Reproducibility Project

Zihan Wang

2025-04-09

```
knitr::opts_chunk$set(echo = TRUE, message = FALSE, warning = FALSE,
                      tidy = TRUE,tidy.opts = list(width.cutoff = 60))
knitr::opts_knit$set(root.dir = "/Users/david/Downloads/Reproducibility Project")
```

Loading Package

```
library(ComplexHeatmap)
library(dplyr)
library(gbm)
library(ggplot2)
library(glmnet)
library(gt)
library(gtsummary)
library(pheatmap)
library(randomForest)
library(survival)
library(table1)
library(xtable)
library(caret)
library(pROC)
library(ROCR)
library(randomForest)
library(xgboost)
library(Metrics)
library(knitr)
library(kableExtra)
library(formatR)
```

table1 (There exists some differences between the paper and my results regarding "Primary Surgery". The classification criteria seems a little vague)

```
### Load the dataf for table 1
dataframe = read.delim("UC_GENOME_Clinical.txt", comment.char = "#",
    header = TRUE, sep = "\t", stringsAsFactors = FALSE)


### Modify the dataframe
dataframe = dataframe %>%
```

```
    rename(`Age at diagnosis` = AGE_AT_DIAGNOSIS, Sex = SEX,
        Race = RACE, `ECOG PS` = BASELINE_ECOG, `Smoking status` = SMOKING_STATUS,
        `Tumor origin at initial diagnosis` = PRIMARY_TUMOR_LOCATION,
        `Primary Surgery` = PRIMARY_SURGERY, Neoadjuvant = NEOADJUVANT_THERAPY,
        Adjuvant = ADJUVANT_THERAPY, `Any Systemic Therapy` = SYSTEMIC_THERAPY,
        `Targeted Therapy` = TARGETED_THERAPY, Chemotherapy = CHEMOTHERAPY,
        Immunotherapy = IMMUNOTHERAPY, `Antibody-drug Conjugate Therapy` = ADC_THERAPY,
        Survival = SURVIVAL_STATUS, `Cause of Death` = DEATH_REASON) %>%
    mutate(Sex = ifelse(Sex == "U", "Unknown", Sex), `Tumor origin at initial diagnosis` = ifelse(grepl
        `Tumor origin at initial diagnosis`), ">1 site", `Tumor origin at initial diagnosis`)) %>%
    mutate(`ECOG PS` = as.factor(`ECOG PS`)) %>%
    mutate(`Tumor origin at initial diagnosis` = factor(`Tumor origin at initial diagnosis`,
        levels = c("Bladder", "Ureter", "Renal pelvis", "Urethra",
            ">1 site"))) %>%
    mutate(`Primary Surgery` = case_when(grepl("cystectomy|cystoprostatectomy|Cystoprotatectomy|cytopros
        `Primary Surgery`, ignore.case = TRUE) ~ "Radical Cystectomy",
        grepl("Nephro-ureterectomy|Nephroureterectomy with endoscopic bladder cuff resection and lympha
            `Primary Surgery`, ignore.case = TRUE) ~ "Radical nephrouterectomy",
        grepl("biopsy", `Primary Surgery`, ignore.case = TRUE) |
            `Primary Surgery` == "" | is.na(`Primary Surgery`) ~
            "No Primary Surgery", TRUE ~ `Primary Surgery`)) %>%
    group_by(`Primary Surgery`) %>%
    mutate(n = n()) %>%
    ungroup() %>%
    mutate(`Primary Surgery` = ifelse(n <= 2, "Other", `Primary Surgery`)) %>%
    select(-n)


dataframe$Neoadjuvant[dataframe$Neoadjuvant != "Yes"] = "No"
dataframe$Adjuvant[dataframe$Adjuvant != "Yes"] = "No"
dataframe$`Targeted Therapy`[dataframe$`Targeted Therapy` !=
    "Yes"] = "No"
dataframe$Chemotherapy[dataframe$Chemotherapy != "Yes"] = "No"
dataframe$Immunotherapy[dataframe$Immunotherapy != "Yes"] = "No"
dataframe$`Antibody-drug Conjugate Therapy`[dataframe$`Antibody-drug Conjugate Therapy` !=
    "Yes"] = "No"


### Generate the table1

summary_table1 = table1(~`Age at diagnosis` + Sex + Race + `ECOG PS` +
    `Smoking status` + `Tumor origin at initial diagnosis` +
    `Primary Surgery` + Neoadjuvant + Adjuvant + `Any Systemic Therapy` +
    `Targeted Therapy` + Chemotherapy + Immunotherapy + `Antibody-drug Conjugate Therapy` +
    Survival + `Cause of Death`, data = dataframe)

kable(as.data.frame(summary_table1), format = "latex", booktabs = TRUE,
    longtable = TRUE) %>%
    kable_styling(latex_options = c("hold_position"))
```

	Overall
	(N=218)

Age at diagnosis	
Mean (SD)	65.5 (9.99)
Median [Min, Max]	66.0 [28.0, 85.0]
Sex	
Female	55 (25.2%)
Male	162 (74.3%)
Unknown	1 (0.5%)
Race	
Asian	5 (2.3%)
Black or African American	19 (8.7%)
Unknown	26 (11.9%)
White	168 (77.1%)
ECOG PS	
0	87 (39.9%)
1	96 (44.0%)
2	28 (12.8%)
3	7 (3.2%)
Smoking status	
Current Smoker	18 (8.3%)
Former Smoker	130 (59.6%)
Never Smoker	69 (31.7%)
Unknown	1 (0.5%)
Tumor origin at initial diagnosis	
Bladder	182 (83.5%)
Ureter	12 (5.5%)
Renal pelvis	11 (5.0%)
Urethra	1 (0.5%)
>1 site	12 (5.5%)
Primary Surgery	
No Primary Surgery	28 (12.8%)
Other	7 (3.2%)
Radical Cystectomy	105 (48.2%)
Radical nephrouterectomy	27 (12.4%)
TURBT (transurethral resection of bladder tumor)	51 (23.4%)
Neoadjuvant	
No	139 (63.8%)
Yes	79 (36.2%)
Adjuvant	
No	161 (73.9%)
Yes	57 (26.1%)
Any Systemic Therapy	
No	30 (13.8%)
Unknown	1 (0.5%)
Yes	187 (85.8%)
Targeted Therapy	
No	204 (93.6%)
Yes	14 (6.4%)
Chemotherapy	
No	53 (24.3%)
Yes	165 (75.7%)

Immunotherapy	
No	62 (28.4%)
Yes	156 (71.6%)
Antibody-drug Conjugate Therapy	
No	202 (92.7%)
Yes	16 (7.3%)
Survival	
Alive	90 (41.3%)
Dead	99 (45.4%)
Unknown	29 (13.3%)
Cause of Death	
	119 (54.6%)
Due to complications of treatment	3 (1.4%)
Due to disease	91 (41.7%)
Due to other cause	3 (1.4%)
Unknown	2 (0.9%)

Figure 1(b) (Same results)

```
### Load the data for figure 1b

dataframe_1b = read.delim("UC_GENOME_Clinical_Expression_Markers.txt",
    header = TRUE, sep = "\t", stringsAsFactors = FALSE)

### Reorder according to the paper

dataframe_1b = dataframe_1b[order(dataframe_1b$HeatmapOrder),
    ]

group = factor(dataframe_1b$CSubtypes)
names(group) = dataframe_1b$SampleID

expr_matrix = t(as.matrix(dataframe_1b[, 9:ncol(dataframe_1b)]))

ha = HeatmapAnnotation(Subtype = group, col = list(Subtype = c(Ba.Sq = "#D84129",
    LumP = "#8EC751", Stroma.rich = "#E6E34E", LumU = "#577BBE",
    NE.like = "#B25BB5", LumNS = "#61A060")), show_legend = FALSE,
    annotation_name_gp = gpar(fontsize = 0))


my_colors = colorRampPalette(c("blue", "white", "red"))(50)

Heatmap(expr_matrix, name = "Expression", col = my_colors, top_annotation = ha,
    show_column_names = FALSE, row_names_gp = gpar(fontsize = 6),
    cluster_rows = FALSE, cluster_columns = FALSE, show_heatmap_legend = FALSE)
```

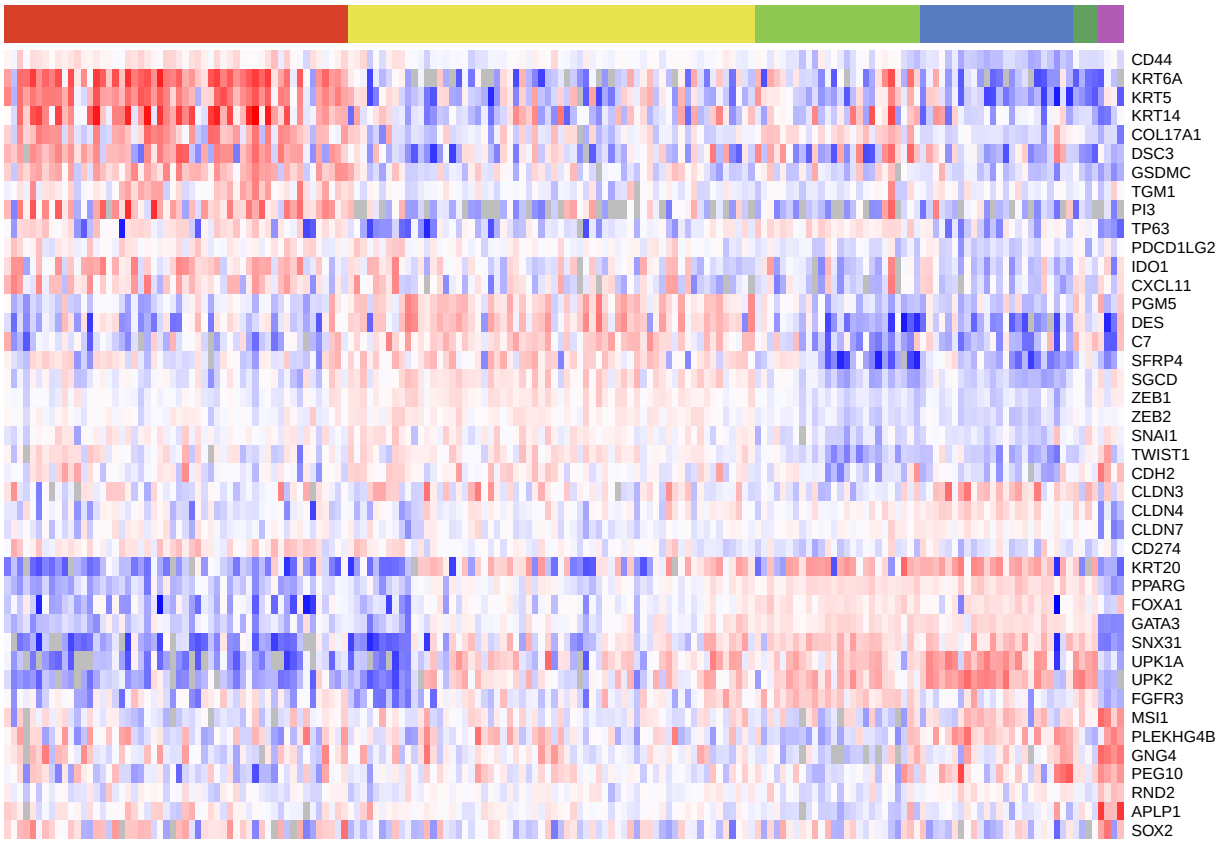


Figure 1(c) (Same results)

```
### Use the same data as figure 1b

dataframe_1c = dataframe_1b %>%
    select(1:3) %>%
    na.omit()

dataframe_1c$Gender = ifelse(dataframe_1c$Gender == "U", "Unknown",
    dataframe_1c$Gender)
dataframe_1c$Gender = ifelse(dataframe_1c$Gender == "F", "Female",
    dataframe_1c$Gender)
dataframe_1c$Gender = ifelse(dataframe_1c$Gender == "M", "Male",
    dataframe_1c$Gender)

### Reorder

dataframe_1c$Gender = factor(dataframe_1c$Gender, levels = c("Unknown",
    "Female", "Male"))
dataframe_1c$CSubtypes = factor(dataframe_1c$CSubtypes, levels = c("Ba.Sq",
    "Stroma.rich", "LumP", "LumU", "LumNS", "NE.like"))

### Get the count numbers and frequency
```

```
df_plot_1c = dataframe_1c %>%
    count(CSubtypes, Gender) %>%
    group_by(CSubtypes) %>%
    mutate(Proportion = n/sum(n)) %>%
    ungroup()

### Generate the stacked bar chart

ggplot(df_plot_1c, aes(x = CSubtypes, y = n, fill = Gender)) +
    geom_bar(stat = "identity") + geom_text(aes(label = round(Proportion,
    2)), position = position_stack(vjust = 0.5), size = 4) +
    scale_fill_manual(values = c(Female = "red2", Male = "skyblue",
        Unknown = "darkgreen")) + ylab("patients") + xlab(NULL) +
    theme_classic(base_size = 14)
```

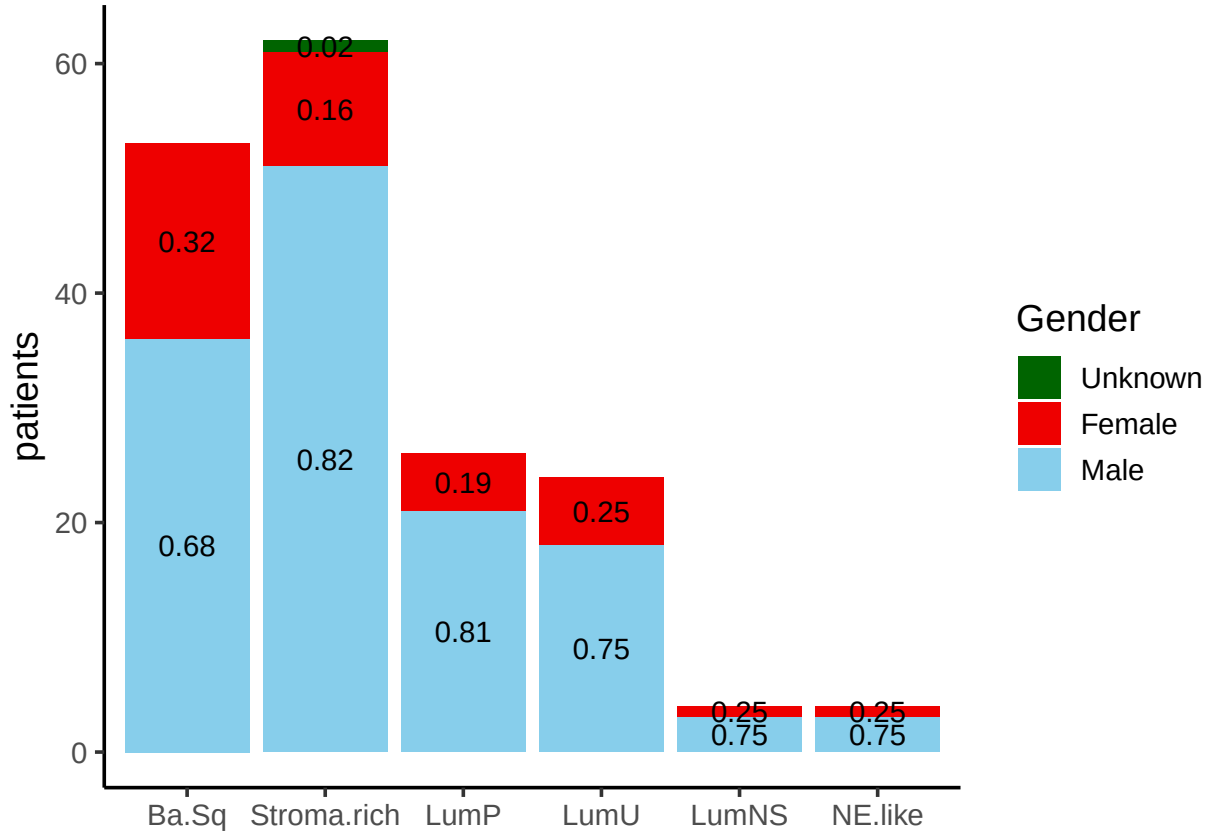


Figure 1(d) (Same results)

```
### Load the data

dataframe_1d = read.delim("IMvigor_ConsensusSubtypes.txt", header = TRUE,
    sep = "\t", stringsAsFactors = FALSE)

dataframe_1d$group = "IMvigor210"
```

```r
dataframe_1b$group = "UC_GENOME"
colnames(dataframe_1b)[2] = "Consensus"
colnames(dataframe_1d)[2] = "Consensus"

### Unify the names

dataframe_1d$Consensus = ifelse(dataframe_1d$Consensus == "Ba/Sq",
    "Ba.Sq", dataframe_1d$Consensus)
dataframe_1d$Consensus = ifelse(dataframe_1d$Consensus == "NE-like",
    "NE.like", dataframe_1d$Consensus)
dataframe_1d$Consensus = ifelse(dataframe_1d$Consensus == "Stroma-rich",
    "Stroma.rich", dataframe_1d$Consensus)


dataframe_1d = rbind(dataframe_1b[, c("Consensus", "group")],
    dataframe_1d[, c(2, 4)])

### Get the count number and proportion

df_plot_1d = dataframe_1d %>%
    count(Consensus, group) %>%
    group_by(group) %>%
    mutate(proportion = n/sum(n)) %>%
    ungroup()

### Reorder

df_plot_1d$Consensus = factor(df_plot_1d$Consensus, levels = c("Ba.Sq",
    "Stroma.rich", "LumP", "LumU", "LumNS", "NE.like"))

### Generate the stacked bar chart

subtype_colors = c(Ba.Sq = "red", Stroma.rich = "gold", LumP = "limegreen",
    LumU = "blue", LumNS = "darkgreen", NE.like = "purple")

ggplot(df_plot_1d, aes(x = group, y = proportion, fill = Consensus)) +
    geom_bar(stat = "identity") + geom_text(aes(label = n), position = position_stack(vjust = 0.5),
    size = 5) + scale_fill_manual(values = subtype_colors) +
    scale_y_continuous(breaks = seq(0, 1, by = 0.2), limits = c(0,
        1), expand = c(0, 0)) + ylab("proportion") + xlab(NULL) +
    theme_classic(base_size = 14)
```

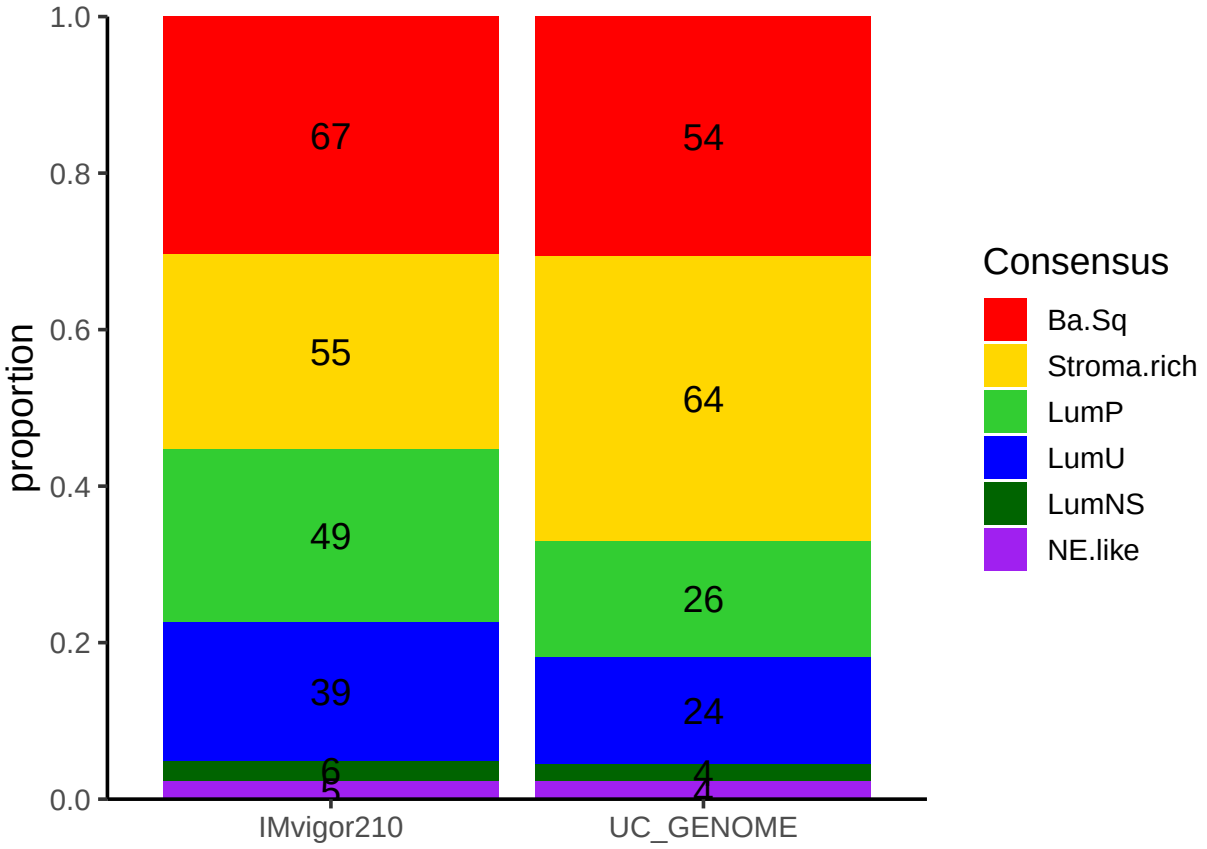


Figure 1(e) (Same results)

```
dataframe_1e = df_plot_1d
dataframe_1e$Consensus = ifelse(dataframe_1e$Consensus != "Stroma.rich",
    "Other", "Stroma.rich")

### Get the count numbers and proportion

df_plot_1e = dataframe_1e %>%
    group_by(Consensus, group) %>%
    summarise(n = sum(n), proportion = sum(proportion), .groups = "drop")

### Reorder

df_plot_1e$Consensus = factor(df_plot_1e$Consensus, levels = c("Other",
    "Stroma.rich"))


### Generate the stacked bar chart

subtype_colors = c(Other = "grey", Stroma.rich = "#F0E442")

ggplot(df_plot_1e, aes(x = group, y = proportion, fill = Consensus)) +
    geom_bar(stat = "identity") + geom_text(aes(label = n), position = position_stack(vjust = 0.5),
```

```
    size = 5) + scale_fill_manual(values = subtype_colors) +
    scale_y_continuous(breaks = seq(0, 1, by = 0.2), limits = c(0,
        1), expand = c(0, 0)) + ylab("proportion") + xlab(NULL) +
    theme_classic(base_size = 14)
```

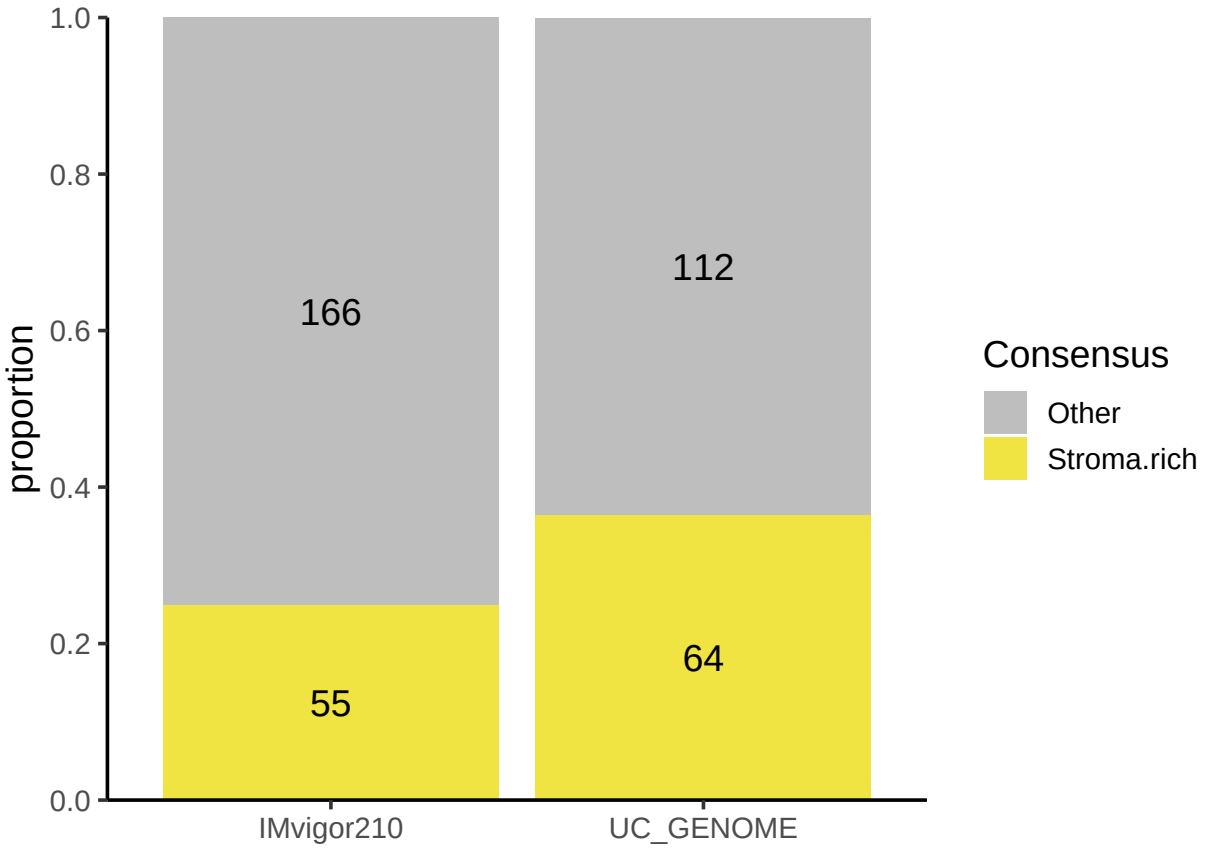


Figure 1(f) (Same results)

```
a = df_plot_1e %>%
    filter(Consensus == "Stroma.rich", group == "IMvigor210") %>%
    pull(n)
b = df_plot_1e %>%
    filter(Consensus == "Stroma.rich", group == "UC_GENOME") %>%
    pull(n)
c = df_plot_1e %>%
    filter(Consensus == "Other", group == "IMvigor210") %>%
    pull(n)
d = df_plot_1e %>%
    filter(Consensus == "Other", group == "UC_GENOME") %>%
    pull(n)

# Construct the 2x2 table for Chi-squared test
tbl = matrix(c(a, b, c, d), nrow = 2, byrow = TRUE, dimnames = list(Consensus = c("Stroma.rich",
    "Other"), Cohort = c("IMvigor210", "UC_GENOME")))
```

```r
p_value = chisq.test(tbl)$p.value

### Generate the figure

p_mat = matrix(c(1, p_value, p_value, 1), nrow = 2)
rownames(p_mat) = colnames(p_mat) = c("IMvigor210", "UC_GENOME")

df_plot_1f = reshape2::melt(p_mat)
colnames(df_plot_1f) = c("Var1", "Var2", "pvalue")
ggplot(df_plot_1f, aes(x = Var2, y = Var1, fill = pvalue)) +
    geom_tile(color = "white") + ylab(NULL) + xlab(NULL) + geom_text(aes(label = signif(pvalue,
    3), color = ifelse(pvalue == 1, "white", "black")), size = 6) +
    scale_color_identity() + scale_fill_gradientn(colors = c("yellow",
    "black"), values = scales::rescale(c(0, 0.25, 1)), limits = c(0,
    1), name = "p-value") + theme_minimal(base_size = 14) + theme(panel.grid = element_blank(),
    axis.text.x = element_text(angle = 45, hjust = 1)) + coord_fixed()
```

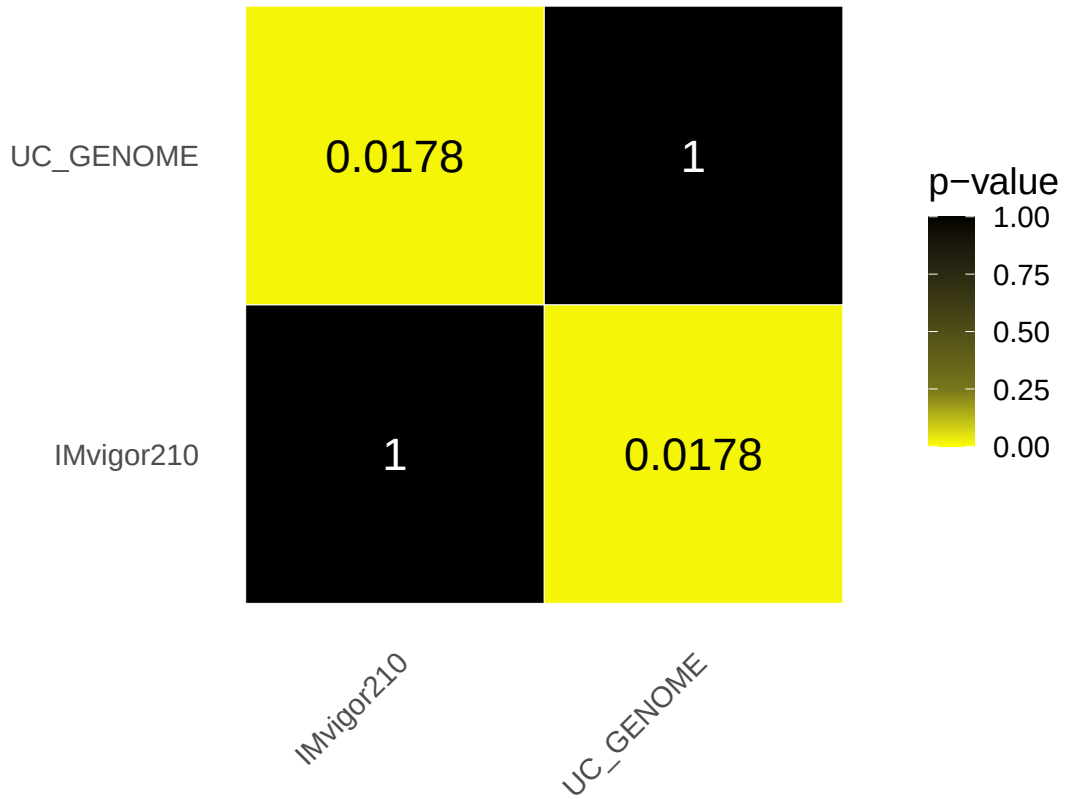


Figure 2(a) (Same results)

```r
### Load the file and transpose the dataframe
dataframe_2a = read.csv("UC_GENOME_Immune_Mut_Clin.csv")
dataframe_2a = t(dataframe_2a[, 2:13])
```

```r
### Change the order

custom_order = c("CDKN2A", "ERCC2", "BRCA2", "AKAP9", "SPEN",
    "LRP1B", "FGFR3", "RNF213", "KDM6A", "ARID1A", "KMT2D", "TP53")

dataframe_2a_ordered = dataframe_2a[custom_order, ]

### Calculate the frequency to display
freq = rowMeans(dataframe_2a_ordered) * 100
rownames(dataframe_2a_ordered) = paste0(rownames(dataframe_2a_ordered),
    " (", round(freq), "%)")

### Set the colors
col_fun = c(`0` = "gray80", `1` = "navy")


### Draw the Heatmap
Heatmap(dataframe_2a_ordered, name = "Mutation", col = col_fun,
    show_row_names = TRUE, show_column_names = FALSE, row_names_side = "left",
    row_title = "Genetic Alteration", cluster_rows = FALSE, cluster_columns = FALSE,
    show_heatmap_legend = FALSE, row_names_gp = gpar(fontsize = 12),
    column_title = "Figure 2(a)", column_title_gp = gpar(fontsize = 14,
        fontface = "bold"), rect_gp = gpar(col = "black", lwd = 0.5))
```

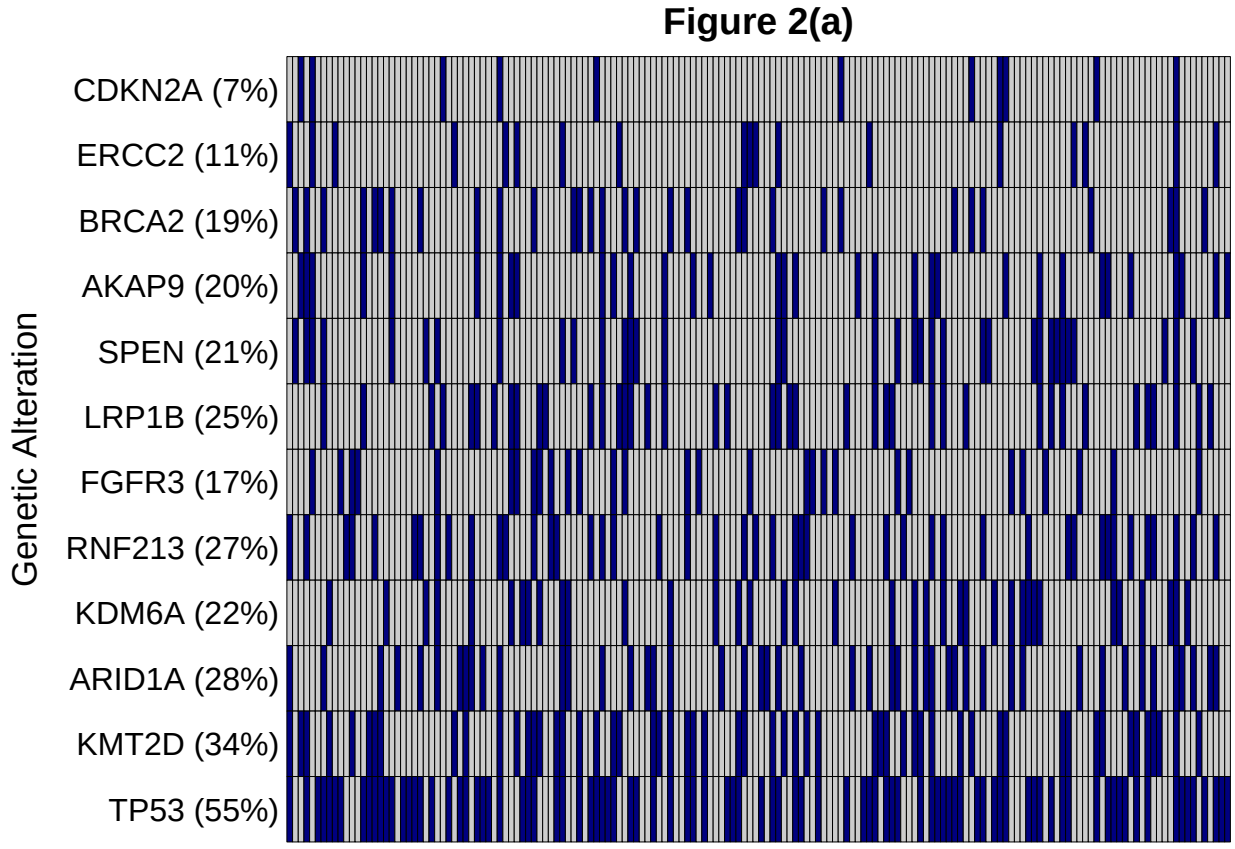


Figure 7(a) (Same results)

```
# Load Dataset
dataframe_7a = read.csv("IMvigor210.csv")

# Process dataset

dataframe_7a = dataframe_7a[, -c(1, 3, 5:6, 68:79)]
dataset = na.omit(dataframe_7a)
colnames(dataset)[ncol(dataset)] = "binaryResponse"
dataset$binaryResponse = as.factor(dataset$binaryResponse)

# Scale by z-score

for (i in c(3:64, 70)) {
    dataset[, i] = (scale(dataset[, i]) + 1)/2
}

# Define train and test sets (balanced), using the random
# seed identical to the original paper settings

set.seed(3456)
cpp3 = dataset
cpp3$binaryResponse = as.numeric(cpp3$binaryResponse)
cpp3 = cpp3[order(cpp3$binaryResponse), ]
cpp3$nrow = c(1:nrow(cpp3))
cpp3$group = (cpp3$nrow%%3 == 0)  #balanced partition
trainIndex = which(cpp3$group == FALSE)
cpp3$binaryResponse = as.factor(cpp3$binaryResponse == 2)
cpp3 = cpp3[, -which(colnames(cpp3) == "nrow")]
cpp3 = cpp3[, -which(colnames(cpp3) == "group")]

Train = cpp3[trainIndex, ]
Test = cpp3[-trainIndex, ]


### Train the glmnet, using cross-validation to choose
### hyperparameters(lambda and alpha)

control = trainControl(method = "repeatedcv", number = 50, repeats = 1)
set.seed(3456)
metric = "Accuracy"

elnet_train = train(binaryResponse ~ ., data = Train, method = "glmnet",
    metric = metric, tuneLength = 15, trControl = control)


### Get the best lambda and alpha

best_alpha = elnet_train$bestTune$alpha
best_lambda = elnet_train$bestTune$lambda
```

```r
### Process the dataframe which is used to generate plot
### later

coef_df = as.data.frame(as.matrix(coef(elnet_train$finalModel,
    elnet_train$bestTune$lambda)))
coef_df$Feature = rownames(coef_df)


customized_order = c("consensusClass.StromaRich", "BCell_60gene",
    "Bindea_DC", "Claudin", "Bindea_pDC", "Murray_M2", "Bindea_Eosinophils",
    "Bindea_iDC", "EMT_DOWN", "Iglesia_IGG_Cluster", "consensusClass.LumU",
    "Bindea_aDC", "Vincent_IPRES_Responder", "McDermott_T_eff",
    "consensusClass.LumP", "Tobacco.Use.HistoryPREVIOUS", "Ayers_IFNG",
    "Bindea_NK_CD56bright_cells", "Bindea_Tgd", "TMB.HighTRUE",
    "age", "Bindea_Th2_cells", "Martinez_Gordon_M1", "ECOG.0",
    "TMB.Numeric")

coef_df = coef_df[-1, ] %>%
    filter(Feature %in% customized_order) %>%
    mutate(Feature = factor(Feature, levels = rev(customized_order))) %>%
    rename(Coefficient = s1) %>%
    filter(!is.na(Coefficient)) %>%
    mutate(Response = ifelse(Coefficient > 0, "Better", "Worse"))


### Draw the plot

ggplot(coef_df, aes(x = Coefficient, y = Feature, fill = Response)) +
    geom_col(width = 0.8) + scale_fill_manual(values = c(Better = "red",
    Worse = "black")) + labs(x = "Mean Beta Coefficient", y = "Model Features",
    title = "Final Model Coefficients") + theme_minimal(base_size = 14) +
    theme(panel.grid = element_blank(), panel.border = element_rect(color = "black",
        fill = NA, linewidth = 0.8), axis.ticks = element_line(color = "black"),
        axis.ticks.length = unit(0.3, "lines"))
```

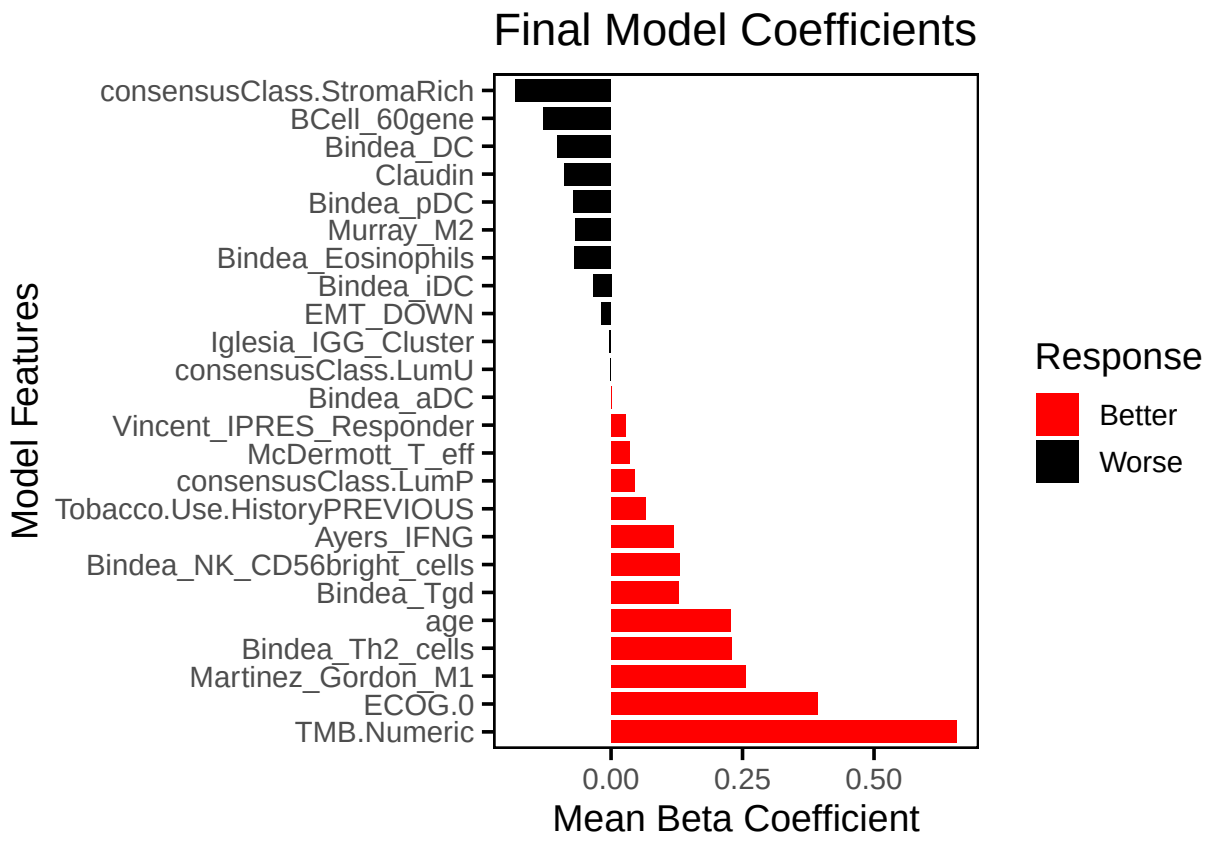


Figure 7(b_reproducibility) (Same results)

```
### Calculate the AUC for the model performance on
### IMvigor210

actual_IMvigor = as.numeric(Test$binaryResponse) - 1
predict_IMvigor = predict(elnet_train, type = "prob", newdata = Test)[,
    2]

roc_IMvigor = prediction(predict_IMvigor, actual_IMvigor)
performance_IMvigor = performance(roc_IMvigor, "tpr", "fpr")

pROC::auc(roc(actual_IMvigor, predict_IMvigor))
```

```
## Area under the curve: 0.834
```

```
### Calculate the AUC for the model performance on UNC_108

dataframe_UNC = read.csv("UNC_108.csv")
dataframe_UNC$Smoking.History = as.character(dataframe_UNC$Smoking.History)
dataframe_UNC$Smoking.History[dataframe_UNC$Smoking.History ==
    "Light"] = "Previous"
dataframe_UNC$Smoking.History = as.factor(dataframe_UNC$Smoking.History)
dataframe_UNC = dataframe_UNC[, -c(1, 64, 65, 68:71, 73:80)]
dataset_UNC = na.omit(dataframe_UNC)
```

```r
for (i in c(1:61, 63, 66:67)) {
    dataset_UNC[, i] = as.numeric(dataset_UNC[, i])
}
dataset_UNC$ECOG.0 = 1 * (dataset_UNC$ECOG == 0)
dataset_UNC$ECOG.2plus = 1 * (dataset_UNC$ECOG > 1)
dataset_UNC = dataset_UNC[, -(which(colnames(dataset_UNC) ==
    "ECOG"))]
dataset_UNC$TMB.high = dataset_UNC$TMB > 10

dataset_UNC = cbind(dataset_UNC[, -(which(colnames(dataset_UNC) ==
    "binaryResponse"))], dataset_UNC$binaryResponse)
colnames(dataset_UNC)[ncol(dataset_UNC)] = "binaryResponse"

for (i in c(62, 64, 67, 70)) {
    dataset_UNC[, i] = as.factor(dataset_UNC[, i])
}

### Scale

for (i in c(1:61, 63, 65)) {
    dataset_UNC[, i] = (scale(dataset_UNC[, i]) + 1)/2
}

# Calculate predicted values (UNC_108)
predict_UNC = (0.06575811 * (dataset_UNC$Smoking.History == "Previous") -
    0.1290173 * dataset_UNC$BCell_60gene - 0.003133251 * dataset_UNC$Iglesia_IGG_Cluster +
    0.0006185082 * dataset_UNC$Bindea_aDC - 0.1017214 * dataset_UNC$Bindea_DC -
    0.06946327 * dataset_UNC$Bindea_Eosinophils - 0.03426605 *
    dataset_UNC$Bindea_iDC + 0.130496 * dataset_UNC$Bindea_NK_CD56bright_cells -
    0.07166815 * dataset_UNC$Bindea_pDC - 0.08772033 * dataset_UNC$Claudin +
    0.1277694 * dataset_UNC$Bindea_Tgd + 0.2294665 * dataset_UNC$Bindea_Th2_cells +
    0.2564395 * dataset_UNC$Martinez_Gordon_M1 - 0.01843717 *
    dataset_UNC$EMT_DOWN - 0.06714543 * dataset_UNC$Murray_M2 +
    0.03606837 * dataset_UNC$McDermott_T_eff + 0.1195824 * dataset_UNC$Ayers_IFNG +
    0.02770162 * dataset_UNC$Vincent_IPRES_Responder + 0.6575133 *
    dataset_UNC$TMB + 0.1388407 * dataset_UNC$TMB.high + 0.3923339 *
    dataset_UNC$ECOG.0 + 0.2274032 * dataset_UNC$Age + 0.04406153 *
    (dataset_UNC$Consensus == "LumP") - 0.0007485804 * (dataset_UNC$Consensus ==
    "LumU") - 0.1816904 * (dataset_UNC$Consensus == "Stromarich"))

predict_UNC = as.numeric(predict_UNC)

actual_UNC = as.numeric(dataset_UNC$binaryResponse) - 1

roc_UNC = prediction(predict_UNC, actual_UNC)
performance_UNC = performance(roc_UNC, "tpr", "fpr")

pROC::auc(roc(actual_UNC, predict_UNC))
```

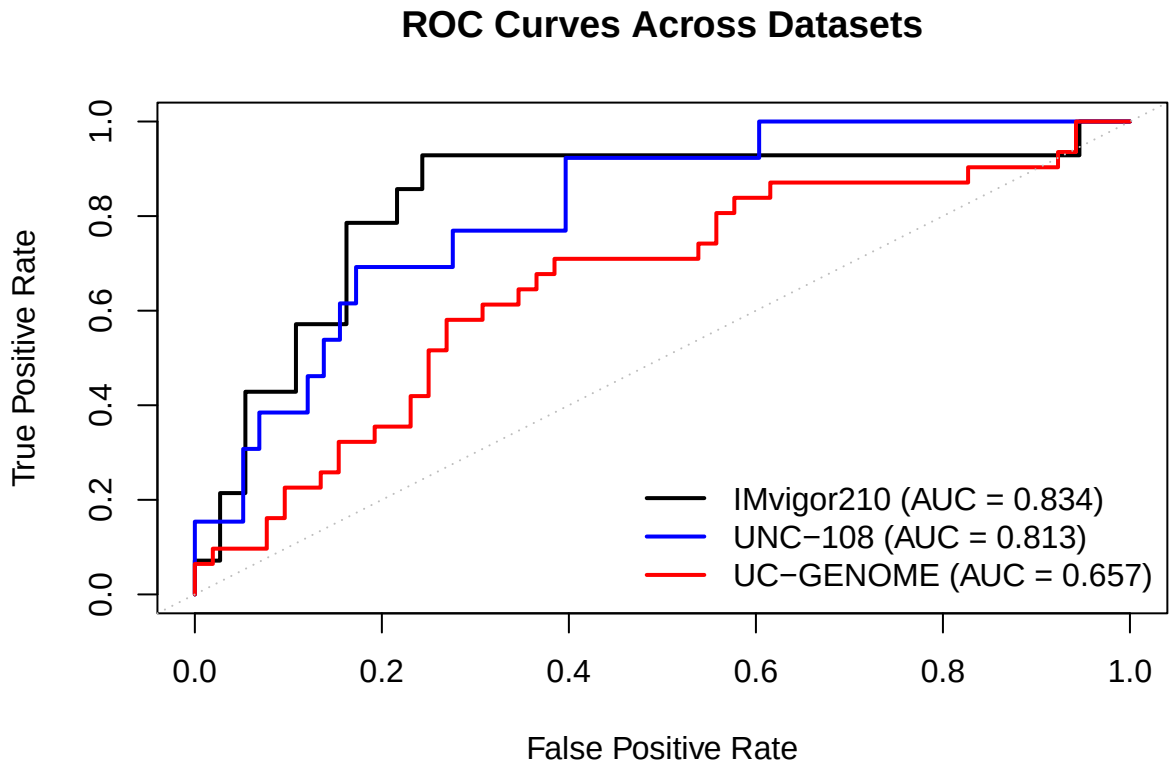
```
## Area under the curve: 0.813
```

```r
### Calculate the AUC for the model performance on
### UC-GENOME

dataframe_UCGENOME = read.csv("UC_GENOME_Immune_Mut_Clin.csv")
dataframe_UCGENOME = dataframe_UCGENOME[, -c(1:13, 76, 78:85,
    89, 90:95)]
dataset_UCGENOME = na.omit(dataframe_UCGENOME)
colnames(dataset_UCGENOME)[ncol(dataset_UCGENOME)] = "binaryResponse"
dataset_UCGENOME = dataset_UCGENOME * 1
dataset_UCGENOME$binaryResponse = as.factor(dataset_UCGENOME$binaryResponse)

### Scale

for (i in c(1:63, which(colnames(dataset_UCGENOME) == "age_ICB"))) {
    dataset_UCGENOME[, i] = (scale(dataset_UCGENOME[, i]) + 1)/2
}

# Calculate predicted values (UC_GENOME)
predict_UCGENOME = 0.06575811 * (1 - dataset_UCGENOME$SMOKING_STATUS.Current -
    dataset_UCGENOME$SMOKING_STATUS.Never) - 0.1290173 * dataset_UCGENOME$BCell_60gene -
    0.003133251 * dataset_UCGENOME$Iglesia_IGG_Cluster + 0.0006185082 *
    dataset_UCGENOME$Bindea_aDC - 0.1017214 * dataset_UCGENOME$Bindea_DC -
    0.06946327 * dataset_UCGENOME$Bindea_Eosinophils - 0.03426605 *
    dataset_UCGENOME$Bindea_iDC + 0.130496 * dataset_UCGENOME$Bindea_NK_CD56bright_cells -
    0.07166815 * dataset_UCGENOME$Bindea_pDC - 0.08772033 * dataset_UCGENOME$Claudin +
    0.1277694 * dataset_UCGENOME$Bindea_Tgd + 0.2294665 * dataset_UCGENOME$Bindea_Th2_cells +
    0.2564395 * dataset_UCGENOME$Martinez_Gordon_M1 - 0.01843717 *
    dataset_UCGENOME$EMT_DOWN - 0.06714543 * dataset_UCGENOME$Murray_M2 +
    0.03606837 * dataset_UCGENOME$McDermott_T_eff + 0.1195824 *
    dataset_UCGENOME$Ayers_IFNG + 0.02770162 * dataset_UCGENOME$Vincent_IPRES_Responder +
    0.6575133 * dataset_UCGENOME$TMB.Numeric + 0.1388407 * (dataset_UCGENOME$TMB.Numeric >
    10) + 0.3923339 * dataset_UCGENOME$ECOG.0 + 0.2274032 * dataset_UCGENOME$age_ICB +
    0.04406153 * (dataset_UCGENOME$Consensus.subtype.LumP) -
    0.0007485804 * (dataset_UCGENOME$Consensus.subtype.LumU) -
    0.1816904 * (dataset_UCGENOME$Consensus.subtype.StromaRich)

predict_UCGENOME = as.numeric(predict_UCGENOME)

actual_UCGENOME = as.numeric(dataset_UCGENOME$binaryResponse) -
    1

roc_UCGENOME = prediction(predict_UCGENOME, actual_UCGENOME)
performance_UCGENOME = performance(roc_UCGENOME, "tpr", "fpr")

pROC::auc(roc(actual_UCGENOME, predict_UCGENOME))
```

```
## Area under the curve: 0.6569
```

```r
### Plot ROC

plot(performance_IMvigor, col = "black", lwd = 2, lty = 1, xlim = c(0,
    1), ylim = c(0, 1), main = "ROC Curves Across Datasets",
    xlab = "False Positive Rate", ylab = "True Positive Rate")
```

```r
plot(performance_UNC, col = "blue", lwd = 2, lty = 1, add = TRUE)


plot(performance_UCGENOME, col = "red", lwd = 2, lty = 1, add = TRUE)

# Reference line

abline(0, 1, col = "gray", lty = 3)

# Add legend

legend("bottomright", legend = c(paste0("IMvigor210 (AUC = ",
    round(pROC::auc(roc(actual_IMvigor, predict_IMvigor)), 3),
    ")"), paste0("UNC-108 (AUC = ", round(pROC::auc(roc(actual_UNC,
    predict_UNC)), 3), ")"), paste0("UC-GENOME (AUC = ", round(pROC::auc(roc(actual_UCGENOME,
    predict_UCGENOME)), 3), ")")), col = c("black", "blue", "red"),
    lty = c(1, 1, 1), lwd = 2, bty = "n")
```



Train the first new prediction method (SVM with radial basis function kernel), select the model performs the best across 5 trained models

```r
### Same as the original paper, using IMvigor210 as the
### training set

dataset_7b_Imvigor = dataset
dataset_7b_Imvigor = model.matrix(binaryResponse ~ ., dataset_7b_Imvigor)[,
    -1]
common_vars = intersect(rownames(coef_df), colnames(dataset_7b_Imvigor))
dataset_7b_Imvigor = cbind(dataset_7b_Imvigor[, common_vars],
    dataset[, 75])
colnames(dataset_7b_Imvigor)[25] = "binaryResponse"
dataset_7b_Imvigor = as.data.frame(dataset_7b_Imvigor)
dataset_7b_Imvigor$binaryResponse = as.factor(dataset_7b_Imvigor$binaryResponse -
    1)


### Create 5 folds
set.seed(1314)
outer_folds_svm = createFolds(dataset_7b_Imvigor$binaryResponse,
    k = 5, returnTrain = T)

auc_outer = c()
model_svm = list()
prediction_svm = list()

for (i in seq_along(outer_folds_svm)) {

    ### Outer layer (4 folds for training, 1 folds for
    ### testing)
    cat("==> Outer Fold", i, "\n")
    train_outer = dataset_7b_Imvigor[outer_folds_svm[[i]], ]
    test_outer = dataset_7b_Imvigor[-outer_folds_svm[[i]], ]

    ### Inner layer (For the 4 training folds, do the
    ### 5-fold CV again, split the data into 4 folds for
    ### training and 1 folds for tuning hyperparameters)
    ctrl_inner = trainControl(method = "cv", number = 5, classProbs = TRUE,
        summaryFunction = twoClassSummary)

    ### Make the correct outcome type (factor)
    train_outer$binaryResponse = factor(train_outer$binaryResponse,
        levels = c(0, 1), labels = c("no", "yes"))
    test_outer$binaryResponse = factor(test_outer$binaryResponse,
        levels = c(0, 1), labels = c("no", "yes"))

    ### 500 combinations of hyperparameters (Regularization
    ### parameter C, Kernel width parameter Gamma)
    svm_tuned = train(binaryResponse ~ ., data = train_outer,
        method = "svmRadial", trControl = ctrl_inner, tuneLength = 500,
        metric = "ROC")

    model_svm[[i]] = svm_tuned

    # Choose the model with the best hyperparameter
```

```
    # combinations to perform predictions
    prob = predict(svm_tuned, newdata = test_outer, type = "prob")[,
        "yes"]
    prediction = prediction(prob, test_outer[, ncol(test_outer)])
    prediction_svm[[i]] = prediction
    actual = ifelse(test_outer$binaryResponse == "yes", 1, 0)

    # Calculate AUC
    auc_val = pROC::auc(roc(actual, prob))
    auc_outer[i] = auc_val

    cat("Fold", i, "AUC =", round(auc_val, 3), "\n")
}
```

```
## ==> Outer Fold 1

## Fold 1 AUC = 0.707
## ==> Outer Fold 2

## Fold 2 AUC = 0.696
## ==> Outer Fold 3

## Fold 3 AUC = 0.692
## ==> Outer Fold 4

## Fold 4 AUC = 0.523
## ==> Outer Fold 5

## Fold 5 AUC = 0.761
```

Train the second new prediction method (XGBoost), select the model performs the best across 5 trained models

```
### Same as the original paper, using IMvigor210 as the
### training set

dataset_7b_Imvigor$binaryResponse = as.numeric(dataset_7b_Imvigor$binaryResponse) -
    1

### Create 5 folds

set.seed(1314)
outer_folds_xgb = createFolds(dataset_7b_Imvigor$binaryResponse,
    k = 5, returnTrain = T)

auc_outer = c()
roc_list = list()
model_xgb = list()
prediction_xgb = list()
```

```
for (i in seq_along(outer_folds_xgb)) {
    cat("==> Outer Fold", i, "\n")

    ### Outer layer (4 folds for training, 1 folds for
    ### testing)

    train_outer = dataset_7b_Imvigor[outer_folds_xgb[[i]], ]
    test_outer = dataset_7b_Imvigor[-outer_folds_xgb[[i]], ]

    y_train = train_outer$binaryResponse
    y_test = test_outer$binaryResponse

    X_train = model.matrix(binaryResponse ~ ., data = train_outer)[,
        -1]
    X_test = model.matrix(binaryResponse ~ ., data = test_outer)[,
        -1]

    dtrain = xgb.DMatrix(data = X_train, label = y_train)
    dtest = xgb.DMatrix(data = X_test)

    best_auc = -Inf
    best_param = list()

    ### Grid search for eta(Learning rate) and max_depth

    for (eta in c(0.001, 0.005, 0.01, 0.02, 0.03, 0.04, 0.05,
        0.1, 0.15, 0.2, 0.25, 0.3)) {
        for (depth in c(2:10)) {
            param = list(objective = "binary:logistic", eval_metric = "auc",
                eta = eta, max_depth = depth, subsample = 0.8,
                colsample_bytree = 0.8, lambda = 1)

            ### Inner layer (For the 4 training folds, do
            ### the 5-fold CV again, split the data into 4
            ### folds for training and 1 folds for tuning
            ### hyperparameters)

            cv = xgb.cv(params = param, data = dtrain, nfold = 5,
                nrounds = 100, early_stopping_rounds = 10, verbose = 0,
                maximize = TRUE, stratified = TRUE)

            auc_cv = max(cv$evaluation_log$test_auc_mean)
            if (auc_cv > best_auc) {
                best_auc = auc_cv
                best_param = param
                best_nround = cv$best_iteration
            }
        }
    }

    ### Choose the model with the best hyperparameter
    ### combinations to perform predictions
    model = xgb.train(params = best_param, data = dtrain, nrounds = best_nround,
```

```
        verbose = 0)

    model_xgb[[i]] = model

    pred = predict(model, newdata = dtest)
    prediction = prediction(pred, y_test)
    roc_obj = roc(y_test, pred)
    roc_list[[i]] = roc_obj
    prediction_xgb[[i]] = prediction
    auc_val = pROC::auc(roc_obj)
    auc_outer[i] = auc_val

    cat("  AUC =", round(auc_val, 3), "\n")
}
```

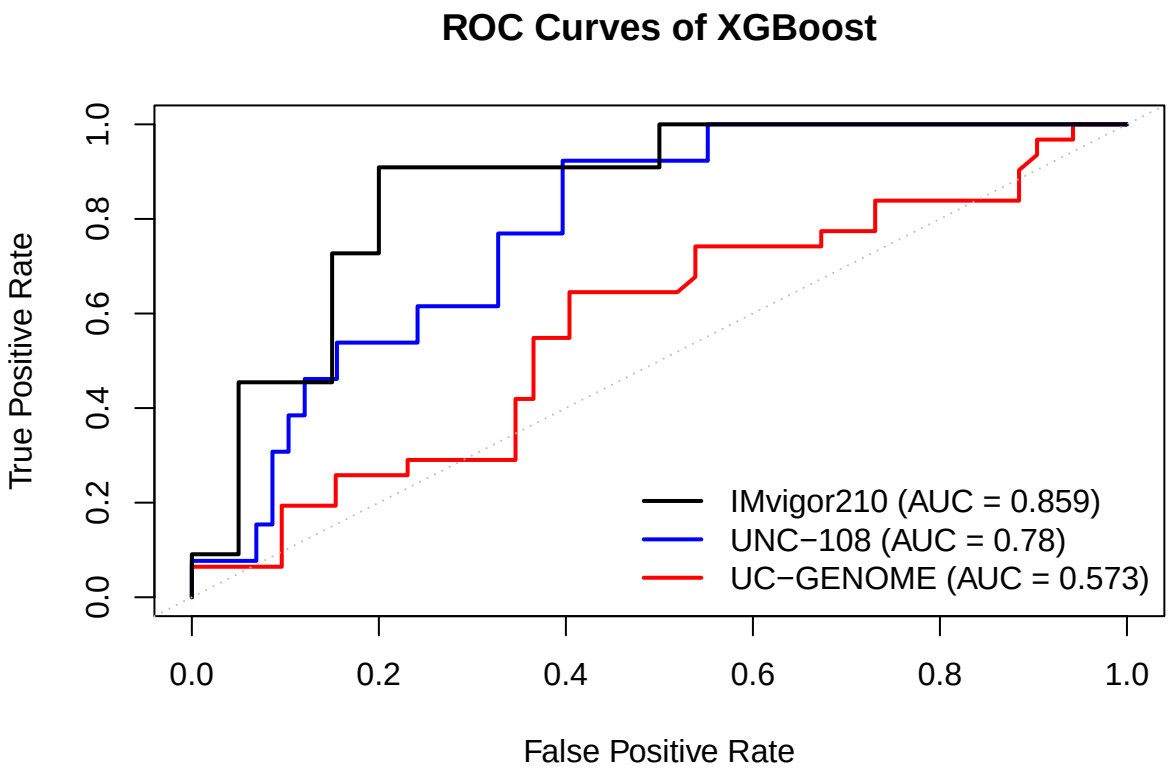
```
## ==> Outer Fold 1

##   AUC = 0.804
## ==> Outer Fold 2

##   AUC = 0.548
## ==> Outer Fold 3

##   AUC = 0.859
## ==> Outer Fold 4

##   AUC = 0.738
## ==> Outer Fold 5

##   AUC = 0.833
```

XGBoost Performance testing (Choosing the third trained model)

```
### Test the XGBoost performance on UNC_108

### Unify the variable names (The test_new_UNC can also be
### used for testing of the SVM on UNC_108)
test_new_UNC = model.matrix(binaryResponse ~ ., dataset_UNC)[,
    -1]

colnames(test_new_UNC)[62] = "consensusClass.LumP"
colnames(test_new_UNC)[63] = "consensusClass.LumU"
colnames(test_new_UNC)[65] = "consensusClass.StromaRich"
colnames(test_new_UNC)[66] = "age"
colnames(test_new_UNC)[69] = "Tobacco.Use.HistoryPREVIOUS"
colnames(test_new_UNC)[70] = "TMB.Numeric"


common_vars_UNC = intersect(rownames(coef_df), colnames(test_new_UNC))
```

```
test_new_UNC = cbind(test_new_UNC[, common_vars], dataset_UNC$binaryResponse)
colnames(test_new_UNC)[25] = "binaryResponse"
test_new_UNC[, ncol(test_new_UNC)] = test_new_UNC[, ncol(test_new_UNC)] -
    1
test_new_UNC = test_new_UNC[, colnames(dataset_7b_Imvigor)]

### Prepare processed data as XGBoost model input

test_new_UNC_features = xgb.DMatrix(data = test_new_UNC[, -ncol(test_new_UNC)])
test_new_UNC_labels = as.numeric(test_new_UNC[, ncol(test_new_UNC)])

### AUC of XGBoost on UNC_108

pred_UNC = predict(model_xgb[3], newdata = test_new_UNC_features)[[1]]
roc_obj_UNC = roc(test_new_UNC_labels, pred_UNC)
auc_val_UNC = pROC::auc(roc_obj_UNC)


### Test the XGBoost performance on UC_GENOME

### Unify the variable names (The test_new_UCGENOME can
### also be used for testing of the SVM on UC_GENOME)
test_new_UCGENOME = as.data.frame(model.matrix(binaryResponse ~
    ., dataset_UCGENOME)[, -1])
test_new_UCGENOME$Tobacco.Use.HistoryPREVIOUS = ifelse(test_new_UCGENOME$SMOKING_STATUS.Current ==
    0 & test_new_UCGENOME$SMOKING_STATUS.Never == 0, 1, 0)
test_new_UCGENOME = as.matrix(test_new_UCGENOME)

colnames(test_new_UCGENOME)[68] = "consensusClass.LumP"
colnames(test_new_UCGENOME)[69] = "consensusClass.LumU"
colnames(test_new_UCGENOME)[70] = "consensusClass.StromaRich"
colnames(test_new_UCGENOME)[72] = "age"

common_vars_UCGENOME = intersect(rownames(coef_df), colnames(test_new_UCGENOME))

test_new_UCGENOME = cbind(test_new_UCGENOME[, common_vars], dataset_UCGENOME$binaryResponse)
colnames(test_new_UCGENOME)[25] = "binaryResponse"
test_new_UCGENOME[, ncol(test_new_UCGENOME)] = test_new_UCGENOME[,
    ncol(test_new_UCGENOME)] - 1
test_new_UCGENOME = test_new_UCGENOME[, colnames(dataset_7b_Imvigor)]

### Prepare processed data as XGBoost model input

test_new_UCGENOME_features = xgb.DMatrix(data = test_new_UCGENOME[,
    -ncol(test_new_UCGENOME)])
test_new_UCGENOME_labels = as.numeric(test_new_UCGENOME[, ncol(test_new_UCGENOME)])

### AUC of XGBoost on UC_GENOME

pred_UCGENOME = predict(model_xgb[3], newdata = test_new_UCGENOME_features)[[1]]
roc_obj_UCGENOME = roc(test_new_UCGENOME_labels, pred_UCGENOME)
auc_val_UCGENOME = pROC::auc(roc_obj_UCGENOME)
```

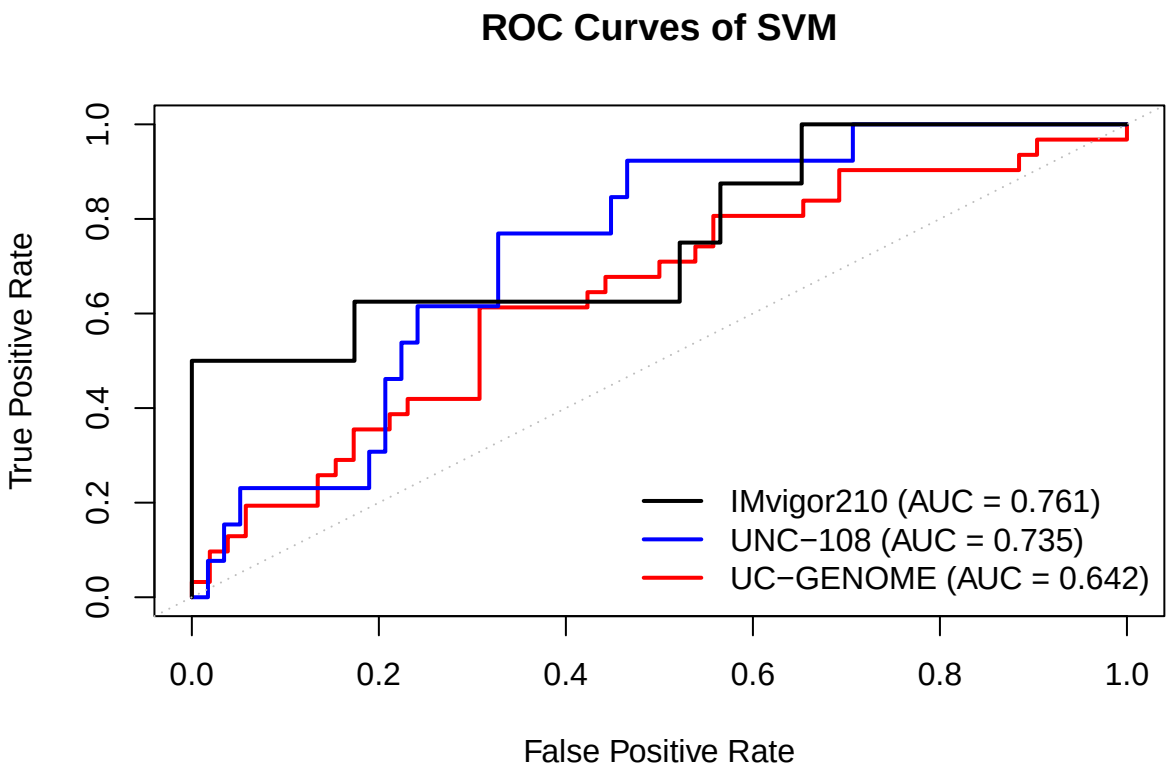
```r
### Draw the ROCs for the XGBoost

roc_UNC = prediction(pred_UNC, test_new_UNC_labels)
performance_UNC = performance(roc_UNC, "tpr", "fpr")

roc_UCGENOME = prediction(pred_UCGENOME, test_new_UCGENOME_labels)
performance_UCGENOME = performance(roc_UCGENOME, "tpr", "fpr")

roc_IMvigor = prediction_xgb[[3]]
performance_IMvigor = performance(roc_IMvigor, "tpr", "fpr")
actual_IMvigor_xgb = dataset_7b_Imvigor[-outer_folds_xgb[[3]],
    ncol(dataset_7b_Imvigor)]
pred_IMvigor_xgb = unlist(roc_IMvigor@predictions)


plot(performance_UCGENOME, col = "red", lwd = 2, lty = 1, xlim = c(0,
    1), ylim = c(0, 1), main = "ROC Curves of XGBoost", xlab = "False Positive Rate",
    ylab = "True Positive Rate")


plot(performance_UNC, col = "blue", lwd = 2, lty = 1, add = TRUE)


plot(performance_IMvigor, col = "black", lwd = 2, lty = 1, add = TRUE)

# Reference line
abline(0, 1, col = "gray", lty = 3)

# Add legend
legend("bottomright", legend = c(paste0("IMvigor210 (AUC = ",
    round(pROC::auc(roc(actual_IMvigor_xgb, pred_IMvigor_xgb)),
        3), ")"), paste0("UNC-108 (AUC = ", round(pROC::auc(roc(actual_UNC,
    pred_UNC)), 3), ")"), paste0("UC-GENOME (AUC = ", round(pROC::auc(roc(actual_UCGENOME,
    pred_UCGENOME)), 3), ")")), col = c("black", "blue", "red"),
    lty = c(1, 1, 1), lwd = 2, bty = "n")
```



SVM Performance testing (Choosing the fifth trained model)

```
### Test the SVM performance on UNC_108

pred_UNC_svm = predict(model_svm[[5]], newdata = test_new_UNC[,
    -ncol(test_new_UNC)], type = "prob")[, "yes"]
roc_obj_UNC_svm = roc(test_new_UNC[, ncol(test_new_UNC)], pred_UNC_svm)
auc_val_UNC_svm = pROC::auc(roc_obj_UNC_svm)

roc_UNC_svm = prediction(pred_UNC_svm, test_new_UNC_labels)
performance_UNC_svm = performance(roc_UNC_svm, "tpr", "fpr")

### Test the SVM performance on UC_GENOME

pred_UCGENOME_svm = predict(model_svm[[5]], newdata = test_new_UCGENOME[,
    -ncol(test_new_UCGENOME)], type = "prob")[, "yes"]
roc_obj_UCGENOME_svm = roc(test_new_UCGENOME[, ncol(test_new_UCGENOME)],
    pred_UCGENOME_svm)
auc_val_UCGENOME_svm = pROC::auc(roc_obj_UCGENOME_svm)

roc_UCGENOME_svm = prediction(pred_UCGENOME_svm, test_new_UCGENOME_labels)
performance_UCGENOME_svm = performance(roc_UCGENOME_svm, "tpr",
    "fpr")
```

```
### Test the SVM performance on IMvigor210

actual_IMvigor_svm = dataset_7b_Imvigor[-outer_folds_svm[[5]],
    ncol(dataset_7b_Imvigor)]
pred_IMvigor_svm = unlist(prediction_svm[[5]]@predictions)
roc_IMvigor_svm = prediction(pred_IMvigor_svm, actual_IMvigor_svm)
performance_IMvigor_svm = performance(roc_IMvigor_svm, "tpr",
    "fpr")

### Draw the ROCs for the SVM

plot(performance_UCGENOME_svm, col = "red", lwd = 2, lty = 1,
    xlim = c(0, 1), ylim = c(0, 1), main = "ROC Curves of SVM",
    xlab = "False Positive Rate", ylab = "True Positive Rate")

plot(performance_UNC_svm, col = "blue", lwd = 2, lty = 1, add = TRUE)

plot(performance_IMvigor_svm, col = "black", lwd = 2, lty = 1,
    add = TRUE)

# Reference line
abline(0, 1, col = "gray", lty = 3)

# Add legend
legend("bottomright", legend = c(paste0("IMvigor210 (AUC = ",
    round(pROC::auc(roc(actual_IMvigor_svm, pred_IMvigor_svm)),
        3), ")"), paste0("UNC-108 (AUC = ", round(pROC::auc(roc(actual_UNC,
    pred_UNC_svm)), 3), ")"), paste0("UC-GENOME (AUC = ", round(pROC::auc(roc(actual_UCGENOME,
    pred_UCGENOME_svm)), 3), ")")), col = c("black", "blue",
    "red"), lty = c(1, 1, 1), lwd = 2, bty = "n")
```



Calculate the Brier Score and F1-Score

```
### The Brier Score and F1-Score of SVM on UNC_108

predict_class_svm_UNC = factor(ifelse(pred_UNC_svm > 0.5, 1,
    0), levels = c(0, 1))
actual_class_svm_UNC = factor(actual_UNC, levels = c(0, 1))
cm_svm_UNC = confusionMatrix(data = predict_class_svm_UNC, reference = actual_class_svm_UNC)

precision = cm_svm_UNC$byClass["Pos Pred Value"]
recall = cm_svm_UNC$byClass["Sensitivity"]
f1_svm_UNC = 2 * precision * recall/(precision + recall)

brier_score_svm_unc = mean((pred_UNC_svm - actual_UNC)^2)


### The Brier Score and F1-Score of SVM on UC_GENOME

predict_class_svm_UCGENOME = factor(ifelse(pred_UCGENOME_svm >
    0.5, 1, 0), levels = c(0, 1))
actual_class_svm_UCGENOME = factor(actual_UCGENOME, levels = c(0,
    1))
cm_svm_UCGENOME = confusionMatrix(data = predict_class_svm_UCGENOME,
    reference = actual_class_svm_UCGENOME)
```

```
precision = cm_svm_UCGENOME$byClass["Pos Pred Value"]
recall = cm_svm_UCGENOME$byClass["Sensitivity"]
f1_svm_UCGENOME = 2 * precision * recall/(precision + recall)

brier_score_svm_ucgenome = mean((pred_UCGENOME_svm - actual_UCGENOME)^2)

### The Brier Score and F1-Score of SVM on IMvigor210

predict_class_svm_IMvigor = factor(ifelse(pred_IMvigor_svm >
    0.5, 1, 0), levels = c(0, 1))
actual_class_svm_IMvigor = factor(actual_IMvigor_svm, levels = c(0,
    1))
cm_svm_IMvigor = confusionMatrix(data = predict_class_svm_IMvigor,
    reference = actual_class_svm_IMvigor)

precision = cm_svm_IMvigor$byClass["Pos Pred Value"]
recall = cm_svm_IMvigor$byClass["Sensitivity"]
f1_svm_IMvigor = 2 * precision * recall/(precision + recall)

brier_score_imvigor = mean((pred_IMvigor_svm - actual_IMvigor_svm)^2)

### The Brier Score and F1-Score of the original glmnet on
### UC_GENOME

predict_class_glmnet_UCGENOME = factor(ifelse(plogis(predict_UCGENOME) >
    0.5, 1, 0), levels = c(0, 1))
actual_class_glmnet_UCGENOME = factor(actual_UCGENOME, levels = c(0,
    1))
cm_glmnet_UCGENOME = confusionMatrix(data = predict_class_glmnet_UCGENOME,
    reference = actual_class_glmnet_UCGENOME)

precision = cm_glmnet_UCGENOME$byClass["Pos Pred Value"]
recall = cm_glmnet_UCGENOME$byClass["Sensitivity"]
f1_glmnet_UCGENOME = 2 * precision * recall/(precision + recall)

brier_score_glmnet_ucgenome = mean((plogis(predict_UCGENOME) -
    actual_UCGENOME)^2)

### The Brier Score and F1-Score of the original glmnet on
### UNC_108

predict_class_glmnet_UNC = factor(ifelse(plogis(predict_UNC) >
    0.5, 1, 0), levels = c(0, 1))
actual_class_glmnet_UNC = factor(actual_UNC, levels = c(0, 1))
cm_glmnet_UNC = confusionMatrix(data = predict_class_glmnet_UNC,
    reference = actual_class_glmnet_UNC)

precision = cm_glmnet_UNC$byClass["Pos Pred Value"]
recall = cm_glmnet_UNC$byClass["Sensitivity"]
f1_glmnet_UNC = 2 * precision * recall/(precision + recall)

brier_score_glmnet_unc = mean((actual_UNC - plogis(predict_UNC))^2)
```

```r
### The Brier Score and F1-Score of the original glmnet on
### IMvigor210

predict_class_glmnet_IMvigor = factor(ifelse(predict_IMvigor >
    0.5, 1, 0), levels = c(0, 1))
actual_class_glmnet_IMvigor = factor(actual_IMvigor, levels = c(0,
    1))
cm_glmnet_IMvigor = confusionMatrix(data = predict_class_glmnet_IMvigor,
    reference = actual_class_glmnet_IMvigor)

precision = cm_glmnet_IMvigor$byClass["Pos Pred Value"]
recall = cm_glmnet_IMvigor$byClass["Sensitivity"]
f1_glmnet_IMvigor = 2 * precision * recall/(precision + recall)

brier_score_glmnet_imvigor = mean((actual_IMvigor - predict_IMvigor)^2)

### The Brier Score and F1-Score of the XGboost on
### UC_GENOME

predict_class_xgb_UCGENOME = factor(ifelse(pred_UCGENOME > 0.5,
    1, 0), levels = c(0, 1))
actual_class_xgb_UCGENOME = factor(actual_UCGENOME, levels = c(0,
    1))
cm_xgb_UCGENOME = confusionMatrix(data = predict_class_xgb_UCGENOME,
    reference = actual_class_xgb_UCGENOME)

precision = cm_xgb_UCGENOME$byClass["Pos Pred Value"]
recall = cm_xgb_UCGENOME$byClass["Sensitivity"]
f1_xgb_UCGENOME = 2 * precision * recall/(precision + recall)

brier_score_xgb_ucgenome = mean((pred_UCGENOME - actual_UCGENOME)^2)

### The Brier Score and F1-Score of the XGBoost on UNC_108

predict_class_xgb_UNC = factor(ifelse(pred_UNC > 0.5, 1, 0),
    levels = c(0, 1))
actual_class_xgb_UNC = factor(actual_UNC, levels = c(0, 1))
cm_xgb_UNC = confusionMatrix(data = predict_class_xgb_UNC, reference = actual_class_xgb_UNC)

precision = cm_xgb_UNC$byClass["Pos Pred Value"]
recall = cm_xgb_UNC$byClass["Sensitivity"]
f1_xgb_UNC = 2 * precision * recall/(precision + recall)

brier_score_xgb_unc = mean((pred_UNC - actual_UNC)^2)

### The Brier Score and F1-Score of the XGBoost on
### IMvigor210

predict_class_xgb_IMvigor = factor(ifelse(pred_IMvigor_xgb >
    0.5, 1, 0), levels = c(0, 1))
actual_class_xgb_IMvigor = factor(actual_IMvigor_xgb, levels = c(0,
    1))
cm_xgb_IMvigor = confusionMatrix(data = predict_class_glmnet_IMvigor,
```

```
    reference = actual_class_glmnet_IMvigor)

precision = cm_xgb_IMvigor$byClass["Pos Pred Value"]
recall = cm_xgb_IMvigor$byClass["Sensitivity"]
f1_xgb_IMvigor = 2 * precision * recall/(precision + recall)

brier_score_xgb_imvigor = mean((pred_IMvigor_xgb - actual_IMvigor_xgb)^2)
```

Generate a table to compare different metrics across three models

```
xgb_results = data.frame(Dataset = c("IMvigor210", "UNC_108",
    "UC_GENOME"), F1_Score = c(f1_xgb_IMvigor, f1_xgb_UNC, f1_xgb_UCGENOME),
    Brier_Score = c(brier_score_xgb_imvigor, brier_score_xgb_unc,
        brier_score_xgb_ucgenome))

svm_results = data.frame(Dataset = c("IMvigor210", "UNC_108",
    "UC_GENOME"), F1_Score = c(f1_svm_IMvigor, f1_svm_UNC, f1_svm_UCGENOME),
    Brier_Score = c(brier_score_imvigor, brier_score_svm_unc,
        brier_score_svm_ucgenome))

glmnet_results = data.frame(Dataset = c("IMvigor210", "UNC_108",
    "UC_GENOME"), F1_Score = c(f1_glmnet_IMvigor, f1_glmnet_UNC,
    f1_glmnet_UCGENOME), Brier_Score = c(brier_score_glmnet_imvigor,
    brier_score_glmnet_unc, brier_score_glmnet_ucgenome))


svm_results$Model = "SVM"
xgb_results$Model = "XGBoost"
glmnet_results$Model = "GLMNET"

combined_results <- rbind(svm_results, xgb_results, glmnet_results)


kable(combined_results[, c("Model", "Dataset", "F1_Score", "Brier_Score")],
    caption = "Comparison of F1 Score and Brier Score across Models and Datasets")
```

Table 2: Comparison of F1 Score and Brier Score across Models and Datasets

Model	Dataset	F1_Score	Brier_Score
SVM	IMvigor210	0.8518519	0.1612847
SVM	UNC_108	0.8760331	0.1492293
SVM	UC_GENOME	0.7692308	0.2328382
XGBoost	IMvigor210	0.8372093	0.1937512
XGBoost	UNC_108	0.8833333	0.1360661
XGBoost	UC_GENOME	0.7441860	0.2539436
GLMNET	IMvigor210	0.8372093	0.1640708
GLMNET	UNC_108	0.2686567	0.3558003
GLMNET	UC_GENOME	0.1379310	0.3073718