Question 1

Since `biomarker-raw.csv` contains protein concentration levels across various samples, let's examine the distribution of a sample of these protein values.

```r
library(ggplot2)
library(reshape2)
```

```r
data <- read.csv("data2/biomarker-raw.csv")

# Convert the selected protein columns to numeric (skip non-numeric values)
proteins <- data[, 3:7]  # Select a sample of 5 protein columns
proteins <- data.frame(lapply(proteins, function(x) as.numeric(as.character(x))))
```

```
## Warning in FUN(X[[i]], ...): NAs introduced by coercion
## Warning in FUN(X[[i]], ...): NAs introduced by coercion
## Warning in FUN(X[[i]], ...): NAs introduced by coercion
## Warning in FUN(X[[i]], ...): NAs introduced by coercion
## Warning in FUN(X[[i]], ...): NAs introduced by coercion
```

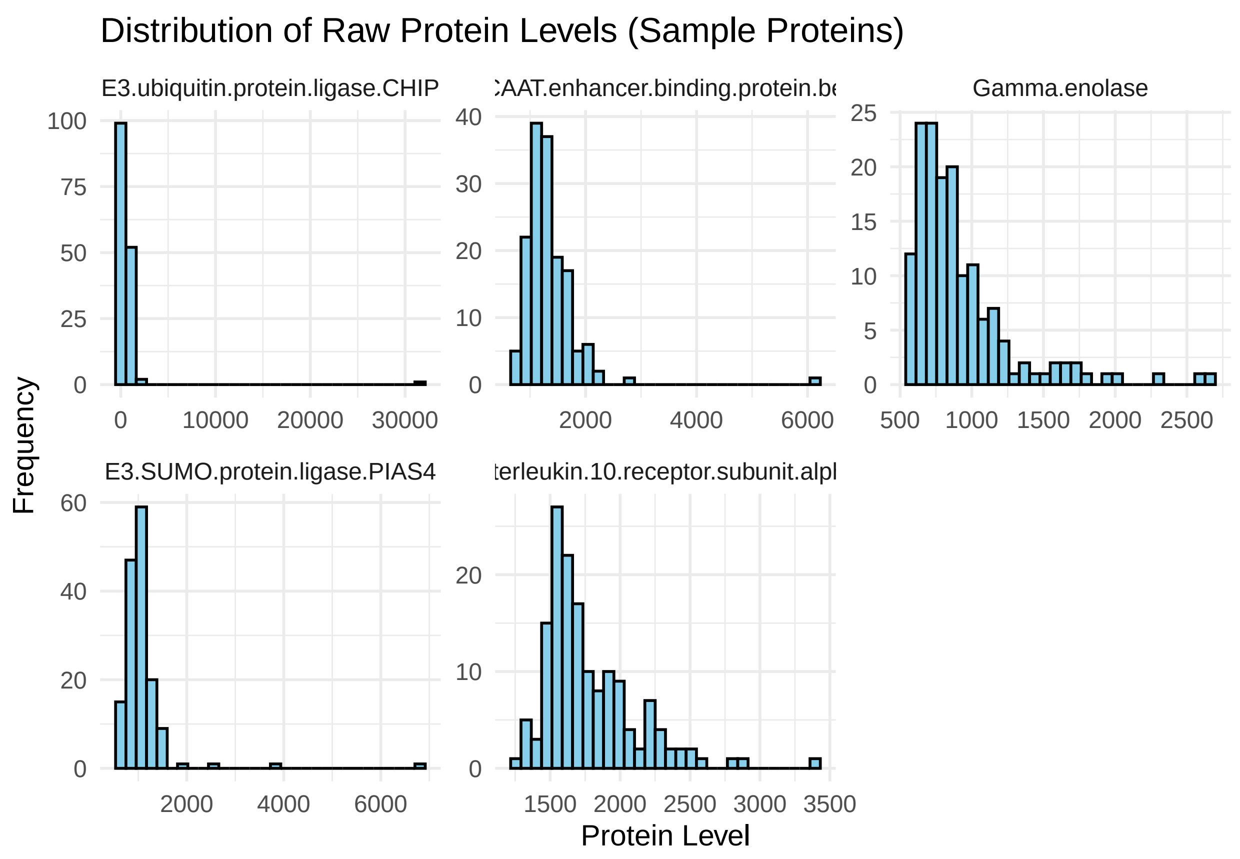
```r
# Melt the data for ggplot

proteins_long <- melt(proteins, variable.name = "Protein", value.name = "Level")
```

```
## No id variables; using all as measure variables
```

```r
# Plot histograms for each selected protein
ggplot(proteins_long, aes(x = Level)) +
  geom_histogram(bins = 30, color = "black", fill = "skyblue") +
  facet_wrap(~Protein, scales = "free") +
  labs(title = "Distribution of Raw Protein Levels (Sample Proteins)",
       x = "Protein Level", y = "Frequency") +
  theme_minimal()
```

```
## Warning: Removed 10 rows containing non-finite outside the scale range
## (‘stat_bin()‘).
```



The histograms reveal that the distributions of raw protein levels are skewed right, with some values extending to higher ranges. This skewness is a common reason to apply a log transformation, which can help normalize these distributions.

Question 2

```r
library(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
```

```r
library(tidyr)
```

```
##
## Attaching package: 'tidyr'
## The following object is masked from 'package:reshape2':
##
##     smiths
```

```r
# Load necessary libraries
library(dplyr)
library(tidyr)

# Load the data (assuming it's processed after removing outlier trimming)
data <- read.csv("data2/biomarker-raw.csv")

# Convert relevant columns to numeric, skipping metadata columns
protein_data <- data[, -c(1,2)]  # Adjust index based on the actual data structure

# Define a function to identify outliers based on ±3 standard deviations
identify_outliers <- function(x) {
  upper_limit <- mean(x, na.rm = TRUE) + 3 * sd(x, na.rm = TRUE)
  lower_limit <- mean(x, na.rm = TRUE) - 3 * sd(x, na.rm = TRUE)
  return(x < lower_limit | x > upper_limit)
}

# Apply the outlier function across all protein columns
outliers <- protein_data %>%
  mutate(across(everything(), identify_outliers)) %>%
  rowwise() %>%
  mutate(outlier_count = sum(c_across(everything()), na.rm = TRUE)) %>%
  ungroup()
```

```
## Warning: There were 5268 warnings in `mutate()`.
## The first warning was:
## i In argument: `across(everything(), identify_outliers)`.
## Caused by warning in `mean.default()`:
## ! argument is not numeric or logical: returning NA
## i Run `dplyr::last_dplyr_warnings()` to see the 5267 remaining warnings.
```

```r
# Merge with subject ID and group columns
outlier_summary <- data %>%
  select(Group) %>%
  bind_cols(outliers["outlier_count"])

# Summarize the number of outlying values per subject and by group
outlier_by_group <- outlier_summary %>%
  group_by(Group) %>%
  summarise(avg_outliers = mean(outlier_count, na.rm = TRUE),
            total_outliers = sum(outlier_count, na.rm = TRUE),
            subjects_with_outliers = sum(outlier_count > 0))

# View results
print(outlier_by_group)
```

```
## # A tibble: 3 x 4
##   Group avg_outliers total_outliers subjects_with_outliers
##   <chr>        <dbl>          <int>                  <int>
## 1 ""          0                    0                       0
## 2 "ASD"       0.0263               2                       2
## 3 "TD"        0                    0                       0
```

Are there specific subjects (not values) that seem to be outliers? Yes, there are specific subjects with outliers. According to the table of nsummary, there are 2 subjects with outlying values in the ASD group.

Are outliers more frequent in one group or the other? Outliers are more frequent in the ASD group. ASD group with a total of 2 outliers, while the TD group has none.

Question 3

```r
library(dplyr)
library(caret)  #data partition
```

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

```r
library(dplyr)  #ttest
library(purrr)  #ttest
```

```
##
## Attaching package: 'purrr'

## The following object is masked from 'package:caret':
##
##     lift
```

```r
library(broom)  #ttest
```

```
## Warning: package 'broom' was built under R version 4.4.1
```

```r
library(tidyverse) #ttest
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v readr     2.1.5

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(rstatix)
```

```
##
## Attaching package: 'rstatix'
##
## The following object is masked _by_ '.GlobalEnv':
##
##     identify_outliers
##
## The following object is masked from 'package:stats':
##
##     filter
```

```r
library(randomForest) #Random Forest
```

```
## Warning: package 'randomForest' was built under R version 4.4.1
```

```
## randomForest 4.7-1.2
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(MASS)   #Logistic Regression
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:rstatix':
##
##     select
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
load('data2/biomarker-clean.RData')
head(biomarker_clean)
```

```
## # A tibble: 6 x 1,319
##   group  ados   CHIP  CEBPB    NSE  PIAS4 `IL-10 Ra`  STAT3   IRF1 `c-Jun`
##   <chr> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>      <dbl>  <dbl>  <dbl>   <dbl>
## 1 ASD       8  0.335  0.520 -0.554  0.650     -0.358  0.305 -0.484   0.309
## 2 ASD      21 -0.0715  1.01   3       1.28       -0.133  1.13   0.253   0.408
## 3 ASD      12 -0.406  -0.531 -0.0592  1.13        0.554 -0.334  0.287  -0.845
## 4 ASD      20 -0.102  -0.251  1.47    0.0773     -0.705  0.893  2.61   -0.372
## 5 ASD      22 -0.395  -0.536  0.0410 -0.299      -0.830  0.899  1.01   -0.843
## 6 ASD      17 -0.126   1.27  -0.892   0.239      -0.344  0.216  0.211   0.221
## # i 1,309 more variables: `Mcl-1` <dbl>, OAS1 <dbl>, `c-Myc` <dbl>,
## #   SMAD3 <dbl>, SMAD2 <dbl>, `IL-23` <dbl>, PDGFRA <dbl>, `IL-12` <dbl>,
## #   STAT1 <dbl>, STAT6 <dbl>, LRRK2 <dbl>, Osteocalcin <dbl>, `IL-5` <dbl>,
## #   GPDA <dbl>, IgA <dbl>, LPPL <dbl>, HEMK2 <dbl>, PDXK <dbl>, TLR4 <dbl>,
## #   REG4 <dbl>, `HSP 27` <dbl>, `YKL-40` <dbl>, `Alpha enolase` <dbl>,
## #   `Apo L1` <dbl>, CD38 <dbl>, CD59 <dbl>, FABPL <dbl>, `GDF-11` <dbl>,
## #   BTC <dbl>, `HIF-1a` <dbl>, S100A6 <dbl>, SECTM1 <dbl>, RSPO3 <dbl>, ...
```

```
set.seed(123)

# Split training (80%) and testing (20%) sets
trainIndex <- createDataPartition(biomarker_clean$group, p = 0.8, list = FALSE)
training_data <- biomarker_clean[trainIndex, ]
testing_data <- biomarker_clean[-trainIndex, ]
```

```
head(training_data)
```

Split Training and Testing

```
## # A tibble: 6 x 1,319
##   group  ados   CHIP  CEBPB     NSE  PIAS4 `IL-10 Ra`   STAT3    IRF1 `c-Jun`
##   <chr> <dbl>  <dbl>  <dbl>   <dbl>  <dbl>      <dbl>   <dbl>   <dbl>   <dbl>
## 1 ASD       8  0.335  0.520 -0.554   0.650     -0.358  0.305  -0.484    0.309
## 2 ASD      22 -0.395 -0.536  0.0410 -0.299     -0.830  0.899   1.01    -0.843
## 3 ASD      17 -0.126  1.27  -0.892   0.239     -0.344  0.216   0.211    0.221
## 4 ASD      15  0.486  0.748 -1.09    0.462      0.570 -0.0682  1.01     1.21
## 5 ASD      10 -0.990 -1.10   0.231  -0.885     -0.151  0.0307 -0.0346  -0.891
## 6 ASD      22 -0.108  3      2.32    3          2.76   1.70    0.209    3
## # i 1,309 more variables: `Mcl-1` <dbl>, OAS1 <dbl>, `c-Myc` <dbl>,
## #   SMAD3 <dbl>, SMAD2 <dbl>, `IL-23` <dbl>, PDGFRA <dbl>, `IL-12` <dbl>,
## #   STAT1 <dbl>, STAT6 <dbl>, LRRK2 <dbl>, Osteocalcin <dbl>, `IL-5` <dbl>,
## #   GPDA <dbl>, IgA <dbl>, LPPL <dbl>, HEMK2 <dbl>, PDXK <dbl>, TLR4 <dbl>,
## #   REG4 <dbl>, `HSP 27` <dbl>, `YKL-40` <dbl>, `Alpha enolase` <dbl>,
## #   `Apo L1` <dbl>, CD38 <dbl>, CD59 <dbl>, FABPL <dbl>, `GDF-11` <dbl>,
## #   BTC <dbl>, `HIF-1a` <dbl>, S100A6 <dbl>, SECTM1 <dbl>, RSPO3 <dbl>, ...
```

```
head(testing_data)
```

```
## # A tibble: 6 x 1,319
##   group  ados    CHIP   CEBPB     NSE   PIAS4 `IL-10 Ra`  STAT3    IRF1 `c-Jun`
##   <chr> <dbl>   <dbl>   <dbl>   <dbl>   <dbl>      <dbl>  <dbl>   <dbl>   <dbl>
## 1 ASD      21 -0.0715  1.01    3       1.28       -0.133  1.13    0.253    0.408
## 2 ASD      12 -0.406  -0.531  -0.0592  1.13        0.554 -0.334   0.287   -0.845
## 3 ASD      20 -0.102  -0.251   1.47    0.0773     -0.705  0.893  2.61    -0.372
## 4 ASD      14 -0.378  -0.0790 -0.727   0.814      -0.811 -0.406 -0.791   -0.647
## 5 ASD      17  0.214   1.85    2.17    2.19       -0.102 -0.551 -0.293    1.80
## 6 ASD      13  1.35   -0.947  -1.28   -0.931      -0.443 -1.32   0.0259  -0.445
## # i 1,309 more variables: `Mcl-1` <dbl>, OAS1 <dbl>, `c-Myc` <dbl>,
## #   SMAD3 <dbl>, SMAD2 <dbl>, `IL-23` <dbl>, PDGFRA <dbl>, `IL-12` <dbl>,
## #   STAT1 <dbl>, STAT6 <dbl>, LRRK2 <dbl>, Osteocalcin <dbl>, `IL-5` <dbl>,
## #   GPDA <dbl>, IgA <dbl>, LPPL <dbl>, HEMK2 <dbl>, PDXK <dbl>, TLR4 <dbl>,
## #   REG4 <dbl>, `HSP 27` <dbl>, `YKL-40` <dbl>, `Alpha enolase` <dbl>,
## #   `Apo L1` <dbl>, CD38 <dbl>, CD59 <dbl>, FABPL <dbl>, `GDF-11` <dbl>,
## #   BTC <dbl>, `HIF-1a` <dbl>, S100A6 <dbl>, SECTM1 <dbl>, RSPO3 <dbl>, ...
```

Apply T-test, Random Forest, and Logistic Regression (Using Top 20 Features)

```r
library(dplyr)

# Ensure correct names and explicit call to dplyr::select
group_column <- training_data$group  # Confirm column name is correct
protein_columns <- dplyr::select(training_data, -group, -ados)  # Explicitly use dplyr



#T test for all protein
t_test_results <- sapply(protein_columns, function(protein) {
  t.test(protein ~ group_column)$p.value
})

# Convert result into frame
t_test_df <- data.frame(
  protein = names(t_test_results),
  p_value = t_test_results
)

# Extract top 20 proteins
top_proteins_ttest <- t_test_df %>%
  arrange(p_value) %>%
  slice(1:20) %>%
  pull(protein)


print(top_proteins_ttest)
```

T-test Selection

```
##  [1] "PTN"                         "RELT"
##  [3] "MAPK2"                       "DERM"
##  [5] "Calcineurin"                 "M2-PK"
##  [7] "TFF3"                        "FSTL1"
##  [9] "CXCL16, soluble"             "MAPK14"
## [11] "Coagulation Factor IX"       "IgD"
## [13] "MIA"                        "Fas, soluble"
## [15] "MMP-2"                      "IGFBP-4"
## [17] "ROR1"                       "Protein S"
## [19] "14-3-3 protein zeta/delta" "TGF-b R III"
```

```
library(randomForest)
library(dplyr)


# Alternative approach if select() conflicts remain
predictors <- training_data[, !names(training_data) %in% c("group", "ados")]
response <- factor(training_data$group)



set.seed(101422)


rf_model <- randomForest(x = predictors,
                         y = response,
                         ntree = 1000,
                         importance = TRUE)


print(rf_model$confusion)
```

Random Forest

```
##     ASD TD class.error
## ASD  35 26   0.4262295
## TD   13 50   0.2063492
```

```
important_proteins_rf <- rf_model$importance %>%
  as_tibble() %>%
  mutate(protein = rownames(rf_model$importance)) %>%
  slice_max(MeanDecreaseGini, n = 20) %>%  # top 20
  pull(protein)


print(important_proteins_rf)
```

```
##  [1] "eIF-4H"                 "MAPK2"                  "PTN"
##  [4] "CSK"                    "MAPK14"                 "M2-PK"
##  [7] "DERM"                   "Lysozyme"               "RELT"
## [10] "ILT-4"                  "CD27"                   "Nectin-like protein 2"
## [13] "Coagulation Factor IX" "Calcineurin"            "Notch 1"
## [16] "IgD"                    "IGFBP-1"                "SOST"
## [19] "GPVI"                   "MMP-2"
```

```r
library(caret)
library(dplyr)



train_data_for_model <- training_data %>%
  dplyr::select(-ados) %>%
  dplyr::mutate(group = as.factor(group))



train_control <- trainControl(method = "none")


logistic_model <- train(group ~ .,
                        data = train_data_for_model,
                        method = "glm",
                        family = "binomial",
                        trControl = train_control)
```

Logistic Regression

```
## Warning: glm.fit: algorithm did not converge
```

```r
logistic_coefficients <- coef(logistic_model$finalModel)


coeff_df <- as.data.frame(logistic_coefficients) %>%
  rownames_to_column(var = "protein") %>%
  filter(protein != "(Intercept)") %>%
  rename(coefficient = logistic_coefficients) %>%
  mutate(abs_coefficient = abs(coefficient))


top_20_proteins <- coeff_df %>%
  arrange(desc(abs_coefficient)) %>%
  slice(1:20) %>%
  pull(protein)


print(top_20_proteins)
```

```
##  [1] "SMAD3"                                  "HXK1"
##  [3] "Myostatin"                              "‘\\‘HIF-1a\\‘‘"
##  [5] "CHKB"                                   "ISLR2"
##  [7] "CSH"                                    "HHLA2"
##  [9] "RNF43"                                  "OAS1"
## [11] "‘\\‘TIMP-1\\‘‘"                         "CD59"
## [13] "SMOC1"                                  "‘\\‘14-3-3 protein beta/alpha\\‘‘"
## [15] "SNP25"                                  "LDLR"
## [17] "S100A4"                                 "EFNB1"
## [19] "EFNB2"                                  "STAT6"
```

```r
all_top_proteins <- list(
  ttest = top_proteins_ttest,
  rf = important_proteins_rf,
  logreg = top_20_proteins
)


fuzzy_intersection_proteins <- all_top_proteins %>%
  unlist() %>%
  table() %>%
  .[. >= 2] %>%
  names()

print(fuzzy_intersection_proteins)
```

Fuzzy Interaction

```
##  [1] "Calcineurin"           "Coagulation Factor IX" "DERM"
##  [4] "IgD"                   "M2-PK"                 "MAPK14"
##  [7] "MAPK2"                 "MMP-2"                 "PTN"
## [10] "RELT"
```

```r
# Evaluate each set of selected proteins on testing data
evaluate_model_accuracy <- function(selected_proteins, model_name) {
  # Explicitly use dplyr::select() and tidyselect::all_of()
  predictors <- dplyr::select(testing_data, tidyselect::all_of(selected_proteins))
  response <- factor(testing_data$group)

  rf_test <- randomForest(x = predictors, y = response, ntree = 100)
  accuracy <- sum(diag(rf_test$confusion)) / sum(rf_test$confusion)

  cat("Accuracy of", model_name, "model on testing data:", accuracy, "\n")
}

# Evaluate individual methods
evaluate_model_accuracy(top_proteins_ttest, "T-test")
```

How are results affected by each modification?

```
## Accuracy of T-test model on testing data: 0.720524
```

```r
evaluate_model_accuracy(important_proteins_rf, "Random Forest")
```

```
## Accuracy of Random Forest model on testing data: 0.7894737
```

```r
#evaluate_model_accuracy(top_20_proteins, "Logistic Regression")

# Evaluate fuzzy intersection
evaluate_model_accuracy(fuzzy_intersection_proteins, "Fuzzy Intersection")
```

```
## Accuracy of Fuzzy Intersection model on testing data: 0.8241758
```

Each method yields different predictive performance, with Random Forest and the Fuzzy Intersection performing similarly well. The fuzzy intersection can be a useful compromise when seeking a balance between different selection criteria, but in this case, it does not significantly outperform Random Forest alone.