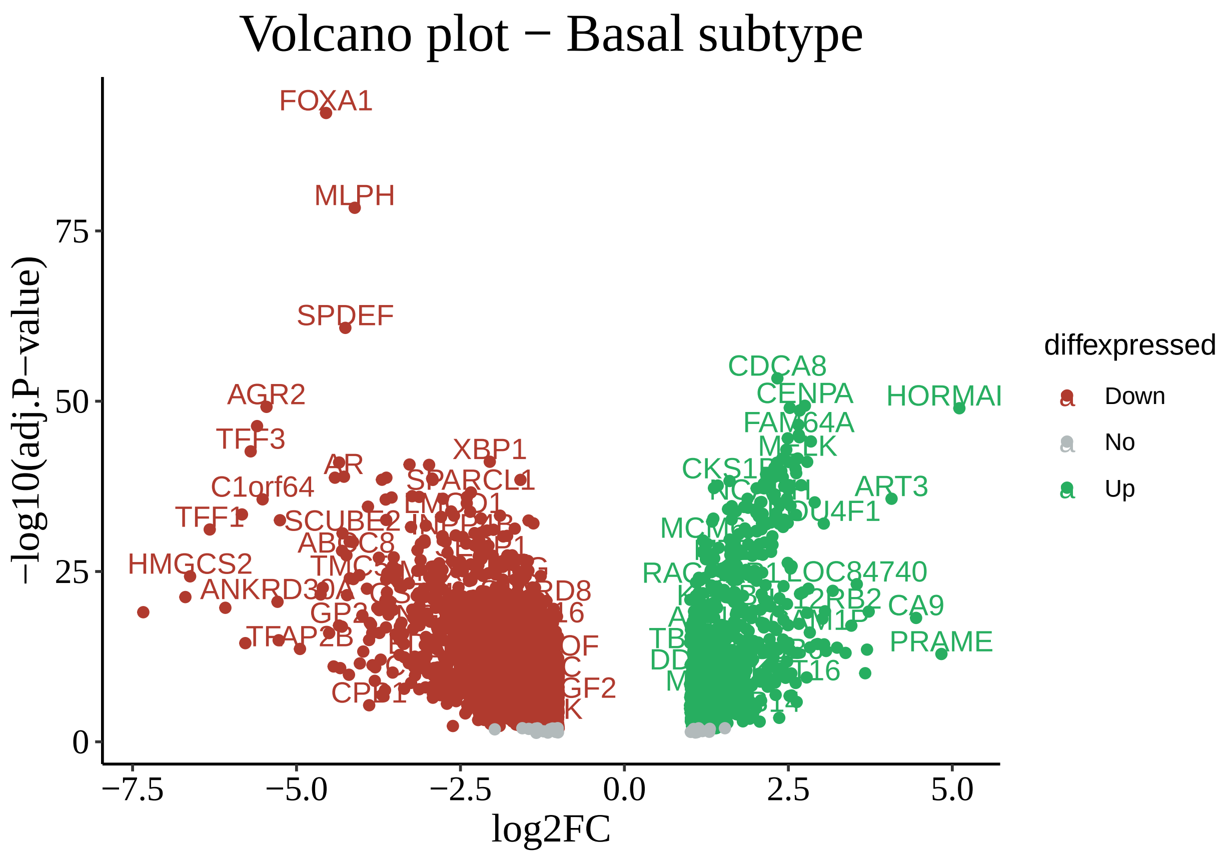
```
## [1] "The number of Down-regulated genes is 2210"
```

```
Top_10_Down_regulated <- Down_regulated[order(Down_regulated$logFC,
                                              decreasing = FALSE), c(1,5)]
# Top 10 down regulated genes
head(Top_10_Down_regulated, 10)
```

```
##             logFC    adj.P.Val
## SCGB2A2 -7.337089 9.471080e-20
## MUCL1   -6.695324 5.650969e-22
## HMGCS2  -6.623130 5.143322e-25
## TFF1    -6.323837 6.768947e-32
## PIP     -6.085665 2.127300e-20
## ABCC11  -5.832444 4.229299e-34
## SCGB1D2 -5.781022 3.289907e-15
## TFF3    -5.699471 2.302837e-43
## AGR3    -5.601868 4.492949e-47
## C1orf64 -5.517075 2.546738e-36
```

```
# Volcano plot
volcano_df <- DEGs_Basal
volcano_df$genes <- rownames(volcano_df)
rownames(volcano_df) <- NULL
volcano_df <- volcano_df[, c(7, 1, 5)]
volcano_df$log.padj <- -log10(volcano_df$adj.P.Val)
volcano_df <- volcano_df[,-3]
volcano_df$diffexpressed <- "No"
volcano_df$diffexpressed[volcano_df$log.padj > 2 & volcano_df$logFC > 0] <- "Up"
volcano_df$diffexpressed[volcano_df$log.padj > 2 & volcano_df$logFC < 0]<- "Down"
```

```r
volcano_df$dflabel <- NA
volcano_df$dflabel[volcano_df$diffexpressed != "No"] <-
  volcano_df$genes[volcano_df$diffexpressed != "No"]
ggplot(volcano_df, aes(logFC, log.padj, col=diffexpressed, label= dflabel)) +
labs(x= 'log2FC', y= '-log10(adj.P-value)') +
geom_point() + theme_classic() +
scale_color_manual(values = c("#B03A2E", "#B2BABB", "#27AE60")) +
geom_text(check_overlap = TRUE,vjust = 0.1, nudge_y = 0.7) +
theme(axis.text = element_text(family = "Times",size = 13 , colour = "black"),
axis.text.x = element_text(family = "Times",colour = "black", size = 13),
axis.text.y = element_text(family = "Times",colour = "black"),
plot.subtitle = element_text(family = "Times",size = 20, colour = "black", hjust = 0.5),
axis.title.y = element_text(family = "Times", size = rel(1.4), angle = 90),
axis.title.x = element_text(family = "Times", size = rel(1.4), angle = 00)) +
labs(subtitle = 'Volcano plot - Basal subtype')
```



```r
################################### ML #######################
genes1 <- rownames(DEGs_Basal)
genes2 <- rownames(DEGs_Her2)
genes3 <- rownames(DEGs_LumA)
genes4 <- rownames(DEGs_LumB)


# Helper function to combine genes from all DEGs
type <- "type"
gene <- "gene"
```

```r
prepare <- function(name, value, xname = type, yname = gene) {
  tibble(rep(name, length(value)), value) %>%
    set_colnames(c(xname, yname))
}

genes <- data.frame(bind_rows(
  prepare("g1", genes1),
  prepare("g2", genes2),
  prepare("g3", genes3),
  prepare("g4", genes4)
))
# Removing duplicates
genes <- unique(genes$gene)
# Number of DEGs from all subtypes
length(genes)
```

```
## [1] 5210
```

```r
# Making a new data frame based on DEGs across all subtypes
up_down_genes <- which(rownames(exp) %in% genes)
ml_df <- exp[up_down_genes, ]
ml_df <- data.frame(t(ml_df))
identical(pheno$sample, rownames(ml_df))
```

```
## [1] TRUE
```

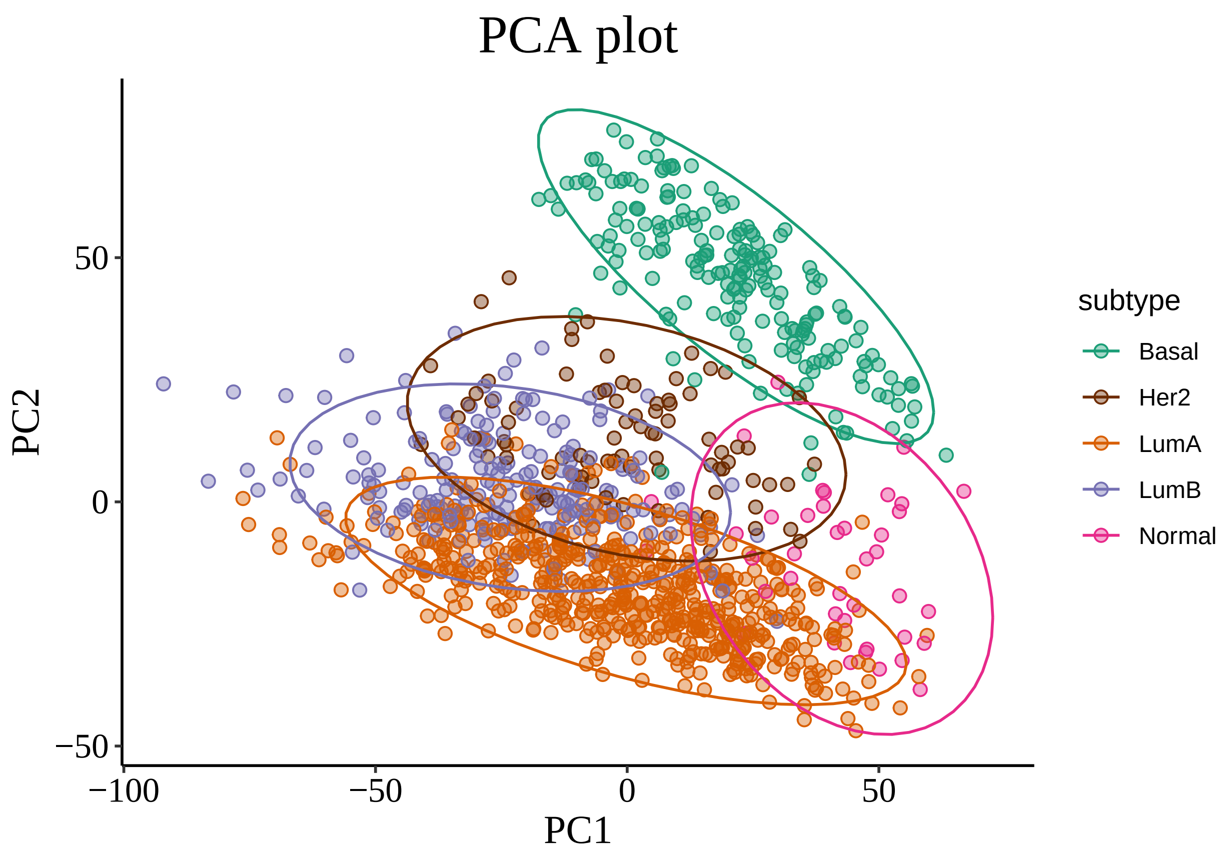
```r
ml_df$subtype <- pheno$subtype

# Making a new data frame based on DEGs across all subtypes
up_down_genes <- which(rownames(exp) %in% genes)
ml_df <- exp[up_down_genes, ]
ml_df <- data.frame(t(ml_df))
identical(pheno$sample, rownames(ml_df))
```

```
## [1] TRUE
```

```r
ml_df$subtype <- pheno$subtype

## plotting with PCA to visualize our result from DEG analysis
pca_df <- ml_df[,-NCOL(ml_df)]
pca_df <- scale(pca_df)
pca = preProcess(x = pca_df, method = 'pca', pcaComp = 2)
pca_df <- data.frame(predict(pca, pca_df))
pca_df$subtype <- pheno$subtype
my_pal <- c("#1B9E77", "#6E2C00","#D95F02", "#7570B3", "#E7298A",
            "#66A61E", "#E6AB02", "#A6761D", "#666666", "#9A7D0A")
ggplot(aes(x = PC1, y = PC2, color = subtype, fill = subtype), data =  pca_df) +
geom_point(size = 2, shape = 21) +
scale_color_manual(values=c(my_pal)) + stat_ellipse() +
scale_fill_manual(values=c(paste(my_pal, "66", sep = ""))) +
theme_classic() + theme(plot.title = element_text(hjust = 0.5),
axis.text = element_text(family = "Times",size = 13 , colour = "black"),
axis.text.x = element_text(family = "Times",colour = "black", size = 13),
axis.text.y = element_text(family = "Times",colour = "black"),
plot.subtitle = element_text(family = "Times",size = 20, colour = "black", hjust = 0.5),
axis.title.y = element_text(family = "Times", size = rel(1.4)),
axis.title.x = element_text(family = "Times", size = rel(1.4))) +
  labs(subtitle = 'PCA plot')
```



```
########################## Feature selection #########################
# Registering my only four cores
registerDoParallel(cores=4)
# Note: Due to prolonged computational time for this step, I skipped running
# codes bellow for this generating this RMarkdown file. However, I did this step under my own
# pace, and saved the result from the most important variables (Top 50).
# Therefor, I just read them from my disc.
# ml_df$subtype <- factor(ml_df$subtype)
# im_var <- train(sutype ~ .,
#                 data=ml_df,
#                 method='parRF',
#                 importance=TRUE,
#                 ntree=100)
# imp <- varImp(im_var)$importance %>%
#   data.frame()
# imp$gene <- rownames(imp)
# rownames(imp) <- NULL
# imp <- imp[order(imp$Basal, decreasing = T),]
# top50 <- imp[1:50,]
############## Making a new data set based on top50 variants
# im_gene_num <- which(colnames(ml_df) %in% top50$gene)
# ml_df_top50 <- data.frame(ml_df[, c(im_gene_num, 5211)])
# write_csv(ml_df_top50, "ml_df_top50.csv")
```

```r
# Reading the top 50 variables identified using random forest classifier
ml_df_top50 <- data.frame(read_csv("ml_df_top50.csv"))
head(ml_df_top50, 5)
```

```
##        A2ML1     ABCC8     AVPR1A     BCL11A     C1orf64    CCDC74B      CD79B
## 1 -0.6552687 0.5841863  0.2056613 -0.5698959 -1.63343532  0.6177424 -0.1802935
## 2 -0.3045549 0.5139104  0.1710634 -0.3749359 -0.16536258 -0.5252317  2.1765295
## 3 -1.0301532 0.3081655  0.4390920 -1.0865617 -1.10713944 -0.3090281  0.4489323
## 4 -0.6077072 0.1101270  0.5145910 -0.1354694  0.07246014  1.3900084  0.3821945
## 5 -0.3887322 0.5670027 -0.5017746 -0.1268630  0.96857525  0.8014512 -0.3869598
##        CDC25C      CDCA8       CDK1       CEP55       CKS1B        CPB1
## 1  0.78711463  0.7312545  0.3135528 -0.06688887 -0.55602716  1.86287496
## 2 -0.04525932  0.6019549  0.4232025  0.58091006  0.04893949 -0.44220027
## 3  0.35867657  0.4507214  1.5690760 -0.04749045  0.11104940  0.88541908
## 4 -0.39194568 -0.2089198 -0.2154981 -0.17245953 -0.66849335  0.07708767
## 5 -0.57551754 -0.5569618 -0.3897442 -0.54883519 -0.89411710 -0.73932860
##       CYP4Z2P       E2F5        ESR1     FAM72B        FGF1   FLJ33360
## 1 -0.4357560 -0.7427553  0.03235605  0.3287194  0.06092869 -0.0638937
## 2  0.8744230 -1.2763629 -1.72914132  0.7142133 -0.12128019  2.1042033
## 3 -1.9285763 -0.2402625  0.38029303  0.8065103  0.20765356  1.3679867
## 4  0.4964707 -0.5695434 -0.11648748 -0.7537710  0.86230237  2.4862768
## 5  0.4618382  0.3792574  0.24304323 -0.4971083  1.19312023 -0.5625229
##          FLT3     FOXA1      FOXC1        GDF5     GUSBP3         HTR7
## 1  0.4959784 0.2918636 -1.6355415  1.71863480  1.0125132 -0.098062505
## 2 -0.3198288 0.3478093 -0.9308167 -0.25525574  0.7655808 -0.006369826
## 3  1.6109678 0.2036629 -0.1084314 -0.26185771  1.6947967 -0.315533440
## 4  0.6023172 0.4521004  0.2714367 -0.08970621 -0.1322909  1.640136876
## 5 -0.8119767 0.4103172 -0.1531218 -0.09660827  0.2541689  1.191989939
##         IGFN1     IQGAP3  LOC145837     LRRC31        MIA      MLPH      MMP23B
## 1  0.3488494  1.2300378  1.1142286  1.0616905 -0.6945687 0.3539845 -0.12538900
## 2  0.5465810  1.2596586 -0.5307742  1.5726017 -0.5363686 1.0129162  0.07117655
## 3 -1.0833852  1.3647327  0.9623514 -1.2375037 -1.0934403 0.2460594  0.34478219
## 4  0.1954087  0.1204333  0.1921646  0.1071296  0.8164866 0.7028838  1.20707742
## 5 -1.0833852 -0.1091088 -0.2851935 -0.3896905  0.7274056 0.3600829  1.20928036
##          MUCL1      MYBL2      NCAPG       NUF2       OIP5       OVGP1
## 1 0.25968719  0.1136456  0.2172409  0.1887974  0.7329173 -0.34293695
## 2 0.02234072  1.5534405  0.4801760  0.7341971  0.1126820 -0.44584170
## 3 0.98452026  1.2392546  0.4698960  0.6862313  0.4821918  0.92756902
## 4 1.15818262 -0.2147929 -0.2013156 -0.4640661 -0.4435284  0.02422206
## 5 0.01461713 -0.2513116 -0.4888225 -0.7371700 -0.5665748  0.05639495
##          PALM2      PAMR1       PHEX      POC1A    RUNX1T1         SLIT3
## 1 -0.92855118 -0.6938997  0.3571677 -0.5106518 -0.3690807 -0.819500762
## 2 -0.44488642  0.4798002  1.1267964  0.9511122 -0.3049541  0.107425379
## 3 -0.74868879 -1.2656970 -1.8461985  0.8317947 -0.7959766 -0.009918558
## 4 -0.08238686  1.3044554 -0.6236773 -0.4231606  0.9075909  0.696343370
## 5 -0.08025242 -0.1029197 -0.2486136 -0.1071440  1.0876588  1.219615463
##          SYT13     TFAP2B       TPX2        TTK       UBE2T    UCKL1AS subtype
## 1  1.47040162  0.6039542  0.4892599 -0.7495335 -0.16410638  2.60565231    LumA
## 2 -0.03749244  1.1200804  1.1188359  0.1309450  1.28214357  2.39953773    Her2
## 3 -1.46189102 -1.9086971  0.2018651 -0.5534073  0.92591490  2.97760514    LumB
## 4 -0.48621121  0.8826654 -0.3002147 -0.6529001  0.03534321  0.01042754    LumA
## 5 -0.65981734  1.0393744 -0.5413539 -0.2966469 -0.79760583 -0.12478585    LumA
```

```r
# Train and test split for SVM classifier
ml_df_top50$subtype <- factor(ml_df_top50$subtype)
set.seed(123)
split <- sample.split(ml_df_top50[,51], SplitRatio = 0.7)
training_set <- subset(ml_df_top50, split == TRUE)
test_set <- subset(ml_df_top50, split == FALSE)

# Helper funtion to claculate confution matrix
confusion_matrix <- function(y_true, y_pred){
  if(!is.null(y_true) && !is.null(y_pred)){
    cm <- table(y_true, y_pred)
    if(dim(cm)[1] == 2){
      Accuracy <- (cm[1,1] + cm[2,2])/(cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
      Precision <- (cm[1,1])/(cm[1,1] + cm[1,2])
      Sensitivity <- (cm[1,1])/(cm[1,1] + cm[2,1])
      Specificity <- (cm[2,2])/ (cm[2,2] + cm[1,2])
      AUC <- roc(as.numeric(y_true) ~ as.numeric(y_pred), quiet = T)$auc[1]
      result <- round(data.frame(Accuracy = Accuracy,
                                 Precision = Precision,
                                 Sensitivity = Sensitivity,
                                 Specificity = Specificity,
                                 AUC = AUC),3)
      return(result)
    }
    else if (NROW(cm) > 2){
      TP <- list()
      for(i in 1:NROW(cm)){
        TP[[i]] <- cm[i,i]
      }
      TP <- data.frame(do.call(rbind,TP))
      FN <- rowSums(cm) - TP[,1]
      FP <- colSums(cm) - TP[,1]
      TN <- list()
      for(i in 1:NROW(cm)){
        TN[[i]] <- sum(cm) - sum(cm[i,]) - sum(cm[,i]) + cm[i,i]
      }
      TN <- data.frame(do.call(rbind, TN))
      con <- cbind(TP, FN, FP, TN)
      colnames(con) <- c("TP", "FN", "FP", "TN")
      rownames(con) <- rownames(cm)
      a <- list()
      for (i in 1:NROW(con)) {
        a[[i]] <- (con$TP[i] + con$TN[i])/(con$TP[i] + con$TN[i] + con$FN[i] + con$FP[i])
      }
      p <- list()
      for (i in 1:NROW(con)) {
        p[[i]] <- con$TP[i]/(con$TP[i] + con$FP[i])
      }
      se <- list()
      for (i in 1:NROW(con)) {
        se[[i]] <- con$TP[i]/(con$TP[i] + con$FN[i])
      }
      sp <- list()
      for (i in 1:NROW(con)) {
        sp[[i]] <- con$TN[i]/(con$TN[i] + con$FP[i])
      }
      a <- do.call(rbind, a)
      p <- do.call(rbind, p)
      se <- do.call(rbind, se)
      sp <- do.call(rbind, sp)
      au <- multiclass.roc(as.numeric(y_true) ~ as.numeric(y_pred), quiet = T)$auc[1]
      au <- rep(au, length.out= NROW(con))
      result <- round(cbind(a, p, se, sp, au),3)
      colnames(result) <- c("Accuracy",
                            "Precision",
                            "Sensitivity",
                            "Specificity",
                            "AUC_average")
      rownames(result) <- rownames(cm)

      return(result)
    }
  }
}


# Helper function to avarage the result from cross validation step
multiclass_con_av <- function(cv){
  mlm <- do.call(cbind, cv)
  colnames(mlm) <- gsub("Fold[0-9]{1,2}.", "", colnames(mlm))
  acc <- list()
  for(i in 1:5){
    name <- unique(colnames(mlm))[i]
    num <- grep(name, colnames(mlm))
    acc[[i]] <- rowMeans(mlm[,num])

  }
  result <- round(do.call(cbind, acc),3)
  colnames(result) <- unique(colnames(mlm))
  return(result)
}

# Performing 10 folds cross validation on SVM classifer
folds <- createFolds(ml_df_top50[,51] , k = 10)
cv <- lapply(folds, function(x){
  training_fold <- training_set[-x, ]
  test_fold <- test_set[-x, ]
  classifier <- svm(formula = subtype ~ .,
                    data = training_fold,
                    type = "C-classification",
                    kernel = "linear",
                    cost = 4,
                    tolerance = 0.001,
                    na.action = na.omit,
                    scale = FALSE)
  y_pred <- predict(classifier, newdata = test_fold[-51])
  result <- confusion_matrix(test_fold[, 51], y_pred)
  return(result)
})
# Result
multiclass_con_av(cv)
```

```
##        Accuracy Precision Sensitivity Specificity AUC_average
## Basal     0.984     0.943       0.965       0.988       0.846
## Her2      0.972     0.860       0.764       0.990       0.846
## LumA      0.917     0.906       0.937       0.895       0.846
## LumB      0.928     0.823       0.801       0.958       0.846
## Normal    0.974     0.691       0.530       0.991       0.846
```