Model #3

Ethan Allavarpu (UID: 405287603)

10/28/2020

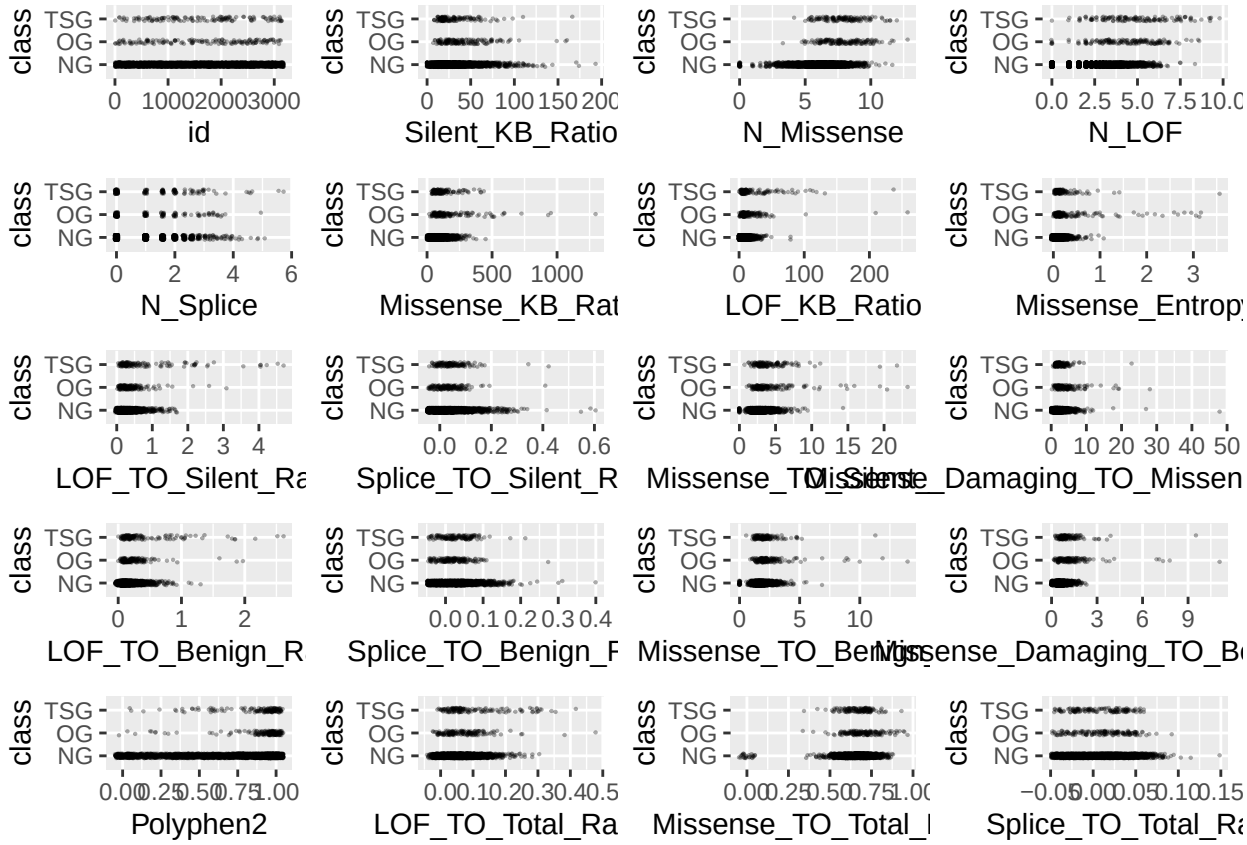
Transforming and Cleaning the Data

```
training <- read.csv("training.csv", stringsAsFactors = TRUE)
training$class <- factor(training$class)
levels(training$class) <- c("NG", "OG", "TSG")
outlier <- function(data) {
  low <- mean(data) - 3 * sd(data)
  high <- mean(data) + 3 * sd(data)
  which(data < low | data > high)
}
library(ggplot2)
scatter <- function(var) {
  ggplot(training, aes_string(var, "class")) +
    geom_jitter(width = 0.05, height = 0.1, size = 0.1,
                colour = rgb(0, 0, 0, alpha = 1 / 3))
}
scat_plot <- lapply(names(training)[-99], scatter)
library(gridExtra)
# grid.arrange(grobs = scat_plot[1:20], ncol = 4)
# grid.arrange(grobs = scat_plot[21:40], ncol = 4)
# grid.arrange(grobs = scat_plot[41:60], ncol = 4)
# grid.arrange(grobs = scat_plot[61:80], ncol = 4)
# grid.arrange(grobs = scat_plot[81:98], ncol = 4)
outlier_index <- sort(table(unlist(lapply(training[,-99], outlier))), decreasing = TRUE)
outlier_index[1:100]
```

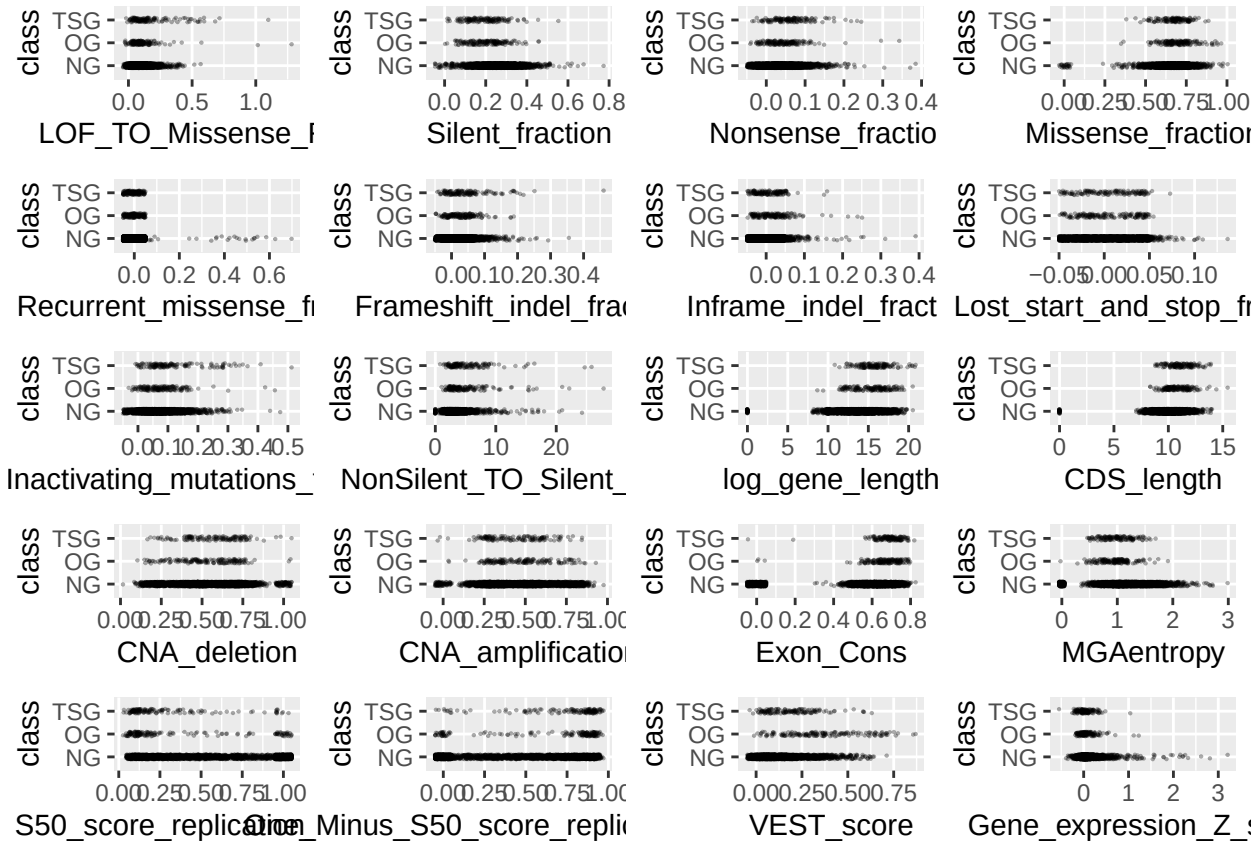
```
##
##  915 1280 2918  517 1914 2182 3052 1173 2215 3049  259  740 1749 1979 2998  417
##   24   24   24   22   22   22   20   19   19   19   18   18   18   18   18   17
##  441  806 2297  422  635 1258 1570 2278 2518 2729   80  150 2694  169  276  341
##   17   17   17   16   16   16   16   16   16   16   15   15   15   14   14   14
## 1528 1556 1726 1809 1911 1955 2071 2624 2641 3120 3142   73  277  364  751 1244
##   14   14   14   14   14   14   14   14   14   14   14   13   13   13   13   13
## 1330 2329 2787  343 1138 1171 1188 1372 1460 2031 2251 2968 2983 3166  352  634
##   13   13   13   12   12   12   12   12   12   12   12   12   12   12   11   11
##  907  923 1096 1858 2636  588 1137 1317 1463 1561 1740 1991 2487 2540 2555 2621
##   11   11   11   11   11   10   10   10   10   10   10   10   10   10   10   10
## 2815 3029   74  144  657  789  857 1267 1610 1932 2022 2093 2142 2534 2666 2721
##   10   10    9    9    9    9    9    9    9    9    9    9    9    9    9    9
## 2848 2900 3027  155
##    9    9    9    8
```

```
training <- training[-as.numeric(names(outlier_index)[1:50]),]
training <- training[-which(training$Missense_TO_Silent_Ratio > 100), ]
training <- training[-which(training$Missense_KB_Ratio > 2000), ]
training <- training[-which(training$LOF_TO_Silent_Ratio > 5), ]
training <- training[-which(training$Gene_expression_Z_score > 4), ]
training <- training[-which(training$dN_to_dS_ratio > 5),]
training <- training[-which(training$Silent_KB_Ratio > 200), ]
training <- training[-which(training$Lost_start_and_stop_fraction > 0.2),]
```

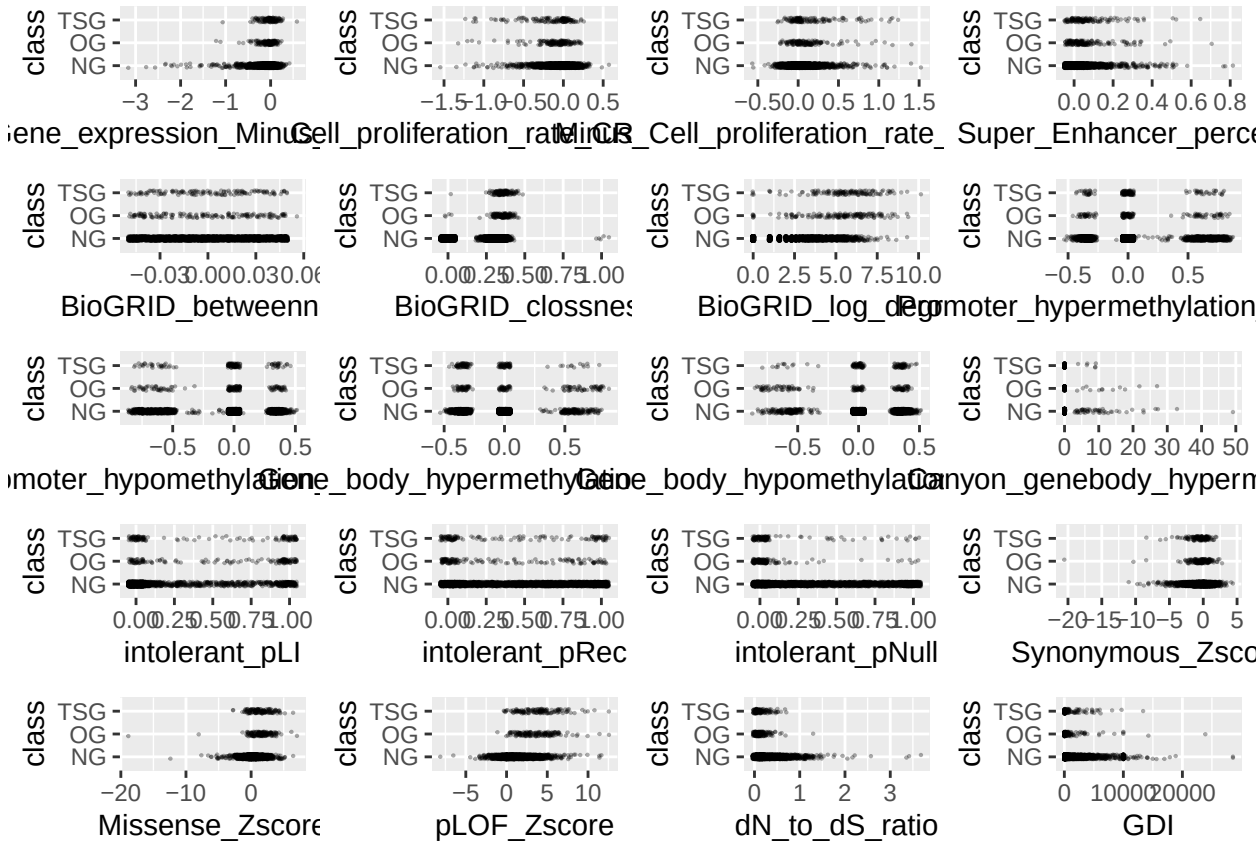
```
library(ggplot2)
scatter <- function(var) {
  ggplot(training, aes_string(var, "class")) +
    geom_jitter(width = 0.05, height = 0.1, size = 0.1,
                colour = rgb(0, 0, 0, alpha = 1 / 3))
}
scat_plot <- lapply(names(training)[-99], scatter)
library(gridExtra)
grid.arrange(grobs = scat_plot[1:20], ncol = 4)
```



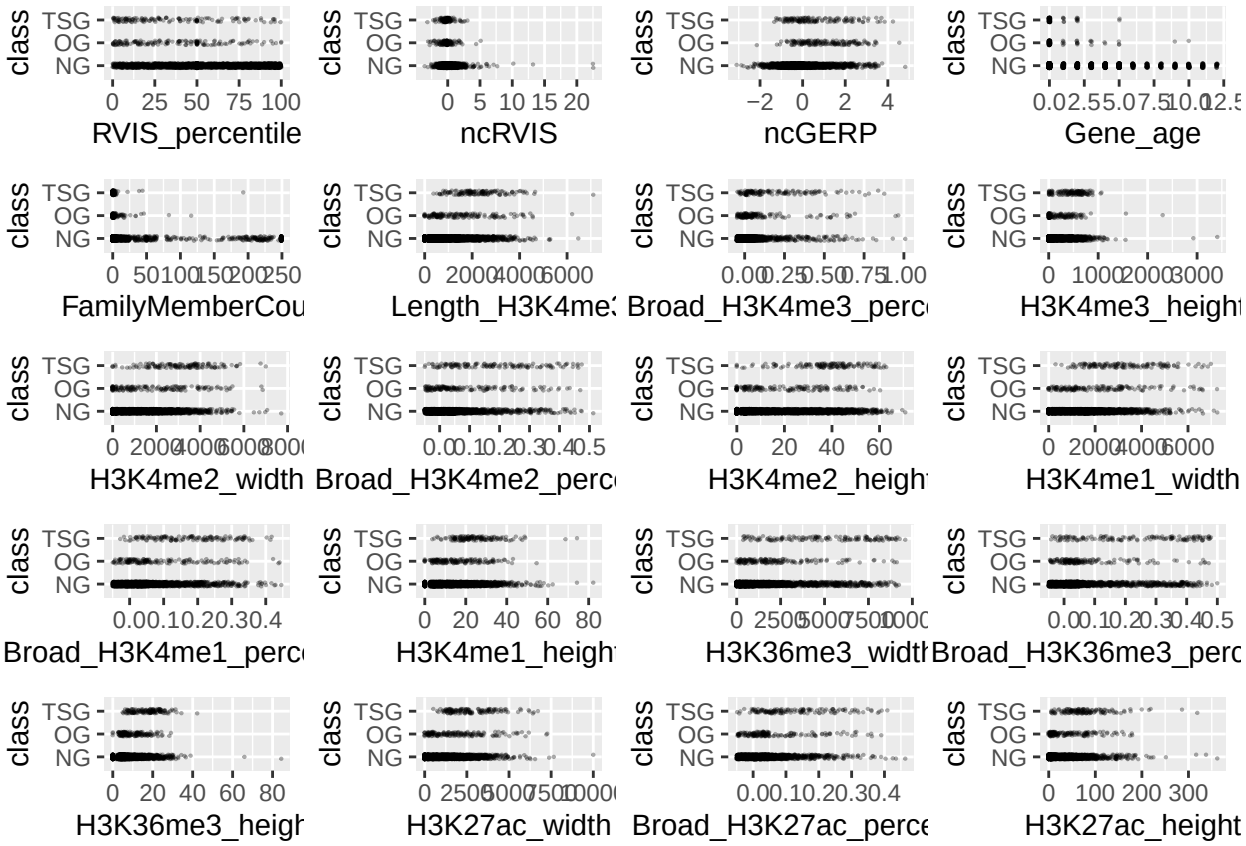

```
grid.arrange(grobs = scat_plot[21:40], ncol = 4)
```



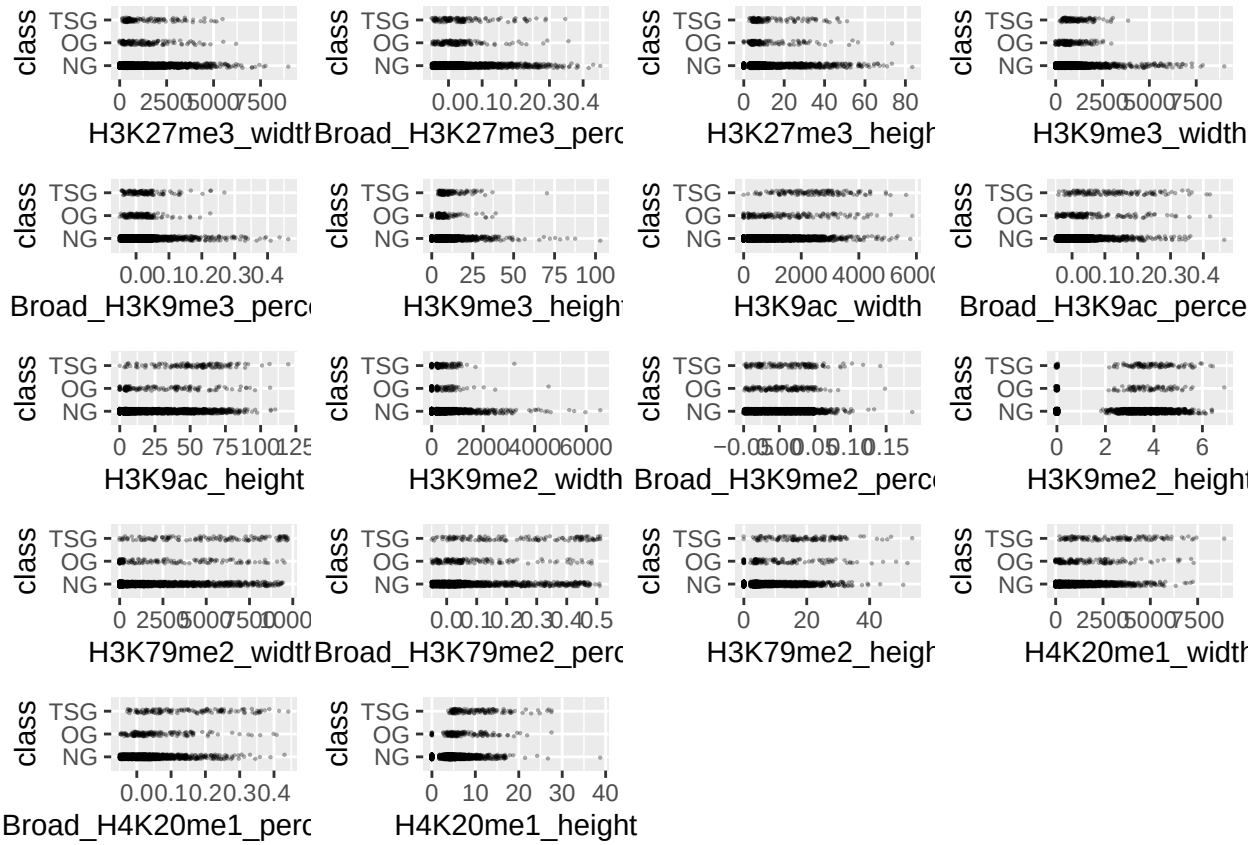

```
grid.arrange(grobs = scat_plot[41:60], ncol = 4)
```




```
grid.arrange(grobs = scat_plot[61:80], ncol = 4)
```




```
grid.arrange(grobs = scat_plot[81:98], ncol = 4)
```




```
library(dplyr)
```

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

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

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

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

```
sig <- logical(98)
names(sig) <- names(training)[-99]
k <- 1
diffs <- logical(98)
for (var in names(training)[-99]) {
  model <- aov(training[[var]] ~ factor(training$class))
  sig[k] <- summary(model)[[1]][1, 5]
  diffs[k] <- all(TukeyHSD(model)$`factor(training$class)`[, 4] < 0.05)
  k <- k + 1
}
sort(sig[diffs])
```

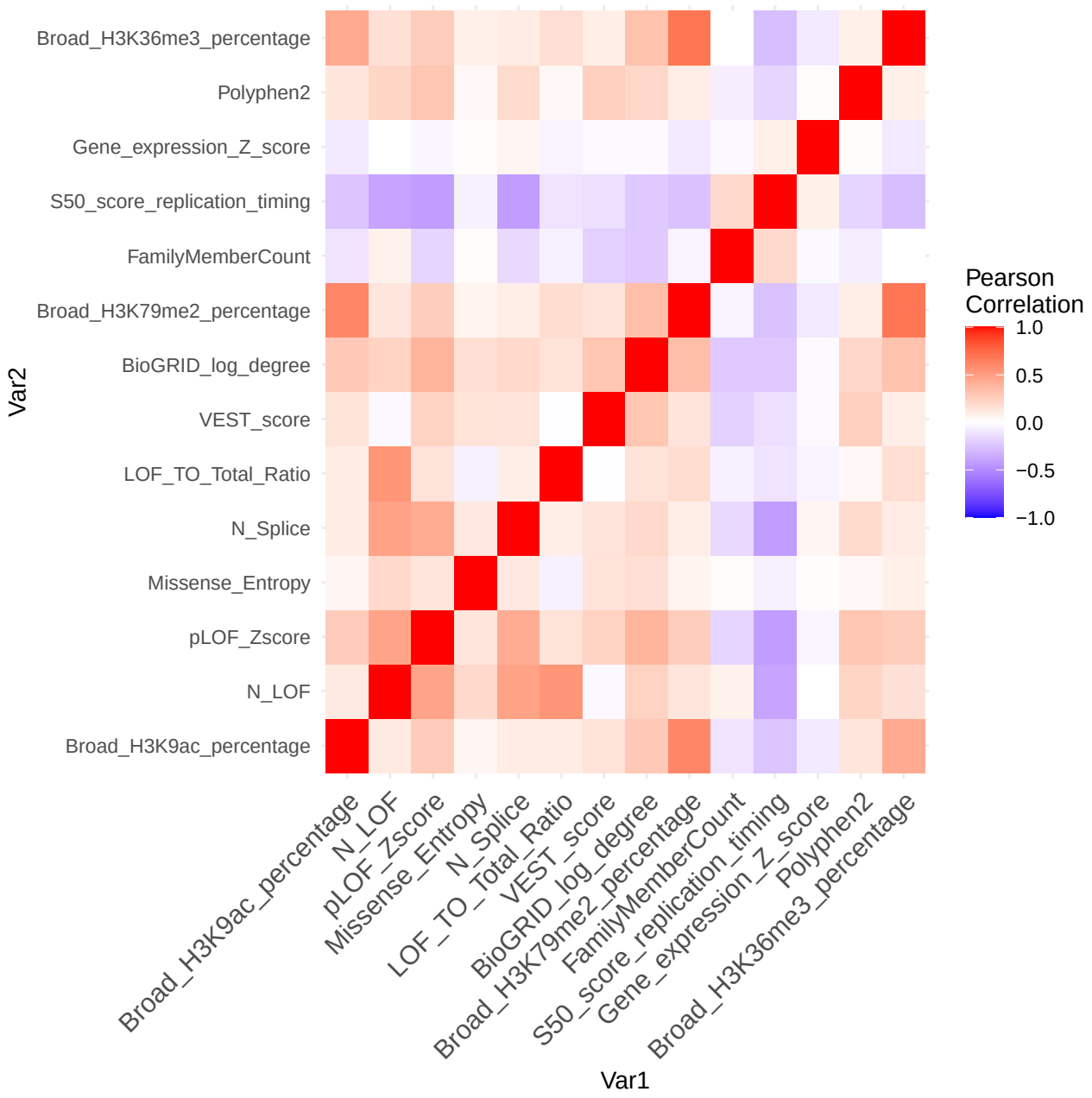
```
##                  Broad_H4K20me1_percentage
##                                6.698302e-151
##                                   VEST_score
##                                3.697355e-125
##                    Broad_H3K9ac_percentage
##                                1.608465e-114
##                           H3K79me2_height
##                                3.175984e-114
##                           H3K79me2_width
##                                1.230355e-113
##                          Missense_Entropy
##                                7.000817e-112
##                  Broad_H3K79me2_percentage
##                                1.225438e-110
##                           intolerant_pLI
##                                5.355081e-110
##                   Broad_H3K4me2_percentage
##                                3.250962e-109
##                           H4K20me1_width
##                                1.919147e-108
##          Missense_Damaging_TO_Benign_Ratio
##                                1.295984e-107
##                  Broad_H3K36me3_percentage
##                                3.144443e-107
##                           H3K36me3_width
##                                2.038220e-106
##                          H4K20me1_height
##                                1.264678e-103
##                       LOF_TO_Silent_Ratio
##                                1.896468e-102
##                   Broad_H3K27ac_percentage
##                                3.134484e-100
##                   Broad_H3K4me1_percentage
##                                 4.772760e-99
##                                 LOF_KB_Ratio
##                                 4.232319e-96
##                         Missense_KB_Ratio
##                                 3.282539e-91
##                                       N_LOF
##                                 7.709721e-91
##                           H3K4me1_width
##                                 1.276727e-82
##                   Broad_H3K4me3_percentage
##                                 7.892674e-81
##                       LOF_TO_Benign_Ratio
##                                 2.438589e-79
##                          H3K36me3_height
##                                 2.489709e-77
##                           H3K4me2_width
##                                 1.956413e-76
##                            H3K9ac_width
##                                 2.813619e-74
##                           H3K9ac_height
##                                 5.352308e-68
```

```
##                               H3K27ac_width
##                                1.174166e-65
##                              Length_H3K4me3
##                                4.430014e-63
##                    Missense_TO_Benign_Ratio
##                                3.089679e-62
##                                      ncGERP
##                                5.454806e-61
##                              H3K27ac_height
##                                1.362616e-60
## Missense_Damaging_TO_Missense_Benign_Ratio
##                                9.141445e-55
##                              H3K4me3_height
##                                2.040044e-52
##                              H3K4me2_height
##                                8.438648e-51
##                                    N_Splice
##                                8.391748e-49
##                           LOF_TO_Total_Ratio
##                                2.203794e-48
##                              H3K4me1_height
##                                7.389259e-44
##                          LOF_TO_Missense_Ratio
##                                2.993957e-43
##                    Frameshift_indel_fraction
##                                3.110939e-34
##              Inactivating_mutations_fraction
##                                3.166691e-32
##           Cell_proliferation_rate_CRISPR_KD
##                                3.400357e-23
##     Minus_Cell_proliferation_rate_CRISPR_KD
##                                3.400357e-23
##                       Inframe_indel_fraction
##                                1.795304e-12
```

```
score <- function (conf_mat) {
  print(sum(diag(conf_mat) * c(1, 20, 20)))
  print(sum(diag(conf_mat) * c(1, 20, 20)) / sum(apply(conf_mat, 2, sum) * c(1, 20, 20)))
}

classify <- function(probs) {
  if (any(probs[2:3] > 0.05)) {
    subset <- probs[2:3]
    output <- which(subset == max(subset))
    if (length(output) > 1) {
        output <- sample(1:2, 1)
    }
  } else {
    output <- 0
  }
  output
}
```

```
library(dplyr)
vars <- training %>% select(Broad_H3K9ac_percentage, N_LOF, pLOF_Zscore,
```

```
                             Missense_Entropy,
                            N_Splice, LOF_TO_Total_Ratio, VEST_score,
                            BioGRID_log_degree,
                            Broad_H3K79me2_percentage, FamilyMemberCount,
                            S50_score_replication_timing, Gene_expression_Z_score,
                            Polyphen2, Broad_H3K36me3_percentage, class)
vars$class <- factor(vars$class)
levels(vars$class) <- c("NG", "OG", "TSG")
cor_mtx = round(cor(vars[, names(vars) != "class"]), 2)
library(reshape2)
#reshape it
melted_cor_mtx <- melt(cor_mtx)

#draw the heatmap
cor_heatmap = ggplot(data = melted_cor_mtx, aes(x=Var1, y=Var2, fill=value)) + geom_tile()
cor_heatmap = cor_heatmap +
scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0, limit = c(-1,1), space =
theme_minimal() +
theme(axis.text.x = element_text(angle = 45, vjust = 1, size = 12, hjust = 1))

cor_heatmap
```




```
library(dplyr)
set.seed(12)
vars <- training %>% select(Broad_H3K9ac_percentage, N_LOF, pLOF_Zscore,
                             Missense_Entropy, log_gene_length, Missense_Damaging_TO_Benign_Ratio,
                            N_Splice, LOF_TO_Total_Ratio, VEST_score,
                            BioGRID_log_degree,
                            Broad_H3K79me2_percentage, FamilyMemberCount,
                            S50_score_replication_timing, Gene_expression_Z_score,
                            Polyphen2, Broad_H3K36me3_percentage, class)
library(caret)
```

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

```
vars_test <- createDataPartition(vars$class, p = 0.8,
                                 list = FALSE)
vars_train <- vars[vars_test, ]
vars_test <- vars[-vars_test, ]

train_cont <- trainControl(method = "cv", number = 5, classProbs = TRUE, savePredictions = TRUE)


knn_ft <- train(class ~ ., data = vars_train, method = "knn", preProc = c("center", "scale"),
                trControl = train_cont, tuneGrid = expand.grid(k = seq(from = 1, to = 25, by = 5)))
for (k in seq(from = 1, to = 25, by = 5)) {
  preds <- predict(knn_ft, newdata = vars_test, type = "prob")
  knn_mod <- table("pred"=unlist(apply(preds, 1, classify)), "obs" = vars_test$class)
  print(knn_mod)
  score(knn_mod)
}
```

```
##     obs
## pred  NG  OG TSG
##    0 503   3   4
##    1  27  21   2
##    2  35   5  20
## [1] 1323
## [1] 0.7945946
##     obs
## pred  NG  OG TSG
##    0 503   3   4
##    1  27  21   3
##    2  35   5  19
## [1] 1303
## [1] 0.7825826
##     obs
## pred  NG  OG TSG
##    0 503   3   4
##    1  29  21   3
##    2  33   5  19
## [1] 1303
## [1] 0.7825826
##     obs
## pred  NG  OG TSG
##    0 503   3   4
##    1  26  21   3
##    2  36   5  19
## [1] 1303
## [1] 0.7825826
##     obs
## pred  NG  OG TSG
##    0 503   3   4
##    1  26  22   2
##    2  36   4  20
## [1] 1343
## [1] 0.8066066
```

```
train_cont <- trainControl(method = "cv", number = 5, classProbs = TRUE, savePredictions = TRUE)
qda_ft <- train(class ~ ., data = vars_train, method = "qda", preProc = c("center", "scale"),
                trControl = train_cont)
preds <- predict(qda_ft, newdata = vars_test, type = "prob")

qda_mod <- table("pred"=apply(preds, 1, classify), "obs" = vars_test$class)
qda_mod
```

```
##     obs
## pred  NG  OG TSG
##    0 498   3   1
##    1  26  21   8
##    2  41   5  17
```

```
score(qda_mod)
```

```
## [1] 1258
## [1] 0.7555556
```

```
train_cont <- trainControl(method = "cv", number = 5, classProbs = TRUE, savePredictions = TRUE)
lda_ft <- train(class ~ ., data = vars_train, method = "lda", preProc = c("center", "scale"),
                trControl = train_cont)
preds <- predict(lda_ft, newdata = vars_test, type = "prob")

lda_mod <- table("pred"=apply(preds, 1, classify), "obs" = vars_test$class)
lda_mod
```

```
##     obs
## pred  NG  OG TSG
##    0 523   3   2
##    1  17  22   3
##    2  25   4  21
```

```
score(lda_mod)
```

```
## [1] 1383
## [1] 0.8306306
```

```
tests <- read.csv("test.csv")
rel_vars <- tests %>% select(Broad_H3K9ac_percentage, N_LOF, pLOF_Zscore,
                             Missense_Entropy, log_gene_length, Missense_Damaging_TO_Benign_Ratio,
                            N_Splice, LOF_TO_Total_Ratio, VEST_score,
                            BioGRID_log_degree, H4K20me1_height,
                            Broad_H3K79me2_percentage, FamilyMemberCount,
                            S50_score_replication_timing, Gene_expression_Z_score,
                            Polyphen2, Broad_H3K36me3_percentage, class)
for (i in 1:14) {
  rel_vars[[i]] <- (rel_vars[[i]] - lda_ft$preProcess$mean[i])/ lda_ft$preProcess$std[i]
}
preds <- predict(lda_ft, newdata = rel_vars, type = "prob")
preds <- apply(preds, 1, classify)
names(preds) <- tests$id
csv_file <- data.frame("id" = tests$id,
                       "class" = preds)
#write.csv(csv_file, "modelpredictions2.csv", row.names = FALSE)
model_1 <- read.csv("modelpredictions.csv")
```

```r
model_2 <- read.csv("modelpredictions2.csv")
model_1_beta <- read.csv("modelpredictions_beta.csv")
table(model_1$class, model_1_beta$class)
```

```
##
##       0   1   2
##   0 862 113 111
##   1  15  95  24
##   2   0   0 143
```

```r
table("BEST MOD" = model_2$class, csv_file$class)
```

```
##
## BEST MOD   0   1   2
##        0 827  13  51
##        1   6 149  19
##        2  33  26 239
```

```r
mean(model_1$class == csv_file$class)
```

```
## [1] 0.7813646
```