Usig XIST, X-linked, and Y-linked genes for cell line sex phenotype and complement predictions

Ali Termos

11/24/23

Options for Printing Report

```
# this will make sure that the code doesn't run off the page when printing a report
knitr::opts_chunk$set(tidy.opts = list(width.cutoff = 50), tidy = TRUE)
```

Loading Libraries, Importing Data sets, and Selecting Genes

Loading used libraries

```
# Clear global environment
rm(list = ls())

# check if packages are installed, if not,
# install packages
if (!require(ggplot2)) {
    install.packages("ggplot2")
    library(ggplot2)
}
```

```
## Loading required package: ggplot2
```

```
if (!require(dplyr)) {
    install.packages("dplyr")
    library(dplyr)
}
```

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

```
if (!require(tidyr)) {
    install.packages("tidyr")
    library(tidyr)
}
```

```
## Loading required package: tidyr
```

Importing data sets

```
# Free up garbage collection (clean RAM) and
# clear objects in global environment
gc()
```

```
##           used (Mb) gc trigger (Mb) max used (Mb)
## Ncells  781005 41.8    1394135 74.5  1394135 74.5
## Vcells 1388905 10.6    8388608 64.0  2291696 17.5
```

```
rm(list = ls())

# Define directories
working_dir <- "C:/Users/aliad/OneDrive/Documents/R_PROJECTS/BIO-598_Final-Project_R/"
data_dir <- "C:/Users/aliad/OneDrive/Documents/R_PROJECTS/BIO-598_Final-Project_R/data/"
results_dir <- "C:/Users/aliad/OneDrive/Documents/R_PROJECTS/BIO-598_Final-Project_R/results/"

# Set current working directory
setwd(working_dir)

# Read the .gct file, skipping the first two
# lines of metadata
gene_counts_raw_data <- read.delim(paste0(data_dir,
    "CCLE_RNAseq_genes_counts_20180929.gct"), header = TRUE,
    sep = "\t", skip = 2, check.names = FALSE)

# Read in the annotation data
annotation_raw_data <- read.delim(paste0(data_dir,
    "Cell_lines_annotations_20181226.txt"), header = TRUE,
    sep = "\t")
```

Selecting genes under study

```
# Sellect genes that are included in the study

# Define and store XIST gene
XIST <- c("XIST")
```

```r
# Define an store the chosen list of X-linked
# genes
X_linked_genes <- c("XIST", "AMELX", "DDX3X", "EIF1AX",
    "KDM5C", "NLGN4X", "RPS4X", "TBL1X", "TMSB4X",
    "USP9X", "KDM6A", "ZFX")

# Define and store the chosen list of Y-linlked
# genes
Y_linked_genes <- c("AMELY", "DDX3Y", "EIF1AY", "KDM5D",
    "NLGN4Y", "RPS4Y1", "TBL1Y", "TMSB4Y", "USP9Y",
    "UTY", "ZFY")
```

Pre-processing

Pre-processing gene counts data set

```r
# Pre-process gene counts data set

# Subset gene couts data set to include selected
# genes only
gene_counts_data <- subset(gene_counts_raw_data, Description %in%
    X_linked_genes | Description %in% Y_linked_genes |
    Description %in% XIST)

# Start by transposing the gene counts data and
# transforming it to a data frame
gene_counts_data <- as.data.frame(t(gene_counts_data))

# Remove the first row
gene_counts_data <- gene_counts_data[-c(1), ]

# Rename column names to match gene names
colnames(gene_counts_data) <- gene_counts_data[c(1),
    ]

# Remove resulting first row of names
gene_counts_data <- gene_counts_data[-c(1), ]

# Store rownames in a column
gene_counts_data$CCLE_ID <- rownames(gene_counts_data)

# Move that column to the front
gene_counts_data <- gene_counts_data[, c(ncol(gene_counts_data),
    1:(ncol(gene_counts_data) - 1))]

# Remove resulting first row of names
gene_counts_data <- gene_counts_data[-c(1), ]

# Reset columns to null
rownames(gene_counts_data) <- NULL
```

Pre-processing annotation data set

```
# Pre-process annotation data set

# Only include cell lines with existing gene
# counts
annotation_data <- subset(annotation_raw_data, CCLE_ID %in%
    gene_counts_data$CCLE_ID)

# Subset dataframe to include columns of interest
annotation_data <- subset(annotation_data, select = c("CCLE_ID",
    "depMapID", "Name", "Gender"))

# Omit NA entries
annotation_data_noNA <- na.omit(annotation_data)

# Subset to include entries that are only male or
# female (some entries are just empty)
annotation_data_onlyMF <- subset(annotation_data_noNA,
    Gender != "")

# Rename gender column as reported gender
colnames(annotation_data_onlyMF)[4] <- "reportedPhenotype"

# Construct an empty string column to later
# include predicted phenotype
annotation_data_onlyMF$predictedPhenotype <- ""

# Construct an empty string column to later
# include predicted chromosome complement
annotation_data_onlyMF$predictedChromComp <- ""

# Reset columns to null
rownames(annotation_data_onlyMF) <- NULL
```

Filter gene counts data to only include cell lines in annotation data

```
# Filter gene counts data set to only match
# existing cell lines in annotation data set

# Only including cell lines in gene counts data
# that match the CCLE_ID column of the annotation
# data
gene_counts_data <- gene_counts_data[gene_counts_data$CCLE_ID %in%
    annotation_data_onlyMF$CCLE_ID, ]
```

Create sum of counts columns: X-linked and Y-Linked sums

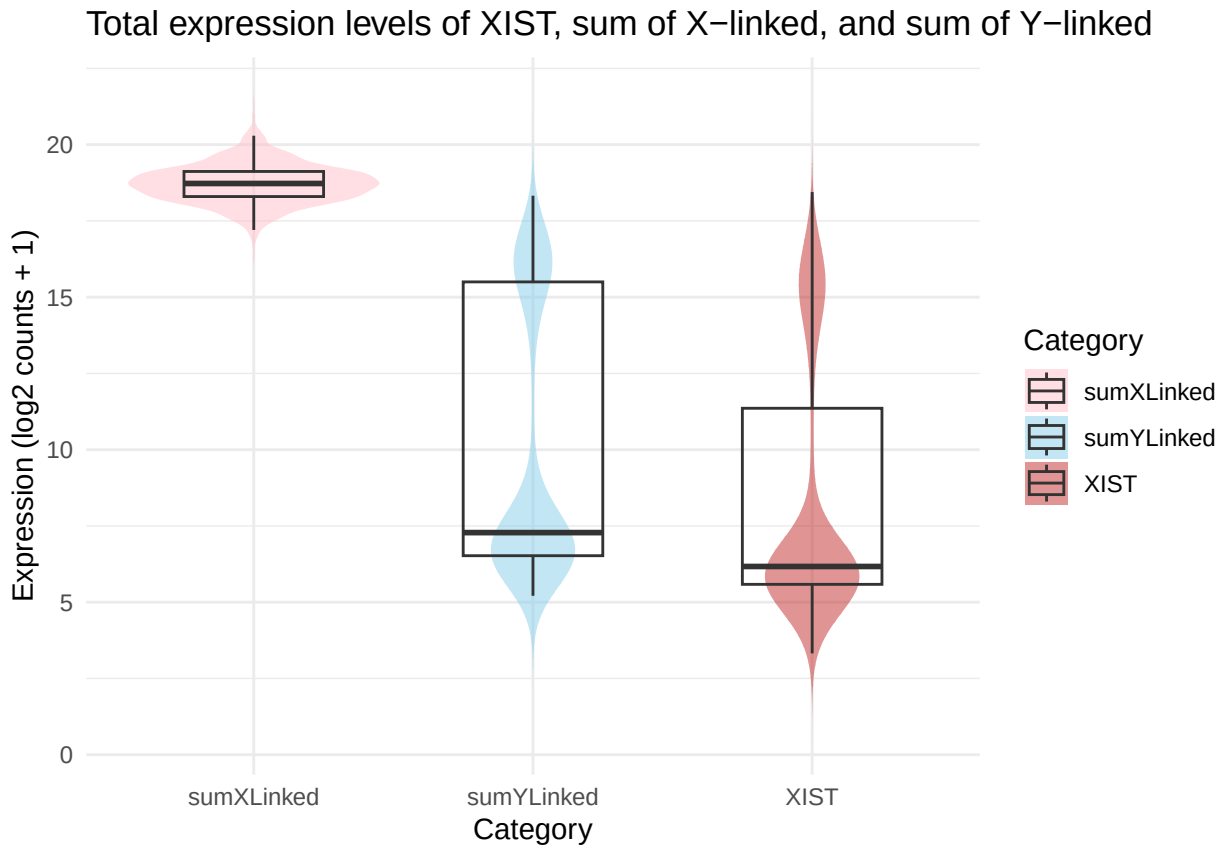
```r
# Create two columns, sum of X-linked expression
# and sum of Y-linked expression for each of the
# cell lines

# Convert all the columns listed in
# Y_linked_genes to numeric, then sum these
# columns for each row and create the sumYLinked
# column
gene_counts_data <- gene_counts_data %>%
    mutate(across(all_of(Y_linked_genes), as.numeric)) %>%
    mutate(sumYLinked = rowSums(select(., all_of(Y_linked_genes))))

# Convert all the columns listed in
# X_linked_genes to numeric, then sum these
# columns for each row and create the sumXLinked
# column
gene_counts_data <- gene_counts_data %>%
    mutate(across(all_of(X_linked_genes), as.numeric)) %>%
    mutate(sumXLinked = rowSums(select(., all_of(X_linked_genes))))

# Remove X-linked and Y-linked genes, resulting
# with XIST, sumYLinked, and sumXLinked
gene_counts_data <- gene_counts_data[, c("CCLE_ID",
    "XIST", "sumXLinked", "sumYLinked")]
```

Defining Thresholds for Classifier

Plot expression levels

```r
# Plot distributions of cell lines for each
# category and define thresholds as the
# corresponding medians of each category

# Transform gene counts data into longer format
# for proper plotting
long_data <- pivot_longer(gene_counts_data, cols = c(XIST,
    sumXLinked, sumYLinked), names_to = "Category",
    values_to = "Value")

# Apply log2 transformation to the values
long_data <- long_data %>%
    mutate(Expression = log2(Value + 1))


# Create the box plot with median lines as
# segments and store in an object for saving it
# later
plot_without_thresholds <- ggplot(long_data, aes(x = Category,
    y = Expression, fill = Category)) + geom_violin(trim = FALSE,
    color = NA, alpha = 0.5) + geom_boxplot(width = 0.5,
```

```
    outlier.shape = NA, alpha = 0) + scale_fill_manual(values = c("pink",
    "skyblue", "#c22b2b")) + labs(title = "Total expression levels of XIST, sum of X-linked, and sum of
    x = "Category", y = "Expression (log2 counts + 1)") +
    theme_minimal() + theme(legend.position = "right")

# Display plot object
plot_without_thresholds
```




```
# Save plot to results folder
ggsave("plot_without_thresholds.jpg", plot = plot_without_thresholds,
    path = results_dir, width = 10, height = 8, dpi = 600)
```

Define thresholds

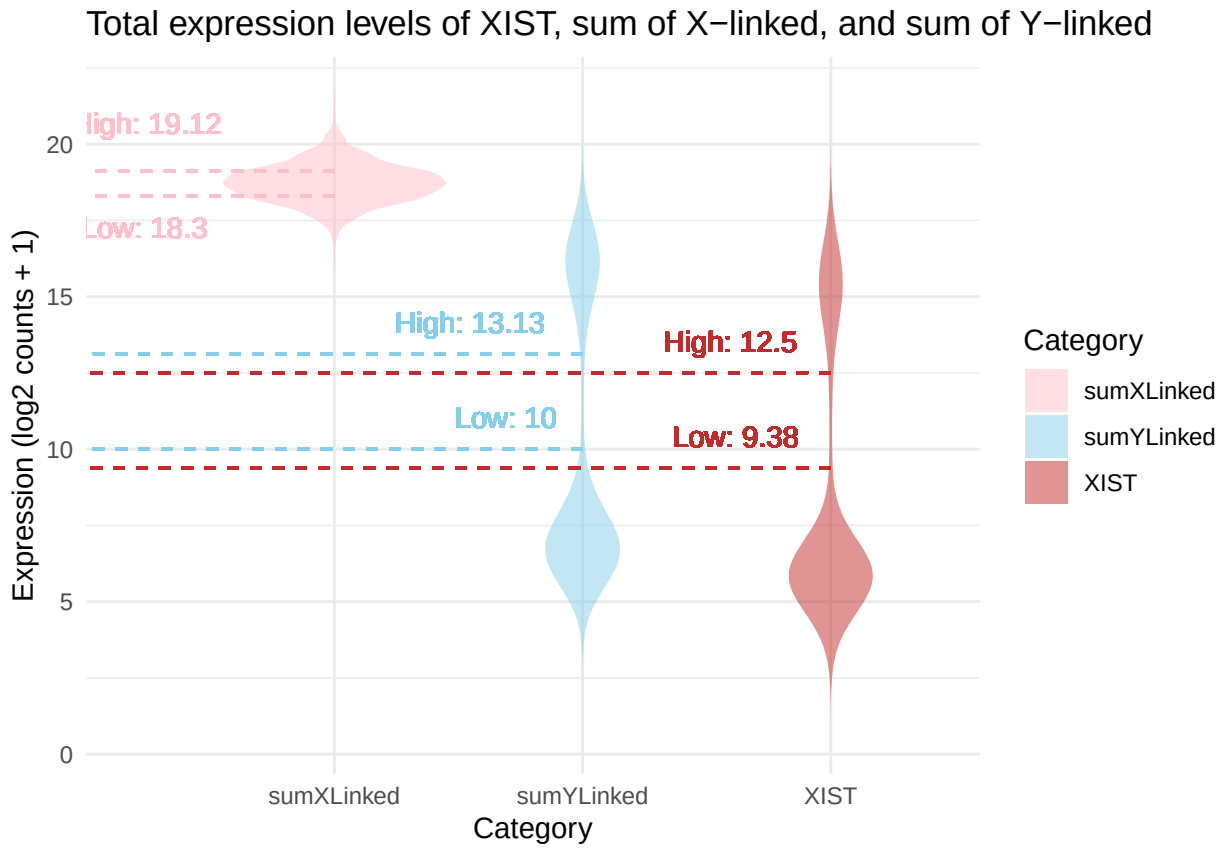
```
# Calculate thresholds based on the lower and
# upper percentiles of the inter-quartile range

# Filter data for the 'sumXLinked' category
sumXLinked_data <- long_data[long_data$Category ==
    "sumXLinked", "Expression"]
# convert data as numeric for calculation
sumXLinked_data <- as.numeric(sumXLinked_data$Expression)
# calculate the five number stats of a box
```

```r
# whisker to data of hinges
fivenum_results <- fivenum(sumXLinked_data)

# Extract the lower and upper hinges Lower hinge
# (25th percentile)
lower_hinge <- fivenum_results[2]
# Upper hinge (75th percentile)
upper_hinge <- fivenum_results[4]

# Set the thresholds for the sum of X-linked gene
# expression as whisker limits
sumXLinked_threshold_high <- round(upper_hinge, 2)
sumXLinked_threshold_low <- round(lower_hinge, 2)

# Observed thresholds based on medians

# Set the thresholds for the sum of Y-linked gene
# expression
sumYLinked_threshold_high <- 13.13
sumYLinked_threshold_low <- 10

# Set the thresholds for XIST gene expression
XIST_threshold_high <- 12.5
XIST_threshold_low <- 9.38
```

Plot thresholds

```r
# Create the plot with thresholds
plot_with_thresholds <- ggplot(long_data, aes(x = Category,
    y = Expression, fill = Category)) + geom_violin(trim = FALSE,
    color = NA, alpha = 0.5) + scale_fill_manual(values = c("pink",
    "skyblue", "#c22b2b")) + geom_segment(aes(x = 1,
    xend = 0, y = sumXLinked_threshold_low, yend = sumXLinked_threshold_low),
    color = "pink", linetype = "dashed") + geom_segment(aes(x = 1,
    xend = 0, y = sumXLinked_threshold_high, yend = sumXLinked_threshold_high),
    color = "pink", linetype = "dashed") + geom_segment(aes(x = 2,
    xend = 0, y = sumYLinked_threshold_low, yend = sumYLinked_threshold_low),
    color = "skyblue", linetype = "dashed") + geom_segment(aes(x = 2,
    xend = 0, y = sumYLinked_threshold_high, yend = sumYLinked_threshold_high),
    color = "skyblue", linetype = "dashed") + geom_segment(aes(x = 3,
    xend = 0, y = XIST_threshold_low, yend = XIST_threshold_low),
    color = "#c22b2b", linetype = "dashed") + geom_segment(aes(x = 3,
    xend = 0, y = XIST_threshold_high, yend = XIST_threshold_high),
    color = "#c22b2b", linetype = "dashed") + geom_text(aes(x = "sumXLinked",
    y = sumXLinked_threshold_low, label = paste("Low:",
        sumXLinked_threshold_low)), vjust = 2, hjust = 2,
    color = "pink") + geom_text(aes(x = "sumXLinked",
    y = sumXLinked_threshold_high, label = paste("High:",
        sumXLinked_threshold_high)), vjust = -1.75,
    hjust = 1.75, color = "pink") + geom_text(aes(x = "sumYLinked",
    y = sumYLinked_threshold_low, label = paste("Low:",
        sumYLinked_threshold_low)), vjust = -1, hjust = 1.25,
```

```
    color = "skyblue") + geom_text(aes(x = "sumYLinked",
    y = sumYLinked_threshold_high, label = paste("High:",
        sumYLinked_threshold_high)), vjust = -1, hjust = 1.25,
    color = "skyblue") + geom_text(aes(x = "XIST",
    y = XIST_threshold_low, label = paste("Low:", XIST_threshold_low)),
    vjust = -1, hjust = 1.25, color = "#c22b2b") +
    geom_text(aes(x = "XIST", y = XIST_threshold_high,
        label = paste("High:", XIST_threshold_high)),
        vjust = -1, hjust = 1.25, color = "#c22b2b") +
    labs(title = "Total expression levels of XIST, sum of X-linked, and sum of Y-linked",
        x = "Category", y = "Expression (log2 counts + 1)") +
    theme_minimal() + theme(legend.position = "right")

# Display plot object
plot_with_thresholds
```




```
# Save plot to results folder
ggsave("plot_with_thresholds.jpg", plot = plot_with_thresholds,
    path = results_dir, width = 10, height = 8, dpi = 600)
```

A Preliminary Classification Model for Sex Prediction

Construct classifying model and assign predictions

```
# Start of classifier construction based on
# predetermined gene expression thresholds based
# on median values from the plot

# Begin loop to iterate through each row of the
# annotation data frame
for (i in 1:nrow(annotation_data_onlyMF)) {

    # Retrieve the cell line ID for the current
    # row
    current_cell_line <- annotation_data_onlyMF$CCLE_ID[i]

    # Subset gene count data to get the row
    # corresponding to the current cell line
    current_row_of_counts <- subset(gene_counts_data,
        CCLE_ID == current_cell_line)

    # Calculate the log-transformed value of XIST
    # gene expression for the current row
    current_XIST_value <- log2(current_row_of_counts[2] +
        1)

    # Calculate the log-transformed sum of
    # X-linked genes expression for the current
    # row
    current_sumXLinked_value <- log2(current_row_of_counts[3] +
        1)

    # Calculate the log-transformed sum of
    # Y-linked genes expression for the current
    # row
    current_sumYLinked_value <- log2(current_row_of_counts[4] +
        1)

    # Classify as male with XY if criteria are
    # met [H(X) OR M(X) OR L(X)] AND H(Y) AND
    # NO(XIST)
    if (current_sumXLinked_value >= 0 & current_sumYLinked_value >
        sumYLinked_threshold_high & current_XIST_value ==
        0) {
        annotation_data_onlyMF$predictedPhenotype[i] <- "male"
        annotation_data_onlyMF$predictedChromComp[i] <- "XY"

        # Classify as female with XX if criteria
        # are met [H(X) OR M(X)] AND [L(Y) OR
        # NO(Y)] AND H(XIST)
    } else if (current_sumXLinked_value > sumXLinked_threshold_low &
        current_sumYLinked_value <= sumYLinked_threshold_low &
        current_XIST_value == 0) {
```

```
        annotation_data_onlyMF$predictedPhenotype[i] <- "female"
        annotation_data_onlyMF$predictedChromComp[i] <- "XX"

        # Classify as male with XXrY if criteria
        # are met [H(X) OR M(X)] AND H(Y) AND
        # L(XIST)
    } else if (current_sumXLinked_value > sumXLinked_threshold_low &
        current_sumYLinked_value >= sumYLinked_threshold_high &
        current_XIST_value <= XIST_threshold_low) {
        annotation_data_onlyMF$predictedPhenotype[i] <- "male"
        annotation_data_onlyMF$predictedChromComp[i] <- "XXrY"

        # Classify as male with XXY if criteria
        # are met [H(X) OR M(X)] AND H(Y) AND
        # H(XIST)
    } else if (current_sumXLinked_value > sumXLinked_threshold_low &
        current_sumYLinked_value >= sumYLinked_threshold_high &
        current_XIST_value >= XIST_threshold_high) {
        annotation_data_onlyMF$predictedPhenotype[i] <- "male"
        annotation_data_onlyMF$predictedChromComp[i] <- "XXY"

        # Classify as male with LOY if criteria
        # are met L(X) AND L(Y) AND NO(XIST)
    } else if (current_sumXLinked_value <= sumXLinked_threshold_low &
        current_sumYLinked_value <= sumYLinked_threshold_low &
        current_XIST_value == 0) {
        annotation_data_onlyMF$predictedPhenotype[i] <- "male"
        annotation_data_onlyMF$predictedChromComp[i] <- "LOY"

        # Classify as female with XXr if criteria
        # are met [H(X) OR M(X)] AND [L(Y) OR
        # NO(Y)] AND [L(XIST) OR NO(XIST)]
    } else if (current_sumXLinked_value > sumXLinked_threshold_low &
        current_sumYLinked_value <= sumYLinked_threshold_low &
        current_XIST_value <= XIST_threshold_low) {
        annotation_data_onlyMF$predictedPhenotype[i] <- "female"
        annotation_data_onlyMF$predictedChromComp[i] <- "XXr"

        # Classify as female with XO if criteria
        # are met L(X) AND NO(Y) AND [L(XIST) OR
        # NO(XIST)]
    } else if (current_sumXLinked_value <= sumXLinked_threshold_low &
        current_sumYLinked_value == 0 & current_XIST_value >=
        XIST_threshold_low) {
        annotation_data_onlyMF$predictedPhenotype[i] <- "female"
        annotation_data_onlyMF$predictedChromComp[i] <- "XO"

        # Assign NA for cases that do not meet
        # any of the above criteria
    } else {
        annotation_data_onlyMF$predictedPhenotype[i] <- NA
        annotation_data_onlyMF$predictedChromComp[i] <- NA
    }
```

```
}
```

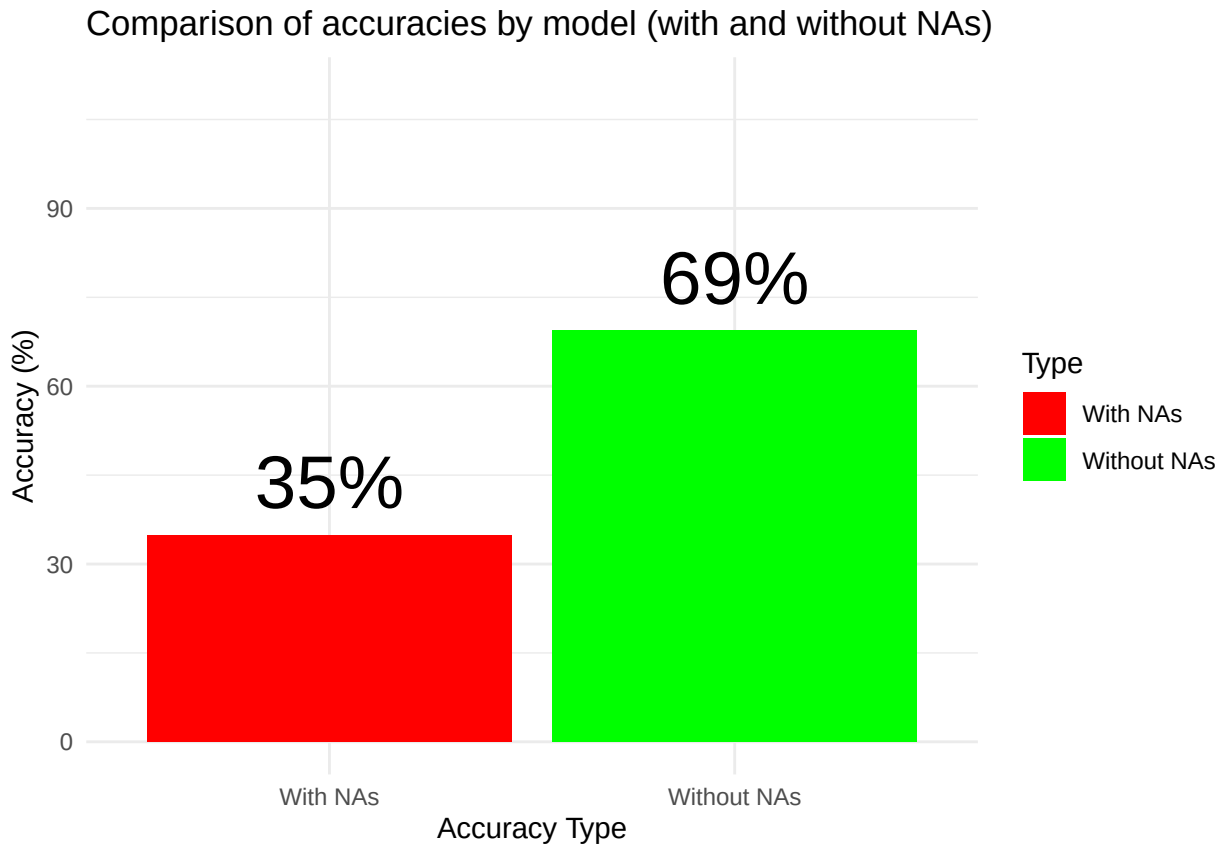
Calculate and plot model accuracy with and without NAs

```
# Using different metrics to access the
# classifier's robustness

# Counting total NA predictions
NA_predictions <- sum(is.na(annotation_data_onlyMF$predictedPhenotype))

# Counting failed predictions without NAs (where
# predicted does not equal reported)
failed_predictions <- sum(annotation_data_onlyMF$predictedPhenotype !=
    annotation_data_onlyMF$reportedPhenotype, na.rm = TRUE)

# Counting failed predictions with NAs (predicted
# != reported)
failed_withNA_predictions <- NA_predictions + failed_predictions

# Counting correct predictions without NAs
# (predicted = reporte)
correct_predictions <- sum(annotation_data_onlyMF$predictedPhenotype ==
    annotation_data_onlyMF$reportedPhenotype, na.rm = TRUE)

# Counting correct predictions without NAs
# (predicted = reporte)
correct_withNA_predictions <- NA_predictions + correct_predictions

# Accuracy of correct predictions (including NAs
# as incorrect)
total_withNA_predictions <- NA_predictions + failed_predictions +
    correct_predictions
accuracy_withNA <- (correct_predictions/total_withNA_predictions) *
    100

# Print the accuracy
print(accuracy_withNA)
```

```
## [1] 34.91879
```

```
# Accuracy of correct predictions (excluding NAs)
total_predictions <- failed_predictions + correct_predictions
accuracy <- (correct_predictions/total_predictions) *
    100

# Creating a data frame for plotting
accuracy_data <- data.frame(Type = c("With NAs", "Without NAs"),
    Accuracy = c(accuracy_withNA, accuracy))

# PLot Accuracies
accuracy_plot <- ggplot(accuracy_data, aes(x = Type,
```

```
    y = Accuracy, fill = Type)) + geom_bar(stat = "identity",
    position = position_dodge()) + geom_text(aes(label = paste0(round(Accuracy,
    0), "%")), vjust = -0.5, color = "black", size = 10) +
    ylim(0, 110) + scale_fill_manual(values = c("red",
    "green")) + labs(title = "Comparison of accuracies by model (with and without NAs)",
    x = "Accuracy Type", y = "Accuracy (%)") + theme_minimal()

# Plot accuracy
accuracy_plot
```

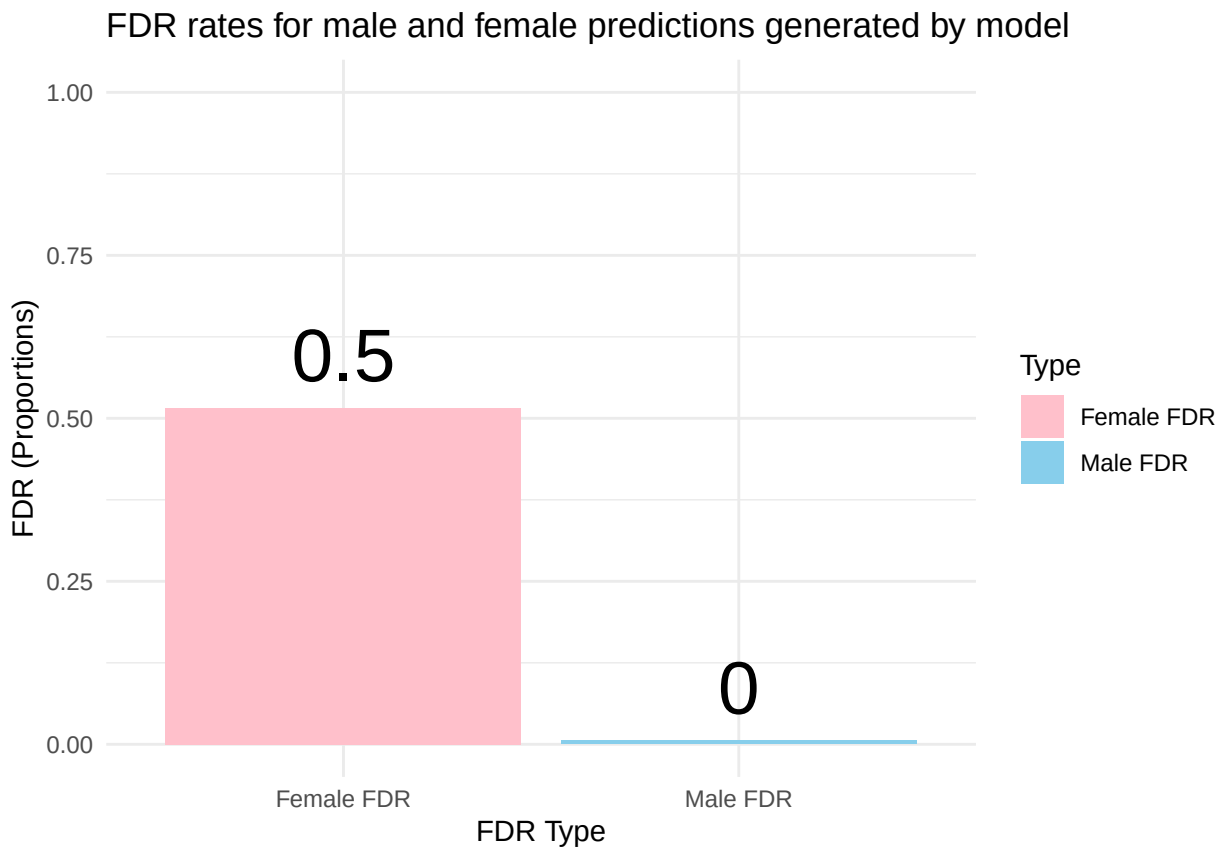



```
# Save plot to results folder
ggsave("accuracy_plot.jpg", plot = accuracy_plot, path = results_dir,
    width = 10, height = 8, dpi = 600)
```

Calculate and plot FDR rates for male and female predictions

```
# Calculating male FDR = FP / (FP + TP)

# True positives for males
TP_male <- sum(annotation_data_onlyMF$predictedPhenotype ==
    "male" & annotation_data_onlyMF$reportedPhenotype ==
    "male", na.rm = TRUE)
```

```r
# False positves for males
FP_male <- sum(annotation_data_onlyMF$predictedPhenotype ==
    "male" & annotation_data_onlyMF$reportedPhenotype ==
    "female", na.rm = TRUE)

# Male FDR
male_FDR <- FP_male/(FP_male + TP_male)

# Calculating female FDR = FP / (FP + TP)

# True positives for females
TP_female <- sum(annotation_data_onlyMF$predictedPhenotype ==
    "female" & annotation_data_onlyMF$reportedPhenotype ==
    "female", na.rm = TRUE)

# False positves for females
FP_female <- sum(annotation_data_onlyMF$predictedPhenotype ==
    "female" & annotation_data_onlyMF$reportedPhenotype ==
    "male", na.rm = TRUE)

# Female FDR
female_FDR <- FP_female/(FP_female + TP_female)

# Creating a data frame for plotting
FDR_data <- data.frame(Type = c("Female FDR", "Male FDR"),
    Accuracy = c(female_FDR, male_FDR))

# PLot Accuracies
FDR_plot <- ggplot(FDR_data, aes(x = Type, y = Accuracy,
    fill = Type)) + geom_bar(stat = "identity", position = position_dodge()) +
    geom_text(aes(label = round(Accuracy, 1)), vjust = -0.5,
        color = "black", size = 10) + scale_y_continuous(limits = c(0,
    1)) + scale_fill_manual(values = c("pink", "skyblue")) +
    labs(title = "FDR rates for male and female predictions generated by model",
        x = "FDR Type", y = "FDR (Proportions)") +
    theme_minimal()

# PLot FDR
FDR_plot
```




```
# Save plot to results folder
ggsave("FDR_plot.jpg", plot = FDR_plot, path = results_dir,
    width = 10, height = 8, dpi = 600)
```

Model Results

```
# Here we output the main file of predictions
# Upon improving accuracy of classifying model,
# the clasifier can be used to predict sex
# phenotypes and complements for the rest of the
# cell lines in the CCLE gene counts data, and
# any other cell line data with available gene
# counts

# Outputting modified annotation data file that
# includes results of classifier
# (annotation_data_onlyMF)
write.csv(annotation_data_onlyMF, file = paste0(results_dir,
    "annotation_data_onlyMF.csv"), row.names = FALSE)
```

Session Info

```
# Collect information about current used session
sessionInfo()
```

```
## R version 4.3.1 (2023-06-16 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 11 x64 (build 22621)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=English_United States.utf8
## [2] LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.utf8
##
## time zone: America/New_York
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] tidyr_1.3.0   dplyr_1.1.3   ggplot2_3.4.4
##
## loaded via a namespace (and not attached):
##  [1] vctrs_0.6.4       cli_3.6.1         knitr_1.45        rlang_1.1.1
##  [5] xfun_0.40         highr_0.10        purrr_1.0.2       generics_0.1.3
##  [9] labeling_0.4.3    glue_1.6.2        colorspace_2.1-0  htmltools_0.5.6.1
## [13] formatR_1.14      scales_1.2.1      fansi_1.0.5       rmarkdown_2.25
## [17] grid_4.3.1        evaluate_0.23     munsell_0.5.0     tibble_3.2.1
## [21] fastmap_1.1.1     yaml_2.3.7        lifecycle_1.0.4   compiler_4.3.1
## [25] pkgconfig_2.0.3   rstudioapi_0.15.0 farver_2.1.1      digest_0.6.33
## [29] R6_2.5.1          tidyselect_1.2.0  utf8_1.2.3        pillar_1.9.0
## [33] magrittr_2.0.3    withr_2.5.2       tools_4.3.1       gtable_0.3.4
```