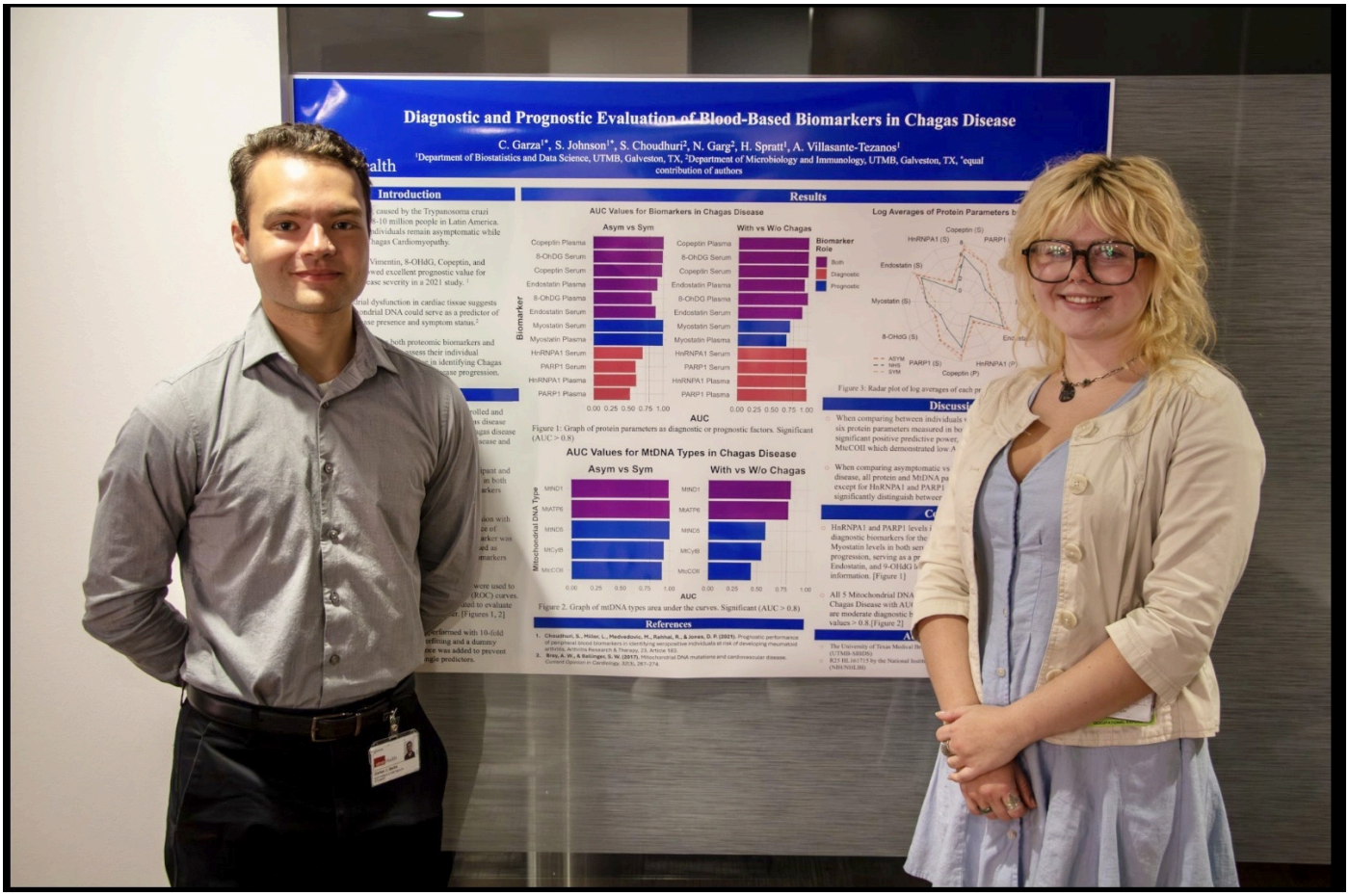
# SIBDS Chagas Disease Project

# Chagas Disease Biomarker Analysis




During my internship at the **University of Texas Medical Branch (UTMB)** in the Department of Data Science and Biostatistics, I worked under the guidance of **Dr. Garg** and **Dr. Choudhuri,** along with my partner Summer Johnson. My role was to analyze biomedical data provided by our Principal Investigator, derive insights, and present the findings.

The research aimed to identify **diagnostic** and **prognostic** biomarkers for *Chagas Disease* — a cardiovascular illness caused by the parasite *T. cruzi*. The disease is widespread in Latin America and is responsible for an estimated 3 million deaths annually. Our long-term goal was to discover blood-based biomarkers that could inform vaccine development and enable more accessible diagnostic tests.

Our data set contained:

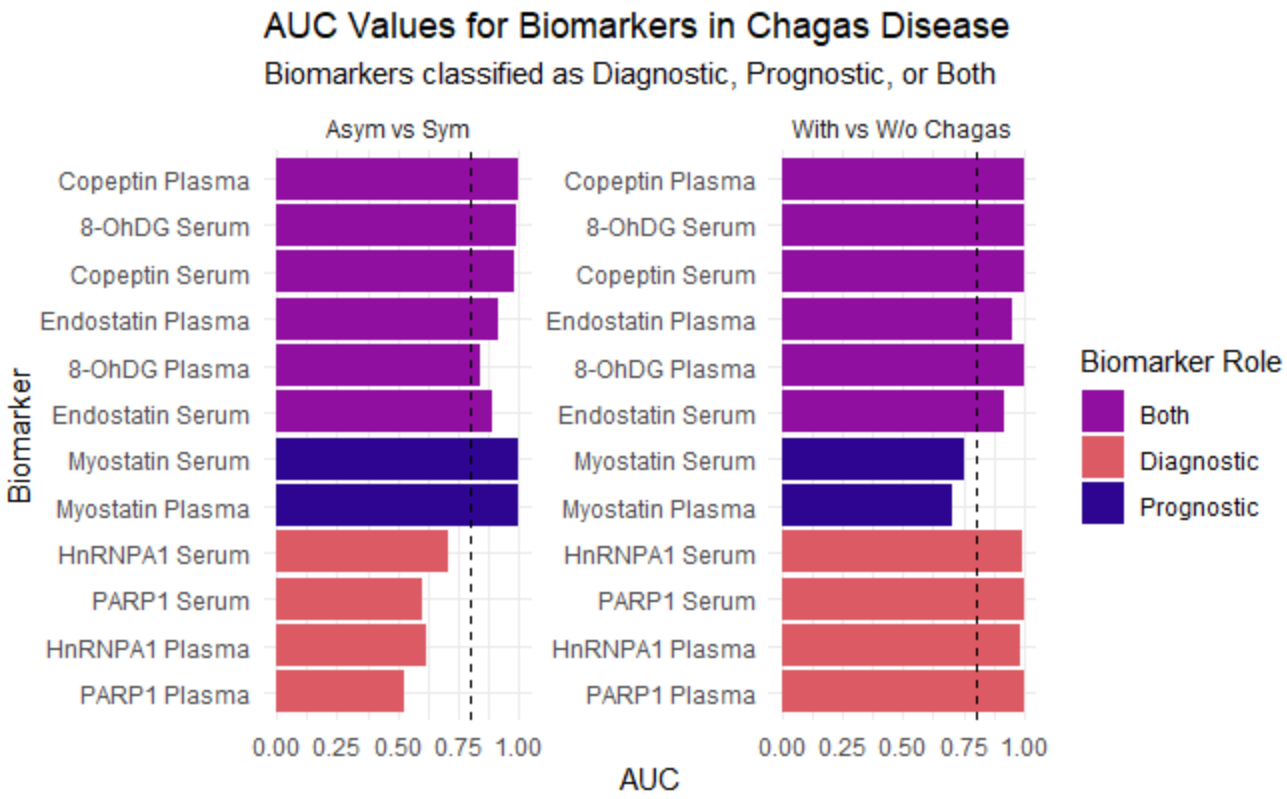
- 42 subjects total
  - 12 Normal Healthy Subjects (controls)
  - 30 with Chagas Disease
    - 15 symptomatic
    - 15 asymptomatic
- Biomarkers recorded:

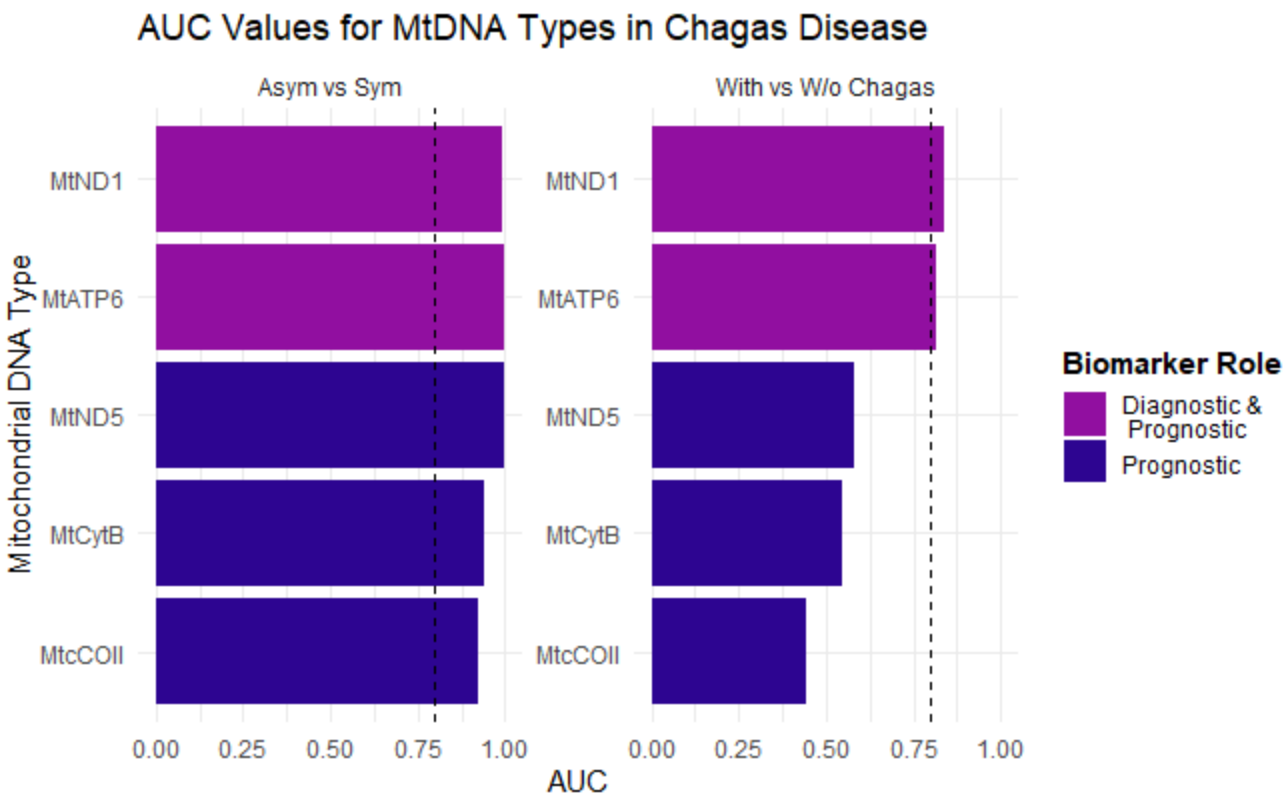
- 6 protein parameters, measured in both plasma and serum (12 total protein biomarkers)
- 5 mitochondrial DNA (mtDNA) types

With my research partner Summer Johnson, we performed logistic regression modeling for each biomarker:

- Diagnostic biomarkers: distinguish Chagas patients from Normal Healthy Subjects.
- Prognostic biomarkers: distinguish symptomatic from asymptomatic patients.

Several biomarkers demonstrated strong diagnostic and prognostic potential. Below are the final results (plots + tables).






Below are step by step processes used for this research.

# Install & Load Packages

```
packages <- c(
  "broom", "corrplot", "dplyr", "emmeans","ggplot2", "ggpubr", "ggrepel","fsmb", "gridExtra", "gr
"readxl", "rlang", "randomForest", "tidyverse","tidyr", "tree", "writexl"
)

# Packages will only be installed if not already!
for (pkg in packages) {
  if (!requireNamespace(pkg, quietly = TRUE)) {
    #install.packages(pkg)
  }
}

library(broom)
library(corrplot)
library(dplyr)
library(emmeans)
library(ggplot2)
library(ggpubr)
library(ggrepel)
library(fmsb) #radar plot
library(gridExtra)
library(grid)
library(glmnet)
```

```r
library(gbm)
library(ISLR)
library(leaps)
library(MASS)
library(nnet)
library(pROC)
library(readxl)
library(rlang)
library(randomForest)
library(tidyverse)
library(tidyr)
library(tree)
library(writexl)
```

# Clean Data

Adjusted the data set to create a column for each protein & type. 8 protein parameters, 2 types. 16 total. Then create numeric values for categorical data NHS, SYM, ASM ->1s or 0s.

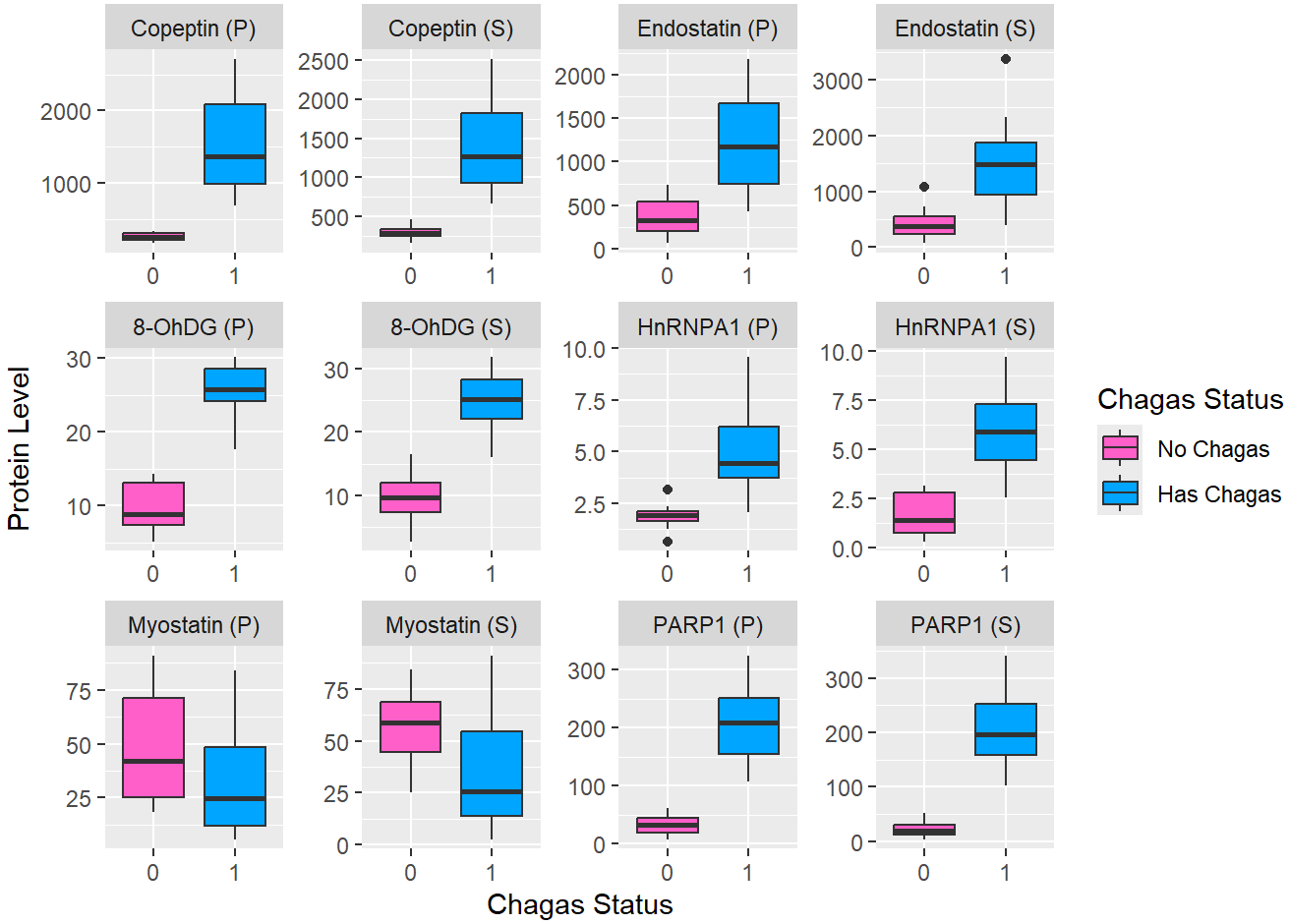
```r
data <- read_xlsx("~/R Notes/proteinparam.xlsx")

#Create columns for each protein parameter and test, new column ordinal chagas present
data <- data %>%
  pivot_wider(names_from = Moleculae, values_from = c(Serum, Plasma),
              names_glue = "{Moleculae}_{.value}") %>%
  mutate(
    chag = case_when(
      Symptom == "NHS" ~ 0,
      Symptom == "ASYM" | Symptom == "SYM" ~ 1,
      TRUE ~ NA_real_
    ),
    sym = case_when(
      Symptom == "ASYM" ~ 0,
      Symptom == "SYM" ~ 1,
      TRUE ~ NA_real_
    )
  )
```

# Exploratory Data Analysis

Create box plots to show variability of protein parameter expression.

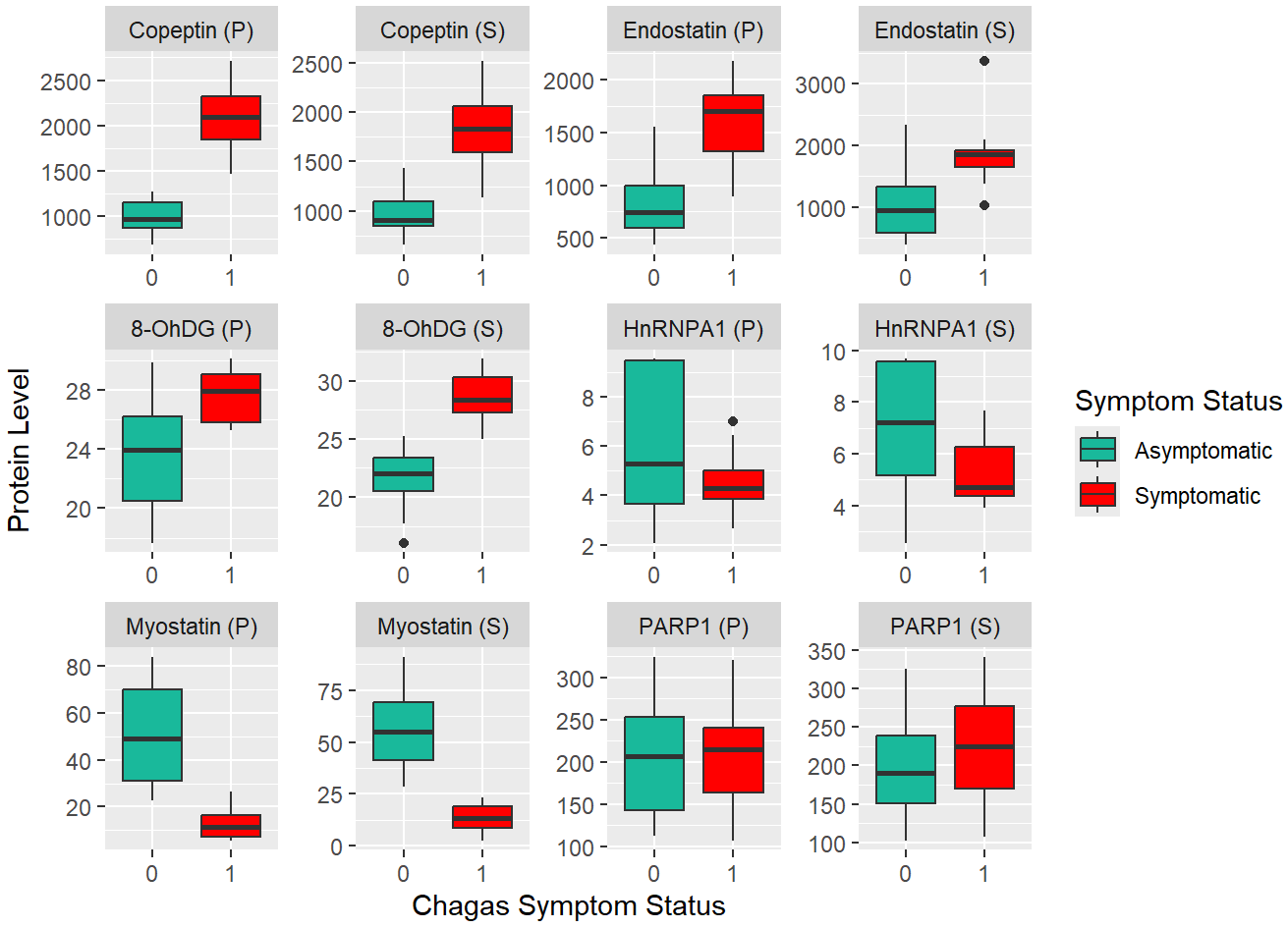
```r
# Create a list marker_labels full of each biomarker
marker_labels <- c(
  "Copeptin_Serum" = "Copeptin (S)",
  "Copeptin_Plasma" = "Copeptin (P)",
```

```
  "HnRNPA1_Serum" = "HnRNPA1 (S)",
  "HnRNPA1_Plasma" = "HnRNPA1 (P)",
  "Endostatin_Serum" = "Endostatin (S)",
  "Endostatin_Plasma" = "Endostatin (P)",
  "etOhDG_Serum" = "8-OhDG (S)",
  "etOhDG_Plasma" = "8-OhDG (P)",
  "Myostatin_Serum" = "Myostatin (S)",
  "Myostatin_Plasma" = "Myostatin (P)",
  "PARP1_Serum" = "PARP1 (S)",
  "PARP1_Plasma" = "PARP1 (P)"
)

#Chagas/ No Chagas boxplot
df_long <- data %>%
  pivot_longer(cols = ends_with("_Serum") | ends_with("_Plasma"),
               names_to = "marker",
               values_to = "value")

chag_labels <- c("0" = "No Chagas", "1" = "Has Chagas")

ggplot(data = df_long,
       aes(x = factor(chag), y = value, fill = factor(chag))) +
  geom_boxplot() +
  facet_wrap(~ marker,
             scales = "free",
             labeller = labeller(marker = marker_labels)) +
  scale_fill_manual(
    name = "Chagas Status",
    values = c("0" = "#FF61CC", "1" = "#00A9FF"),
    labels = chag_labels
  ) +
  labs(
    x = "Chagas Status",
    y = "Protein Level",
    fill = "Symptom Status"
  )
```



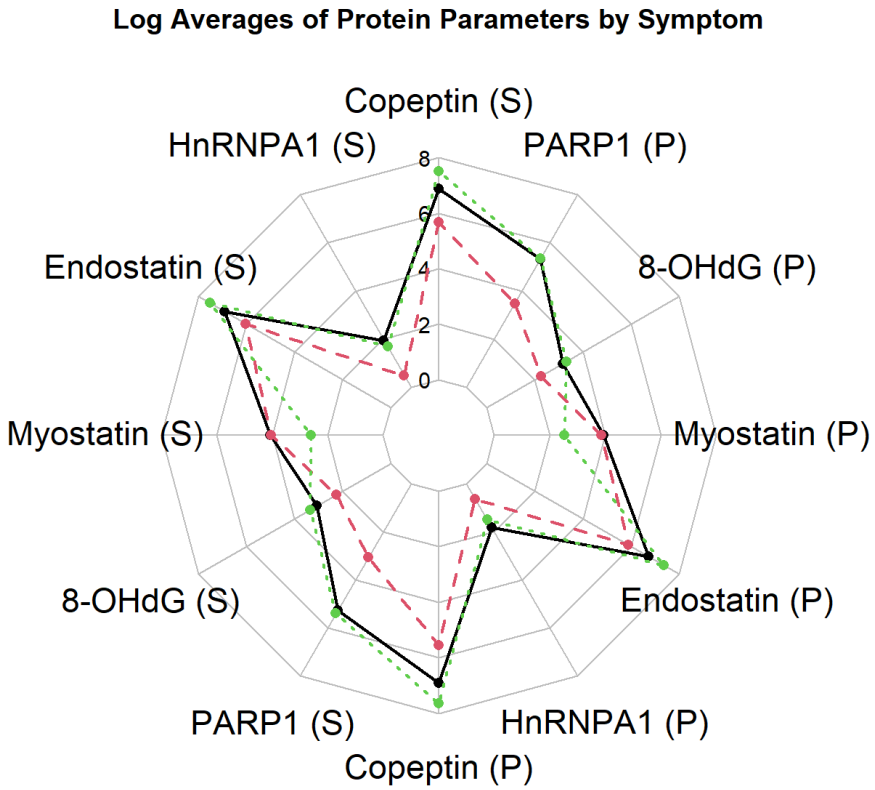

```r
#Sym/Asym boxplot
df_long2 <- data %>%
  pivot_longer(cols = ends_with("_Serum") | ends_with("_Plasma"),
               names_to = "marker",
               values_to = "value") %>%
  filter(sym == 0 | sym == 1)

sym_labels <- c("0" = "Asymptomatic", "1" = "Symptomatic")

ggplot(df_long2,
       aes(x = factor(sym), y = value, fill = factor(sym))) +
  geom_boxplot() +
  facet_wrap(~ marker,
             scales = "free",
             labeller = labeller(marker = marker_labels)) +
  scale_fill_manual(
    name = "Symptom Status",
    values = c("0" = "#1ABC9C", "1" = "red"),
    labels = sym_labels
  ) +
  labs(
    x = "Chagas Symptom Status",
    y = "Protein Level",
```

```
    fill = "Symptom Status"
  )
```




```
#Group averages
avg_data <- data %>%
  group_by(Symptom) %>%
  summarise(across(matches("(_Serum|_Plasma)$"), mean, na.rm = TRUE)) %>%
  as.data.frame()

log_data <- avg_data %>%
  mutate(across(where(is.numeric), log))

# Set rownames from Symptom and remove that column
rownames(log_data) <- log_data$Symptom
log_data$Symptom <- NULL

# Set min and max values explicitly
min_val <- 0  # your observed raw minimum
max_val <- 8  # your observed raw maximum

custom_labels <- c("Copeptin (S)", "HnRNPA1 (S)", "Endostatin (S)", "Myostatin (S)",
                   "8-OHdG (S)", "PARP1 (S)", "Copeptin (P)", "HnRNPA1 (P)", "Endostatin (P)", "My
                   "8-OHdG (P)", "PARP1 (P)")
```

```r
# Create df with required min and max rows
df_radar <- log_data
colnames(df_radar) <- custom_labels
df_radar <- rbind(rep(max_val, ncol(df_radar)),
                  rep(min_val, ncol(df_radar)),
                  df_radar)

# Define your custom sizes
axis_label_size <- 0.8     # Size of the axis label text
variable_label_size <- 1.2 # Size of the variable labels (vlcex)
title_size <- 1.5          # Size of the title


radarchart(df_radar,
           axistype = 1,
           caxislabels = round(seq(min_val, max_val, length.out = 5), 2),
           plwd = 2,
           cglcol = "grey",
           cglty = 1,
           axislabcol = "black",
           calcex = axis_label_size * 1.2,   # scale axis labels
           vlcex = variable_label_size * 1.2, # scale variable names
           title = "Log Averages of Protein Parameters by Symptom"
)
```




```
print(avg_data)
```

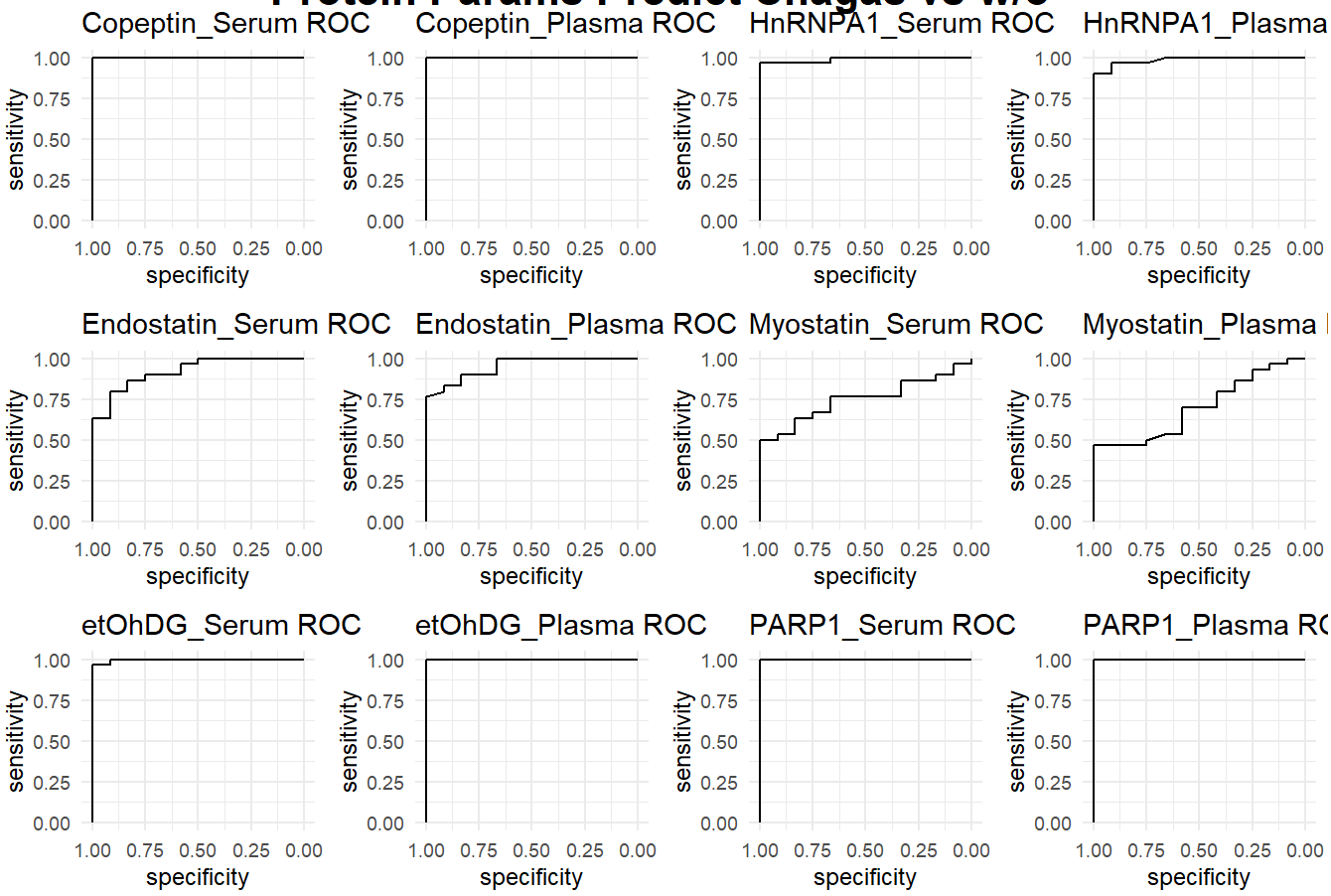
```
  Symptom Copeptin_Serum HnRNPA1_Serum Endostatin_Serum Myostatin_Serum
1    ASYM       961.3104      6.997077        1006.0905        57.74151
2     NHS        291.0829      1.647571         422.1989        56.70189
3     SYM      1840.1470      5.365387        1841.0096        13.58880
  etOhDG_Serum PARP1_Serum Copeptin_Plasma HnRNPA1_Plasma Endostatin_Plasma
1    21.600002    196.7232        999.8742       6.152438          836.1117
2     9.520813     21.4447        255.6459       1.889371          363.0873
3    28.515085    221.4311       2085.7043       4.514841         1580.6557
  Myostatin_Plasma etOhDG_Plasma PARP1_Plasma
1         51.32528     23.762719     205.4989
2         47.69188      9.742266      32.5762
3         12.54273     27.658256     209.2229
```

# Analysis

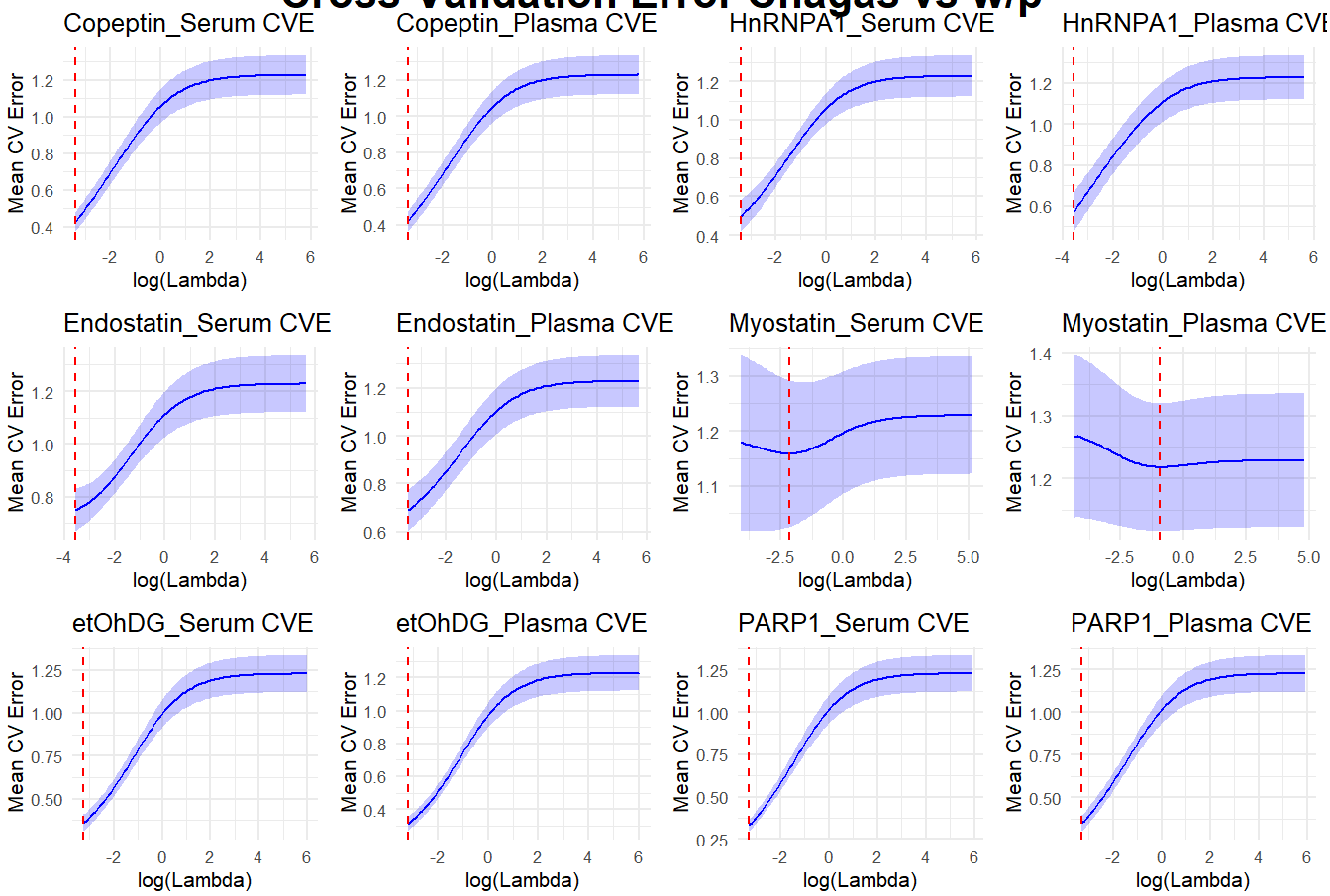
## Perform Logistic Regression to see if each Protein Parameter can classify whether a patient has or does not have Chagas Disease.

```r
list_roc_log_chag <- list()
list_cve_log_chag <- list()

# Create a summary table
summary_results <- data.frame(
  ProteinParameter = character(),
  AUC = numeric(),
  CVE = numeric(),
  stringsAsFactors = FALSE
)

protein_list <- c("Copeptin", "HnRNPA1", "Endostatin",
                  "Myostatin", "etOhDG", "PARP1")
sample_types <- c("Serum", "Plasma")


for (protein_name in protein_list) {
  for (sample_type in sample_types) {
    var_name <- paste0(protein_name, "_", sample_type)

    # Logistic regression
    formula <- as.formula(paste("chag ~", var_name))
    log.reg <- glm(formula, data = data, family = "binomial")
    #print(summary(log.reg))

    # Odds ratios and confidence intervals
    #cat("\nOdds Ratios:\n")
    #print(exp(coef(log.reg)))

    #cat("\n95% CI for Odds Ratios:\n")
    #print(exp(confint(log.reg)))

    # Predict probabilities
    pred.prob <- predict(log.reg, type = "response")

    # ROC and AUC
    roc_log <- roc(data$chag, pred.prob)
    roc_plot <- ggroc(roc_log) +
                ggtitle(paste(var_name, "ROC")) +
                theme_minimal(base_size = 9)
    list_roc_log_chag[[var_name]] <- roc_plot

    auc_val <- auc(roc_log)

     # CV error as ggplot
    set.seed(67)
    x <- as.matrix(cbind(var = data[[var_name]], dummy = rnorm(nrow(data), 0, 1e-6)))
    y <- as.factor(data$chag)
    cvfit <- cv.glmnet(x, y, family = "binomial", alpha = 0)
```

```
    cv_df <- data.frame(lambda = log(cvfit$lambda), cvm = cvfit$cvm, cvsd = cvfit$cvsd)

    # Extract CVE (at lambda.min)
    cve_val <- min(cvfit$cvm)

    cve_plot <- ggplot(cv_df, aes(x = lambda, y = cvm)) +
  geom_line(color = "blue") +
  geom_ribbon(aes(ymin = cvm - cvsd, ymax = cvm + cvsd), alpha = 0.2, fill = "blue") +
  geom_vline(xintercept = log(cvfit$lambda.min), linetype = "dashed", color = "red") +
  labs(title = paste(var_name, "CVE"), x = "log(Lambda)", y = "Mean CV Error") +
  theme_minimal(base_size = 8)

    list_cve_log_chag[[var_name]] <- cve_plot

  # Save to summary table
  summary_results <- rbind(summary_results, data.frame(
    ProteinParameter = var_name,
    AUC = auc_val,
    CVE = cve_val
  ))
  }
}

grid.arrange(grobs = list_roc_log_chag,
               top = textGrob("Protein Params Predict Chagas vs w/o",
                              just = "top",
                              gp = gpar(fontsize = 16, fontface = "bold")),
               ncol = 4)
```




```
grid.arrange(grobs = list_cve_log_chag,
             top = textGrob("Cross-Validation Error Chagas vs w/p",
                            just = "top",
                            gp = gpar(fontsize = 16, fontface = "bold")),
             ncol = 4)
```



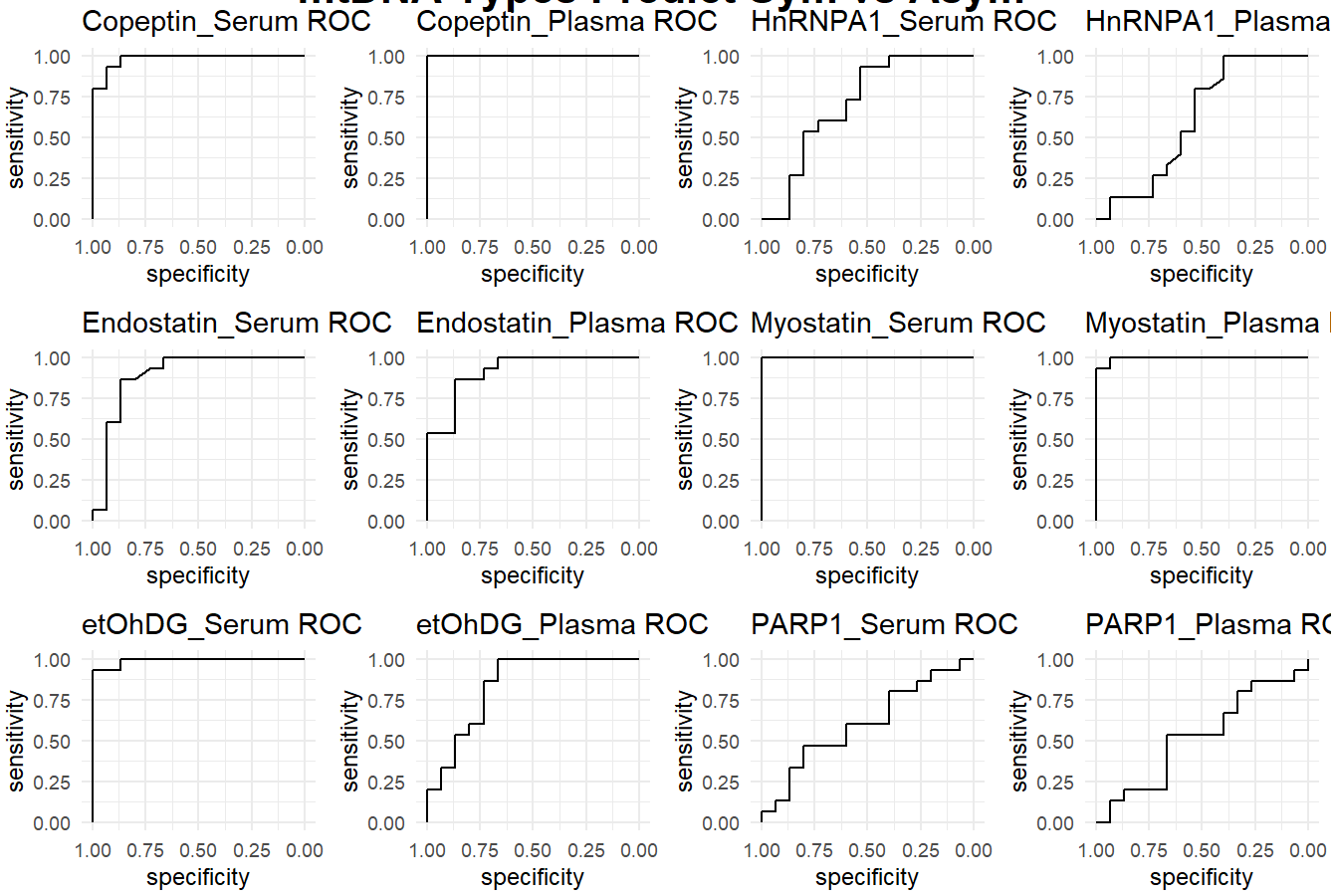

```
print(summary_results)
```

```
    ProteinParameter       AUC       CVE
1     Copeptin_Serum 1.0000000 0.4258889
2    Copeptin_Plasma 1.0000000 0.4209512
3      HnRNPA1_Serum 0.9888889 0.5012373
4     HnRNPA1_Plasma 0.9847222 0.5720976
5    Endostatin_Serum 0.9222222 0.7513106
6  Endostatin_Plasma 0.9513889 0.6884595
7     Myostatin_Serum 0.7500000 1.1589647
8    Myostatin_Plasma 0.7013889 1.2179962
9        etOhDG_Serum 0.9972222 0.3600053
10     etOhDG_Plasma 1.0000000 0.3096185
11       PARP1_Serum 1.0000000 0.3324260
12      PARP1_Plasma 1.0000000 0.3489596
```

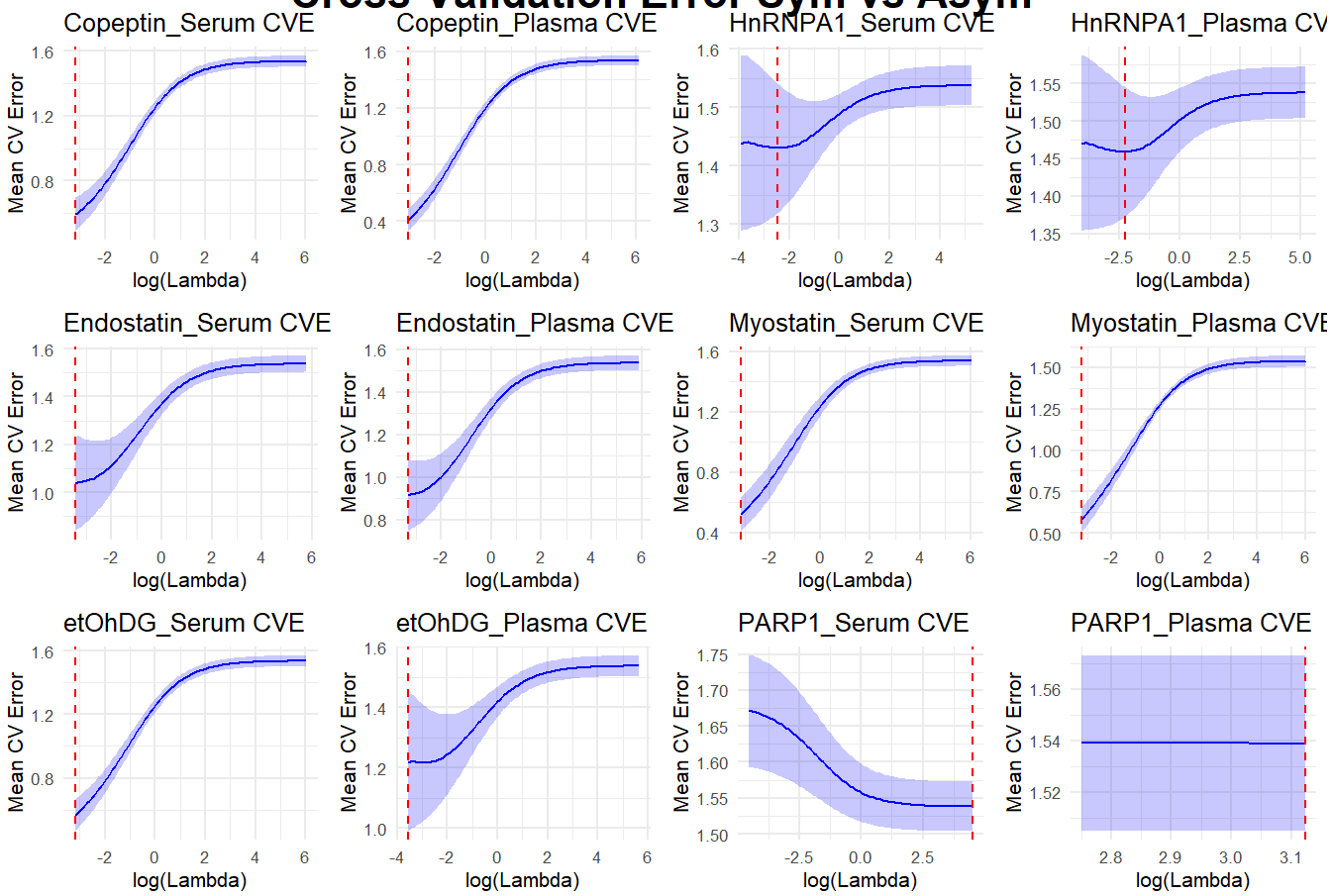
## Perform Logistic Regression to see if each Protein Parameter can classify whether a Chagas Patient is Symptomatic or Asymptomatic.

```r
list_roc_log_asymsym <- list()
list_cve_log_asymsym <- list()

# Create a summary table
summary_results <- data.frame(
  ProteinParameter = character(),
  AUC = numeric(),
  CVE = numeric(),
  stringsAsFactors = FALSE
)

protein_list <- c("Copeptin", "HnRNPA1", "Endostatin",
                  "Myostatin", "etOhDG", "PARP1")
sample_types <- c("Serum", "Plasma")

# Remove rows with NA in outcome or predictor
clean_data <- data[!is.na(data[[var_name]]) & !is.na(data$sym), ]

for (protein_name in protein_list) {
  for (sample_type in sample_types) {
    var_name <- paste0(protein_name, "_", sample_type)

    # Logistic regression
    formula <- as.formula(paste("sym ~", var_name))
    log.reg <- glm(formula, data = clean_data, family = "binomial")
    #print(summary(log.reg))

    # Odds ratios and confidence intervals
    #cat("\nOdds Ratios:\n")
    #print(exp(coef(log.reg)))

    #cat("\n95% CI for Odds Ratios:\n")
    #print(exp(confint(log.reg)))

    # Predict probabilities
    pred.prob <- predict(log.reg, type = "response")

    # ROC and AUC
    roc_log <- roc(clean_data$sym, pred.prob)
    roc_plot <- ggroc(roc_log) +
                ggtitle(paste(var_name, "ROC")) +
                theme_minimal(base_size = 9)
    list_roc_log_asymsym[[var_name]] <- roc_plot

    auc_val <- auc(roc_log)

    # CV error as ggplot
    set.seed(67)
    x <- as.matrix(cbind(var = clean_data[[var_name]], dummy = rnorm(nrow(clean_data), 0, 1e-6)))
    y <- as.factor(clean_data$sym)
```

```
    cvfit <- cv.glmnet(x, y, family = "binomial", alpha = 0)

    cv_df <- data.frame(lambda = log(cvfit$lambda), cvm = cvfit$cvm, cvsd = cvfit$cvsd)

    # Extract CVE (at lambda.min)
    cve_val <- min(cvfit$cvm)

    cve_plot <- ggplot(cv_df, aes(x = lambda, y = cvm)) +
  geom_line(color = "blue") +
  geom_ribbon(aes(ymin = cvm - cvsd, ymax = cvm + cvsd), alpha = 0.2, fill = "blue") +
  geom_vline(xintercept = log(cvfit$lambda.min), linetype = "dashed", color = "red") +
  labs(title = paste(var_name, "CVE"), x = "log(Lambda)", y = "Mean CV Error") +
  theme_minimal(base_size = 8)

  list_cve_log_asymsym[[var_name]] <- cve_plot

    # Save to summary table
  summary_results <- rbind(summary_results, data.frame(
    ProteinParameter = var_name,
    AUC = auc_val,
    CVE = cve_val
  ))

  }
}

grid.arrange(grobs = list_roc_log_asymsym,
               top = textGrob("mtDNA Types Predict Sym vs Asym",
                              just = "top",
                              gp = gpar(fontsize = 16, fontface = "bold")),
               ncol = 4)
```




```
grid.arrange(grobs = list_cve_log_asymsym,
             top = textGrob("Cross-Validation Error Sym vs Asym",
                            just = "top",
                            gp = gpar(fontsize = 16, fontface = "bold")),
             ncol = 4)
```




```
print(summary_results)
```

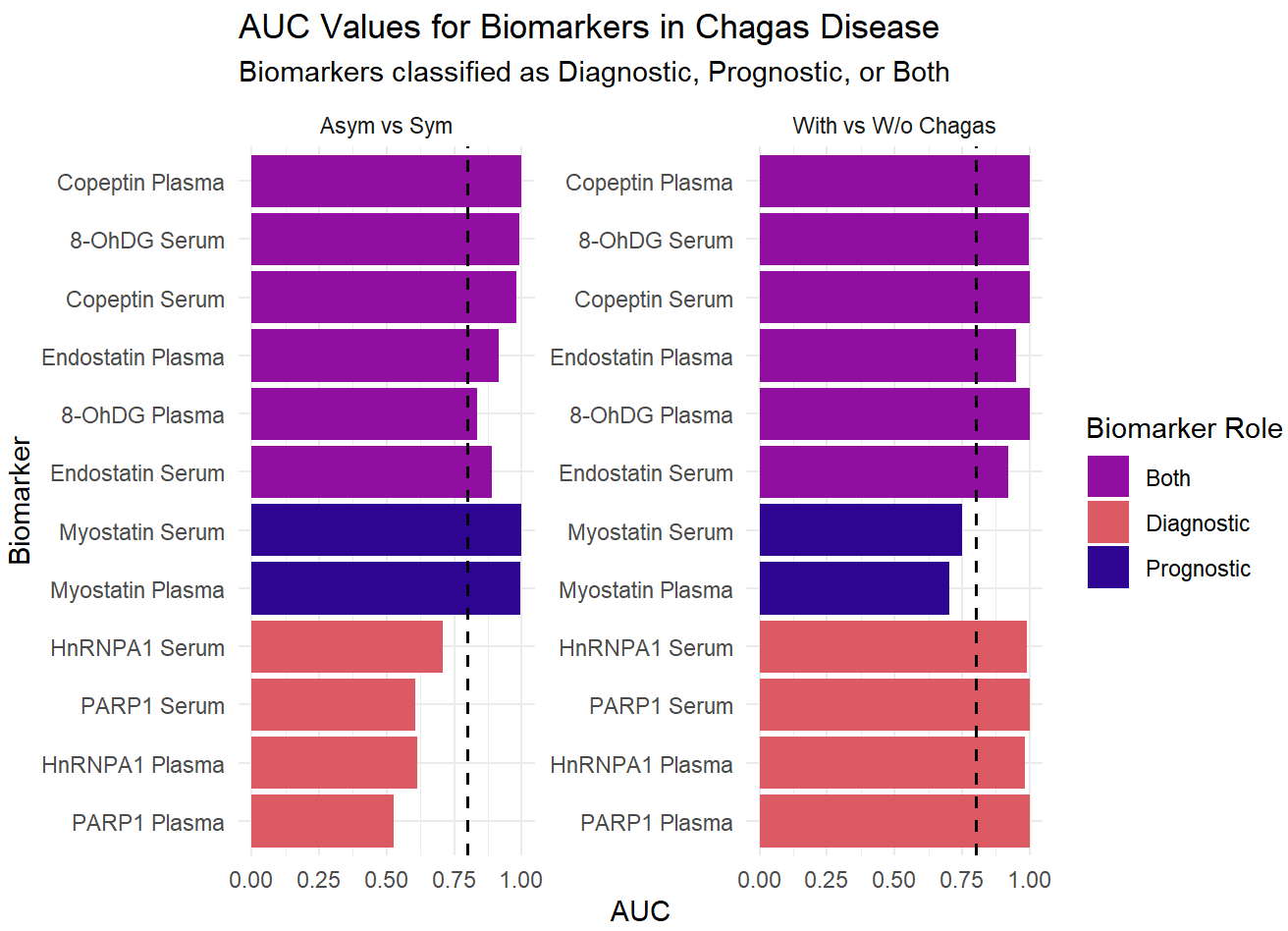
```
    ProteinParameter       AUC       CVE
1     Copeptin_Serum 0.9822222 0.5935176
2    Copeptin_Plasma 1.0000000 0.4061874
3      HnRNPA1_Serum 0.7066667 1.4303023
4     HnRNPA1_Plasma 0.6133333 1.4589186
5   Endostatin_Serum 0.8911111 1.0360013
6  Endostatin_Plasma 0.9155556 0.9151671
7    Myostatin_Serum 1.0000000 0.5204222
8   Myostatin_Plasma 0.9955556 0.5867828
9       etOhDG_Serum 0.9911111 0.5646229
10     etOhDG_Plasma 0.8355556 1.2158489
11       PARP1_Serum 0.6044444 1.5383728
12      PARP1_Plasma 0.5244444 1.5389460
```

## Data Visualization - Graphing Results

```
#Create a tidy dataframe of AUC results
diagnosis_auc <- tribble(
  ~Biomarker, ~Sample, ~AUC,
```

```
  "Copeptin", "Serum", 1,
  "Copeptin", "Plasma", 1,
  "HnRNPA1", "Serum", 0.9888889,
  "HnRNPA1", "Plasma", 0.9847222,
  "Endostatin", "Serum", 0.9222222,
  "Endostatin", "Plasma", 0.9513889,
  "Myostatin", "Serum", 0.75,
  "Myostatin", "Plasma", 0.7013889,
  "8-OhDG", "Serum", 0.9972222,
  "8-OhDG", "Plasma", 1,
  "PARP1", "Serum", 1,
  "PARP1", "Plasma", 1
)

prognosis_auc <- tribble(
  ~Biomarker, ~Sample, ~AUC,
  "Copeptin", "Serum", 0.9822222,
  "Copeptin", "Plasma", 1,
  "HnRNPA1", "Serum", 0.7066667,
  "HnRNPA1", "Plasma", 0.6133333,
  "Endostatin", "Serum", 0.8911111,
  "Endostatin", "Plasma", 0.9155556,
  "Myostatin", "Serum", 1,
  "Myostatin", "Plasma", 0.9955556,
  "8-OhDG", "Serum", 0.9911111,
  "8-OhDG", "Plasma", 0.8355556,
  "PARP1", "Serum", 0.6044444,
  "PARP1", "Plasma", 0.5244444
)

#Categorize AUC levels
auc_category <- function(x) {
  case_when(
    x >= 0.9 ~ "Excellent (≥ 0.9)",
    x >= 0.8 ~ "Good (0.8–0.9)",
    TRUE ~ "Moderate/Low (< 0.8)"
  )
}

#Category columns
diagnosis_auc <- diagnosis_auc %>%
  mutate(Category = auc_category(AUC),
         Group = "With vs W/o Chagas")

prognosis_auc <- prognosis_auc %>%
  mutate(Category = auc_category(AUC),
         Group = "Asym vs Sym")

#Combine
combined_auc <- bind_rows(diagnosis_auc, prognosis_auc)
```

```r
# Classification vector
classification_df <- tribble(
  ~Biomarker, ~Classification,
  "Copeptin", "Both",
  "HnRNPA1", "Diagnostic",
  "Endostatin", "Both",
  "Myostatin", "Prognostic",
  "8-OhDG", "Both",
  "PARP1", "Diagnostic"
)

# Merge classification into AUC data
combined_auc_labeled <- combined_auc %>%
  left_join(classification_df, by = "Biomarker")

# Plot with biomarker role
ggplot(combined_auc_labeled, aes(x = reorder(paste(Biomarker, Sample, sep = " "), AUC),
                                 y = AUC, fill = Classification)) +
  geom_col() +
  coord_flip() +
  facet_wrap(~Group, scales = "free_y") +
  scale_fill_manual(values = c("Prognostic" = "#2d0594FF",
                               "Diagnostic" = "#DD5E66FF",
                               "Both" = "#9512A1FF")) +
  geom_hline(yintercept = 0.8, linetype = "dashed", color = "black", linewidth = 0.7) +
  labs(title = "AUC Values for Biomarkers in Chagas Disease",
       subtitle = "Biomarkers classified as Diagnostic, Prognostic, or Both",
       x = "Biomarker",
       y = "AUC",
       fill = "Biomarker Role") +
    theme_minimal()
```




# Import Data

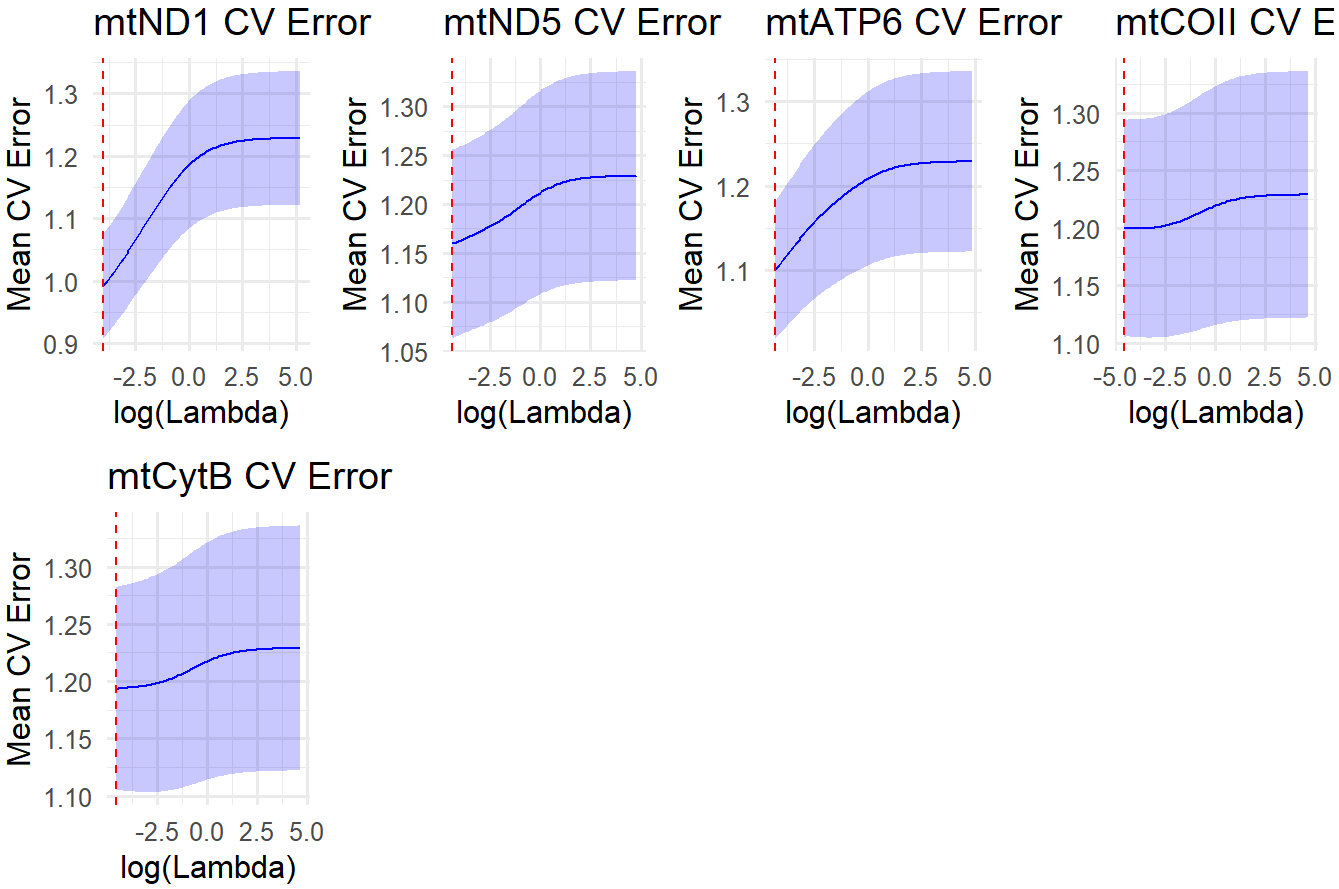
```r
mdata <- read_xlsx("~/R Notes/MitochondrialDNA.xlsx")

# "disease" column for asym,sym (1) and NHS (0)
mdata <- mdata %>%
  mutate(disease = case_when(
    Symptom == "NHS" ~ 0,
    Symptom == "Asymptomatic C" ~ 1,
    Symptom == "Symptomatic C" ~ 1
    )
  )

# "sym" column for asym (0) and sym (1)
mdata <- mdata %>%
  mutate(sym = case_when(
    Symptom == "Asymptomatic C" ~ 0,
    Symptom == "Symptomatic C" ~ 1
  ))

# These lists will be used for loops :D
```

```
list_DNA_Types <- c("mtND1", "mtND5", "mtATP6", "mtCOII", "mtCytB")
list_Group_Types <- c("NHS", "Asymptomatic C", "Symptomatic C")
```

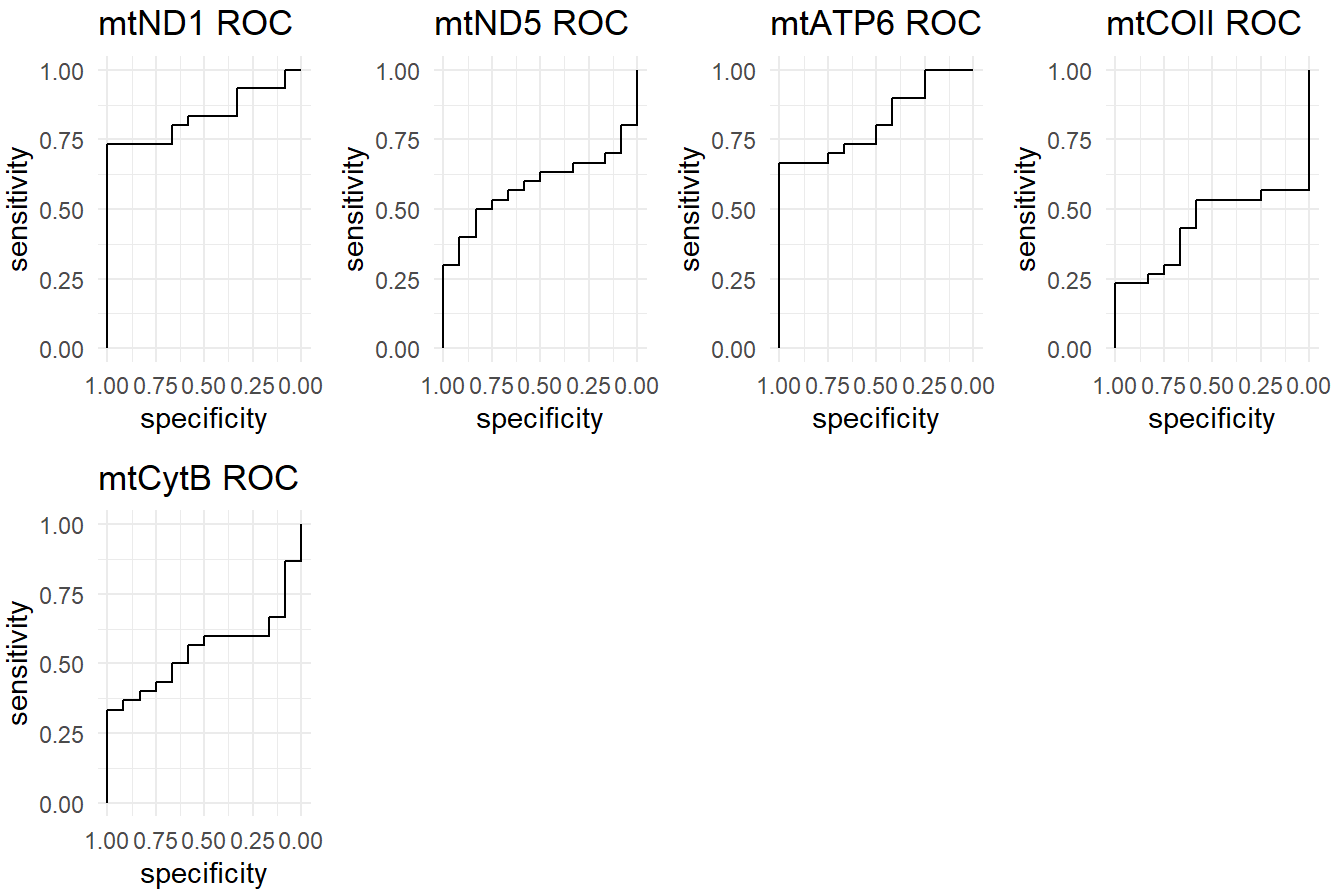
## Perform Logistic Regression to see if each Mitochondrial DNA Type can classify whether a patient has or does not have Chagas Disease.

```
# Empty lists to store ggplots
plot_list <- list()
cve_plot_list <- list()

# Create a summary table
summary_results <- data.frame(
  ProteinParameter = character(),
  AUC = numeric(),
  CVE = numeric(),
  stringsAsFactors = FALSE
)


for (DNA_Type in list_DNA_Types) {
  # --- ROC plots ---
  formula_text <- paste("disease ~", DNA_Type)
  mit_model <- glm(as.formula(formula_text), data = mdata, family = binomial)
  prob <- predict(mit_model, type = "response")
  roc_obj <- roc(mdata$disease, prob)

  roc_plot <- ggroc(roc_obj) +
    ggtitle(paste(DNA_Type, "ROC")) +
    theme_minimal()

  auc_value <- auc(roc_obj)
  #print(paste0(DNA_Type, " AUC: ", auc_value))
  plot_list[[DNA_Type]] <- roc_plot

  # --- CV error plots ---
  set.seed(67)
  x <- as.matrix(cbind(
  var   = mdata[[DNA_Type]],
  dummy = rnorm(nrow(mdata), 0, 1e-6)  # tiny noise column
))
y <- as.factor(mdata$disease)

cvfit <- cv.glmnet(x, y, family = "binomial", alpha = 0)

  cv_df <- data.frame(
    lambda = log(cvfit$lambda),
    cvm    = cvfit$cvm,
```

```
    cvsd   = cvfit$cvsd
  )

  # Extract CVE (at lambda.min)
  cve_value <- min(cvfit$cvm)

  cve_plot <- ggplot(cv_df, aes(x = lambda, y = cvm)) +
    geom_line(color = "blue") +
    geom_ribbon(aes(ymin = cvm - cvsd, ymax = cvm + cvsd),
                alpha = 0.2, fill = "blue") +
    geom_vline(xintercept = log(cvfit$lambda.min),
               linetype = "dashed", color = "red") +
    labs(title = paste(DNA_Type, "CV Error"),
         x = "log(Lambda)", y = "Mean CV Error") +
    theme_minimal(base_size = 12)

  cve_plot_list[[DNA_Type]] <- cve_plot

  # Save to summary table
  summary_results <- rbind(summary_results, data.frame(
    biomarker = DNA_Type,
    AUC = auc_value,
    CVE = cve_value
  ))
}


# Arrange plots
grid.arrange(grobs = cve_plot_list,
             top = textGrob("Cross-Validation Error by mtDNA Type",
                            gp = gpar(fontsize = 16, fontface = "bold")),
             ncol = 4)
```




```
grid.arrange(grobs = plot_list,
             top = textGrob("mtDNA Types Predict Diseased vs NHS",
                            gp = gpar(fontsize = 16, fontface = "bold")),
             ncol = 4)
```




```
print(summary_results)
```

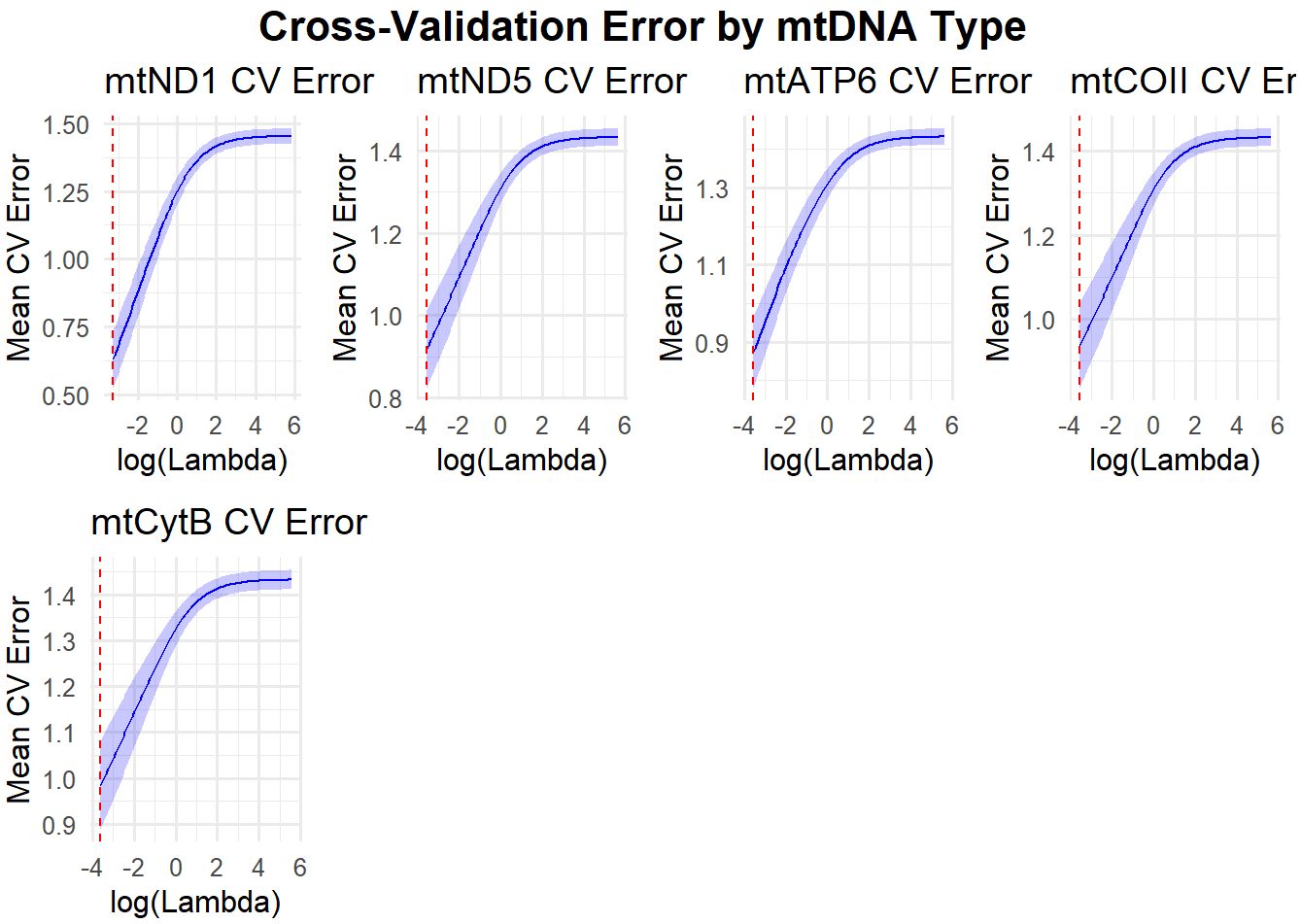
```
  biomarker       AUC       CVE
1     mtND1 0.8361111 0.9922597
2     mtND5 0.5833333 1.1596412
3    mtATP6 0.8138889 1.1010238
4    mtCOII 0.4416667 1.1996443
5    mtCytB 0.5444444 1.1934739
```

## Perform Logistic Regression to see if each Mitochondrial DNA Type can classify whether a patient is Symptomatic or Asymptomatic.

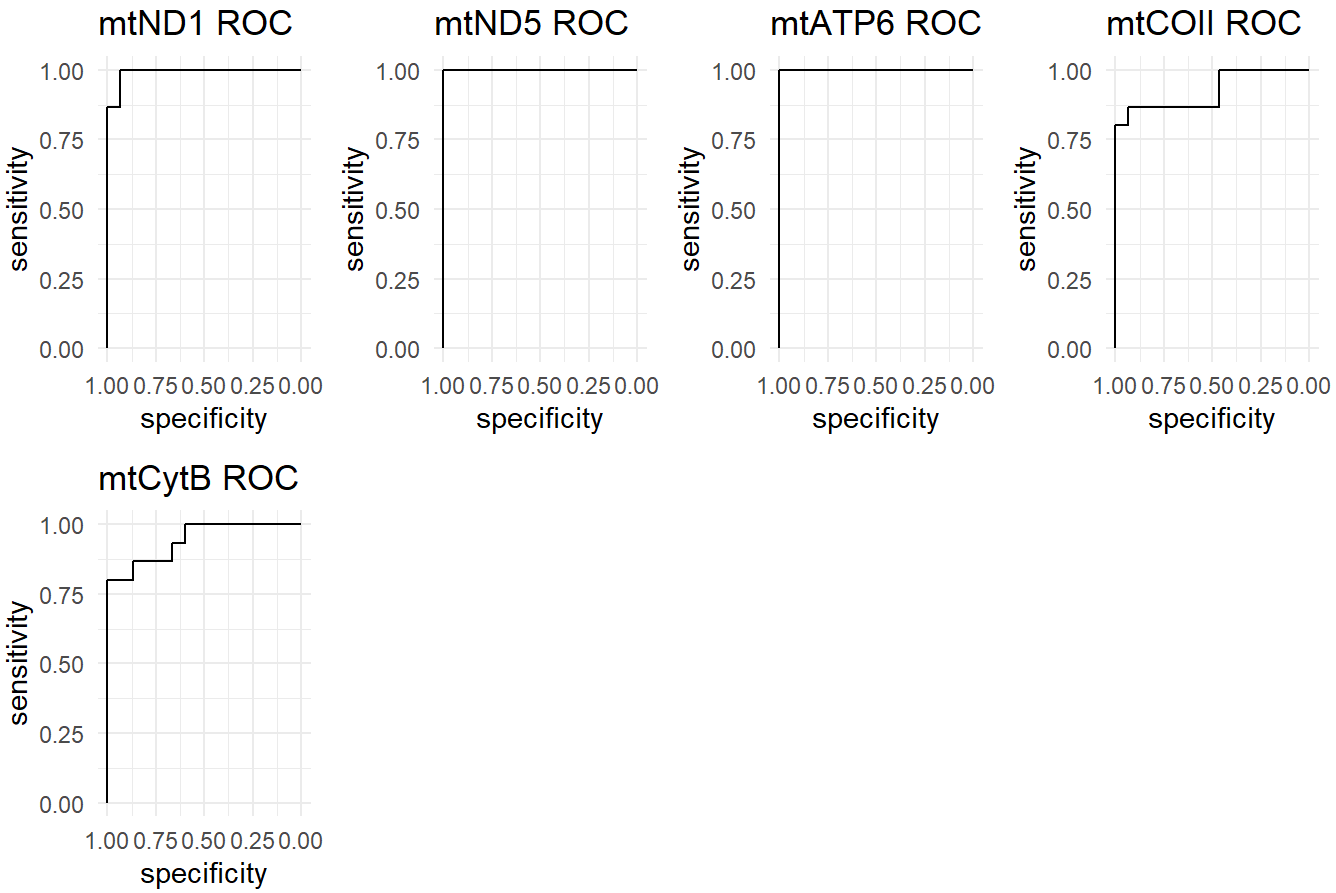
```
# Empty lists to store ggplots
plot_list <- list()
cve_plot_list <- list()

# Create a summary table
summary_results <- data.frame(
  ProteinParameter = character(),
  AUC = numeric(),
  CVE = numeric(),
```

```
  stringsAsFactors = FALSE
)

# Remove rows with NAs in the 'sym' column
asmdata <- mdata %>%
  filter(!is.na(sym))

for (DNA_Type in list_DNA_Types) {
  # Design matrix (with intercept)
  x <- model.matrix(as.formula(paste("~", DNA_Type)), data = asmdata)
  y <- as.factor(asmdata$sym)

  # Cross-validated glmnet model
  cvfit <- cv.glmnet(x, y, family = "binomial", alpha = 0)

  # --- ROC plots (use cv.glmnet fit) ---
  prob <- predict(cvfit, newx = x, s = "lambda.min", type = "response")
  roc_obj <- roc(asmdata$sym, as.numeric(prob))

  roc_plot <- ggroc(roc_obj) +
    ggtitle(paste(DNA_Type, "ROC")) +
    theme_minimal()

  auc_value <- auc(roc_obj)
  #print(paste0(DNA_Type, " AUC: ", auc_value))
  plot_list[[DNA_Type]] <- roc_plot

  # --- CV error plots ---
  set.seed(67)
  cv_df <- data.frame(
    lambda = log(cvfit$lambda),
    cvm    = cvfit$cvm,
    cvsd   = cvfit$cvsd
  )

  # Extract CVE (at lambda.min)
  cve_value <- min(cvfit$cvm)

  cve_plot <- ggplot(cv_df, aes(x = lambda, y = cvm)) +
    geom_line(color = "blue") +
    geom_ribbon(aes(ymin = cvm - cvsd, ymax = cvm + cvsd),
                alpha = 0.2, fill = "blue") +
    geom_vline(xintercept = log(cvfit$lambda.min),
               linetype = "dashed", color = "red") +
    labs(title = paste(DNA_Type, "CV Error"),
         x = "log(Lambda)", y = "Mean CV Error") +
    theme_minimal(base_size = 12)

  cve_plot_list[[DNA_Type]] <- cve_plot

  # Save to summary table
```

```
  summary_results <- rbind(summary_results, data.frame(
    biomarker = DNA_Type,
    AUC = auc_value,
    CVE = cve_value
  ))
}

# Arrange plots
grid.arrange(grobs = cve_plot_list,
             top = textGrob("Cross-Validation Error by mtDNA Type",
                            gp = gpar(fontsize = 16, fontface = "bold")),
             ncol = 4)
```




```
grid.arrange(grobs = plot_list,
             top = textGrob("DNA Types Predict Asymptomatic vs Symptomatic",
                            gp = gpar(fontsize = 16, fontface = "bold")),
             ncol = 4)
```




```
print(summary_results)
```

```
  biomarker       AUC       CVE
1     mtND1 0.9911111 0.6311849
2     mtND5 1.0000000 0.9183945
3    mtATP6 1.0000000 0.8741642
4    mtCOII 0.9244444 0.9374986
5    mtCytB 0.9422222 0.9852878
```

## Visualize Results

```
# AUC dataframe
auc_mtdata <- data.frame(
  Gene = c("MtND1", "MtND5", "MtATP6", "MtcCOII", "MtCytB"),
  Control_vs_Disease = c(0.836, 0.583, 0.813, 0.441, 0.544),
  Sym_vs_Asym = c(0.991, 1, 1, 0.924, 0.942)
)

# Pivot to long format
data_long <- pivot_longer(auc_mtdata,
                          cols = -Gene,
```

```
                          names_to = "Comparison",
                          values_to = "AUC")

# Classify role
gene_classification <- auc_mtdata %>%
  mutate(
    Role = case_when(
      Control_vs_Disease >= 0.8 & Sym_vs_Asym >= 0.8 ~ "Diagnostic &\n Prognostic",
      Control_vs_Disease >= 0.8 ~ "Diagnostic",
      Sym_vs_Asym >= 0.8 ~ "Prognostic",
      TRUE ~ "Neither"
    )
  )

# Reshape for ggplot
data_long <- pivot_longer(gene_classification,
                          cols = c(Control_vs_Disease, Sym_vs_Asym),
                          names_to = "Comparison",
                          values_to = "AUC") %>%
  mutate(
    ComparisonLabel = case_when(
      Comparison == "Control_vs_Disease" ~ "With vs W/o Chagas",
      Comparison == "Sym_vs_Asym" ~ "Asym vs Sym"
    )
  )

# Base plot
plot_main <- ggplot(data_long, aes(x = AUC, y = reorder(Gene, AUC), fill = Role)) +
  geom_col() +
  facet_wrap(~ComparisonLabel, scales = "free_y") +
  geom_vline(xintercept = 0.8, linetype = "dashed", color = "black", linewidth = 0.5) +
  scale_fill_manual(
    values = c(
      # Pink = "#DD5E66",
      # Blue = "#2d0594",
      # Purple = "#9512A1",
      "Diagnostic &\n Prognostic" = "#9512A1",
      "Diagnostic" = "#DD5E66",
      "Prognostic" = "#2d0594",
      "Neither" = "gray70"
    )
  ) +
  labs(
    title = "AUC Values for MtDNA Types in Chagas Disease",
    x = "AUC", y = "Mitochondrial DNA Type"
  ) +
  theme_minimal() +
  theme(legend.title = element_text(face = "bold")) +
  labs(fill = "Biomarker Role")
```

```
plot(plot_main)
```