Analysis-main

Lucas Childs

2025-11-06

1. What is the reason for log transforming protein levels in `biomarker-raw.csv`?

```
set.seed(1234)
library(tidyverse)
library(here)
rawdata <- read.csv(here("data", "biomarker-raw.csv"))

# random sample of 4 proteins to look at distributions of their levels
rand_indices <- sample(3:ncol(rawdata), 4)

prot1d <- as.numeric(rawdata[2:nrow(rawdata), rand_indices[1]])
prot1 <- prot1d[!is.na(prot1d)]

prot2d <- as.numeric(rawdata[2:nrow(rawdata), rand_indices[2]])
prot2 <- prot2d[!is.na(prot2d)]

prot3d <- as.numeric(rawdata[2:nrow(rawdata), rand_indices[3]])
prot3 <- prot3d[!is.na(prot3d)]

prot4d <- as.numeric(rawdata[2:nrow(rawdata), rand_indices[4]])
prot4 <- prot4d[!is.na(prot4d)]

summary(prot1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2568    4649    5169    5243    5668    7435
```

```
summary(prot2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3317    8790   12179   17164   18800  122168
```

```
summary(prot3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   283.1   378.7   414.2   457.8   496.9  1894.7
```

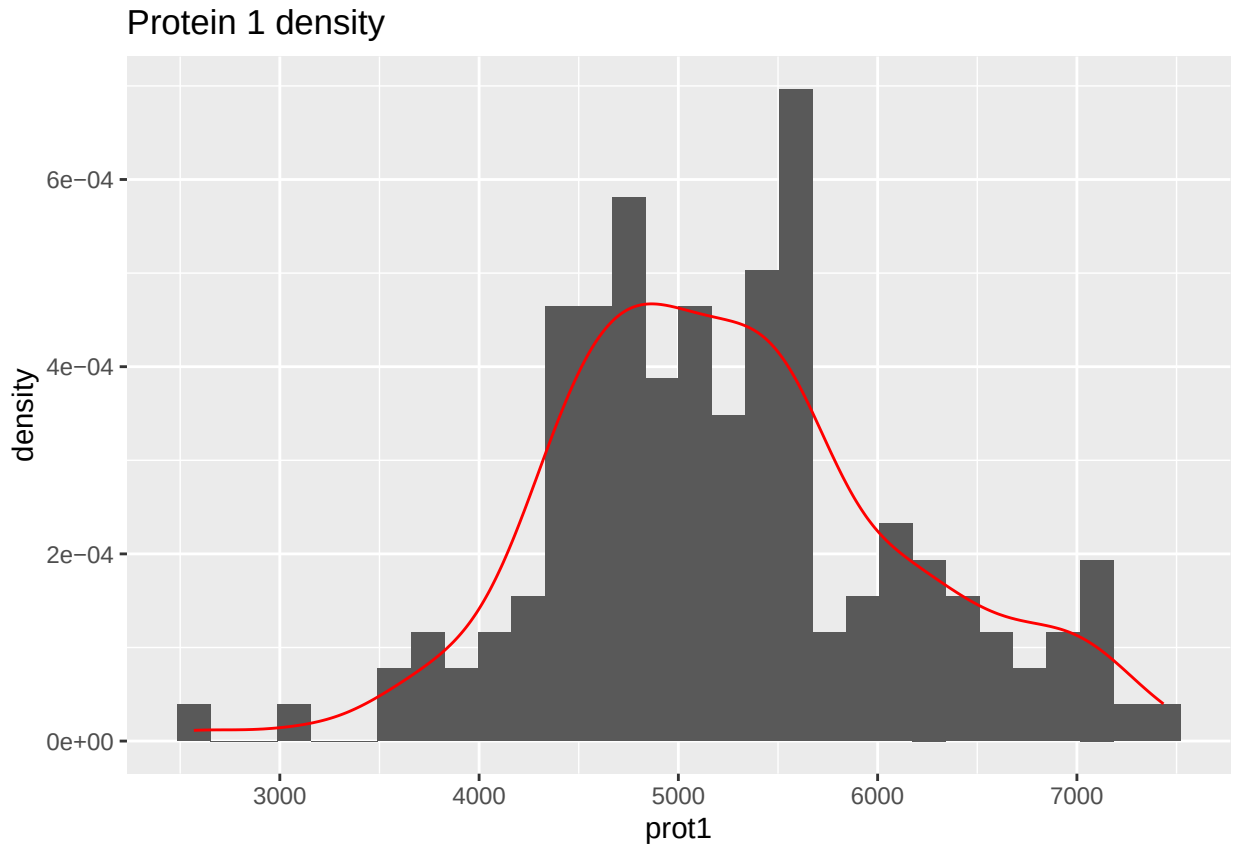
```
summary(prot4)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   954.9  5434.7  6539.2  6551.7  7719.1 10917.8
```

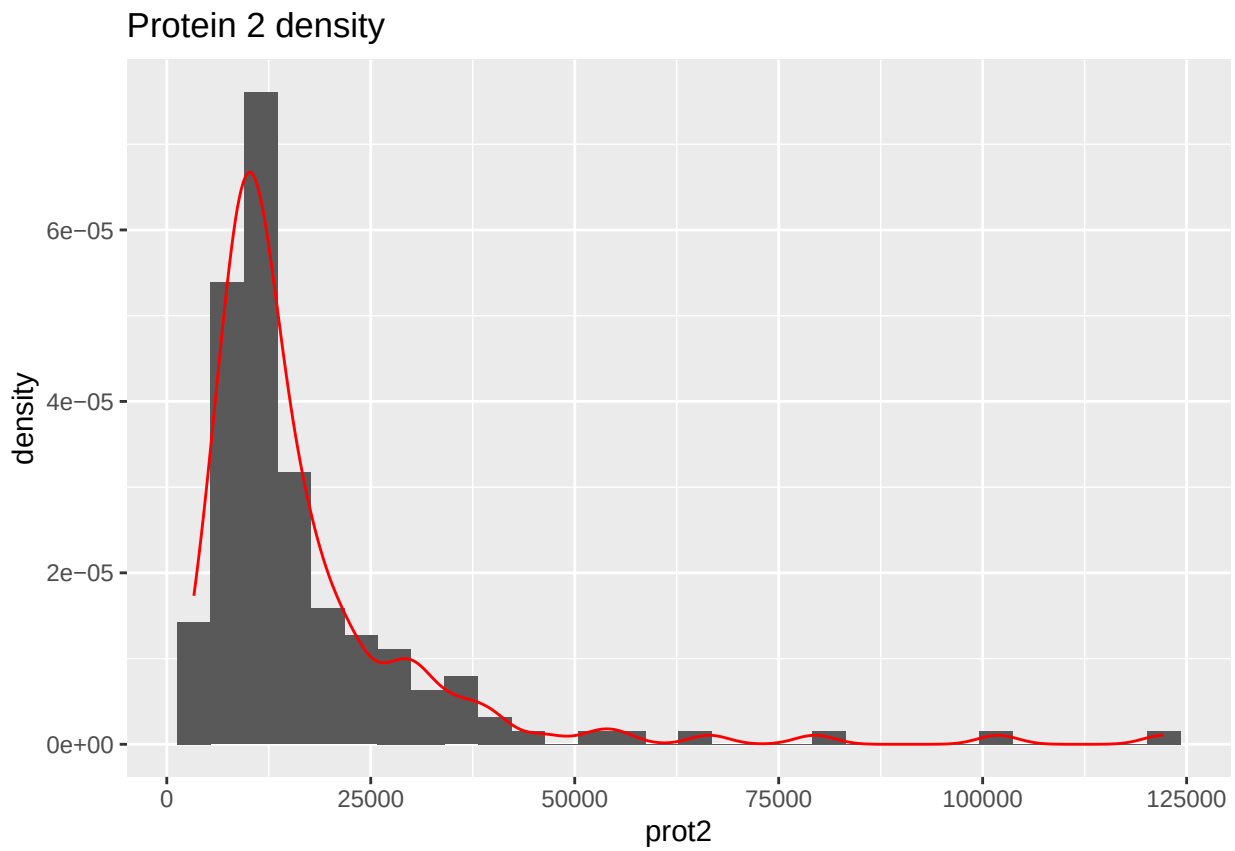
Looking at the summary statistics for 4 randomly selected proteins, the mean values differ significantly, with `prot1`'s mean being 17,164 and `prot3`'s mean being ≈ 458. `prot2`'s mean is roughly 5000 units greater than its median as well, indicating a right skew.

The log transformation of the protein levels helps compress the scale of protein levels since we have a wide range of positive values, where some are very large. Additionally, the logarithm helps with skewed data, and we have evidence that some of the data is skewed, since `prot2` has a mean much larger than its median.

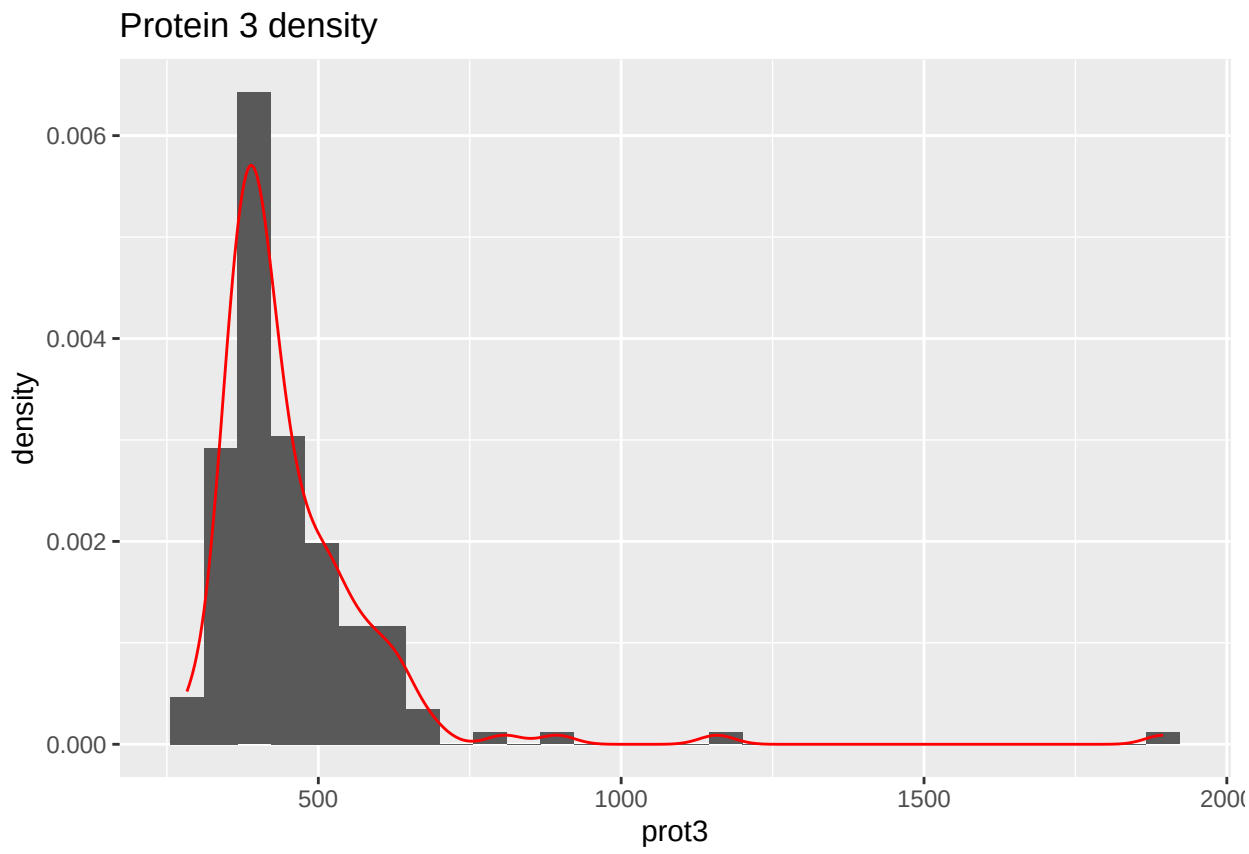
```
library(ggplot2)
ggplot(as.data.frame(prot1), aes(x = prot1)) +
  geom_histogram(aes(y=after_stat(density)), bins = 30) +
  geom_density(color="red") +
  ggtitle('Protein 1 density')
```



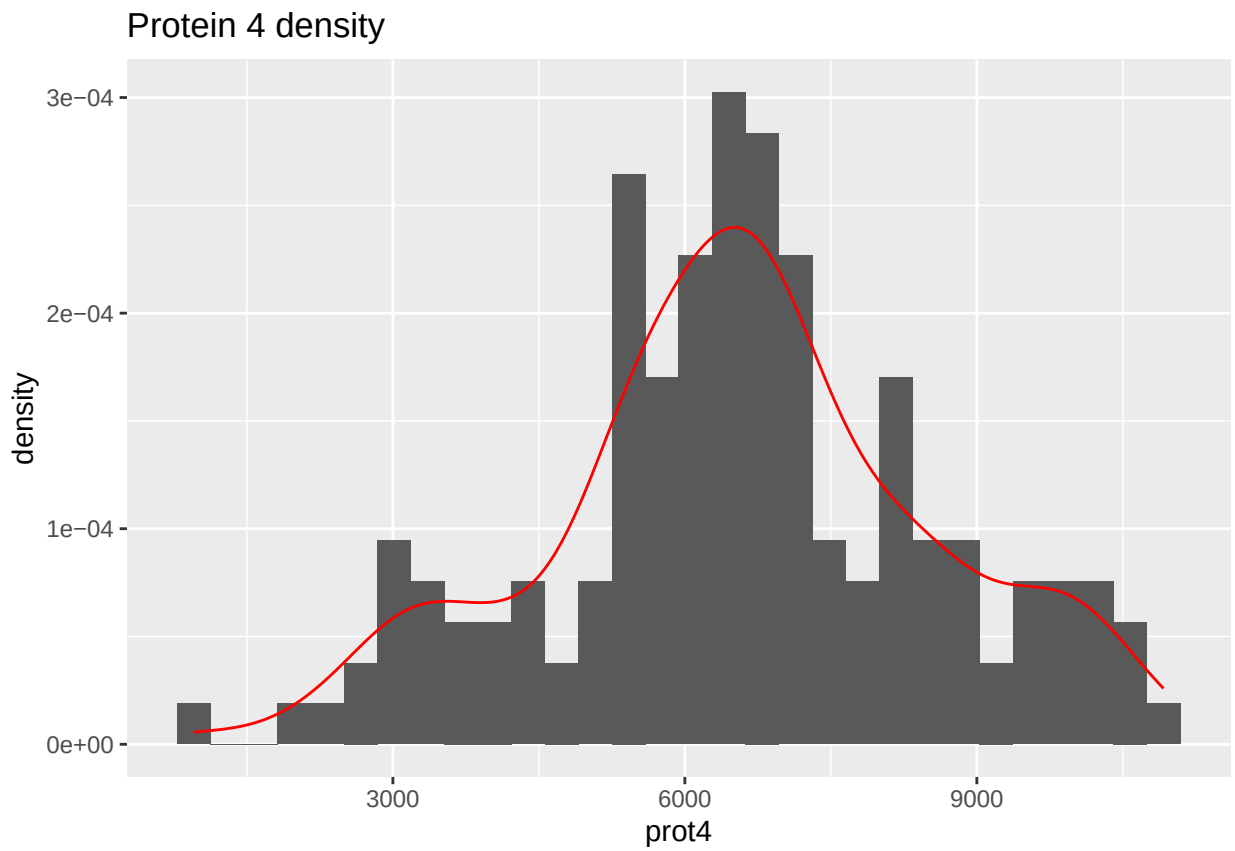

```
ggplot(as.data.frame(prot2), aes(x = prot2)) +
  geom_histogram(aes(y=after_stat(density)), bins = 30) +
  geom_density(color="red") +
  ggtitle('Protein 2 density')
```




```
ggplot(as.data.frame(prot3), aes(x = prot3)) +
  geom_histogram(aes(y=after_stat(density)), bins = 30) +
  geom_density(color="red") +
  ggtitle('Protein 3 density')
```



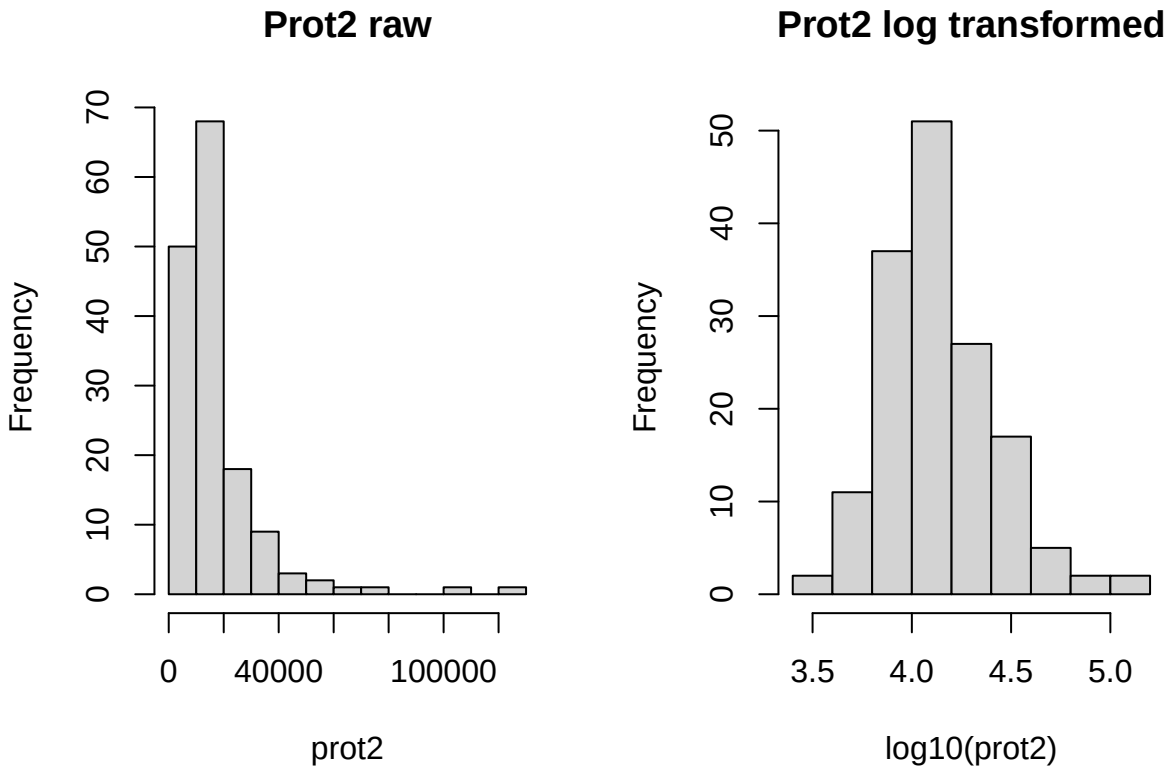

```
ggplot(as.data.frame(prot4), aes(x = prot4)) +
  geom_histogram(aes(y=after_stat(density)), bins = 30) +
  geom_density(color="red") +
  ggtitle('Protein 4 density')
```



From the density plots, `prot2` looks the most skewed (strongly right-skewed) `prot3` looks slightly right-skewed as well. Both proteins contain large outliers, however the scale of `prot2`'s protein level is much higher, so the same follows for its outliers.

Comparison of `prot2` histogram to its log transformed counterpart:

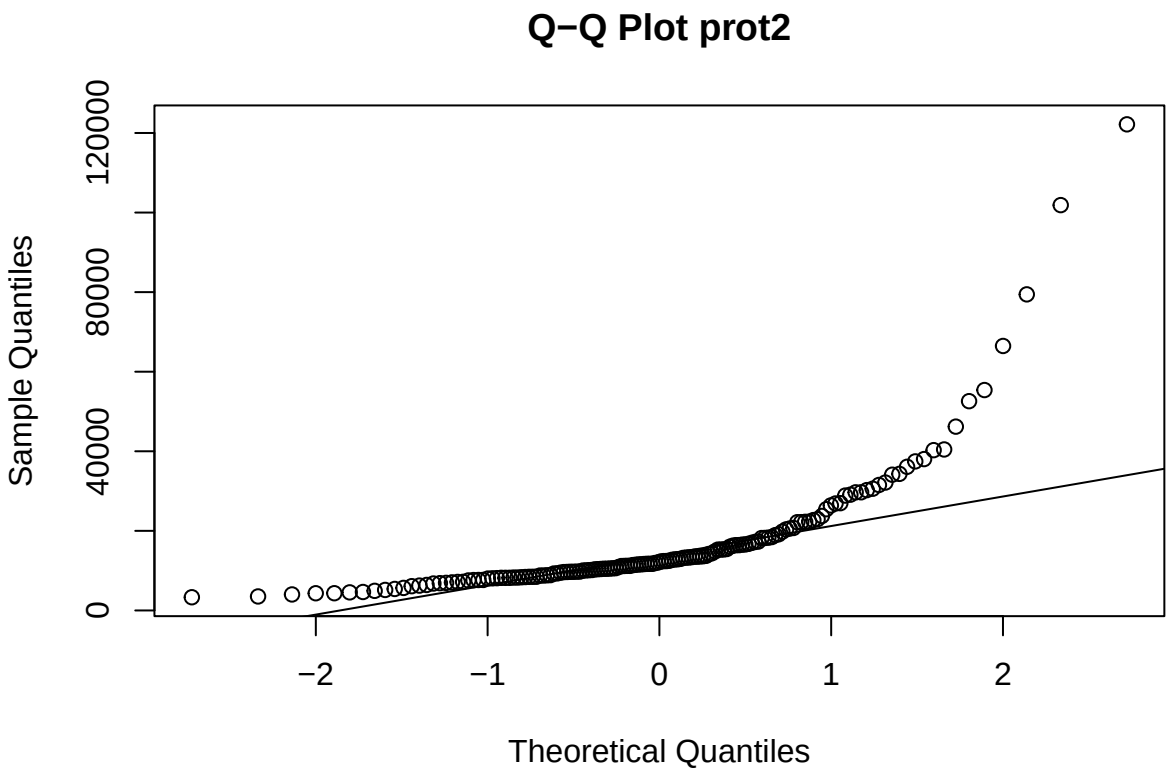
```
par(mfrow=c(1,2))
hist(prot2, main="Prot2 raw")
hist(log10(prot2), main="Prot2 log transformed")
```



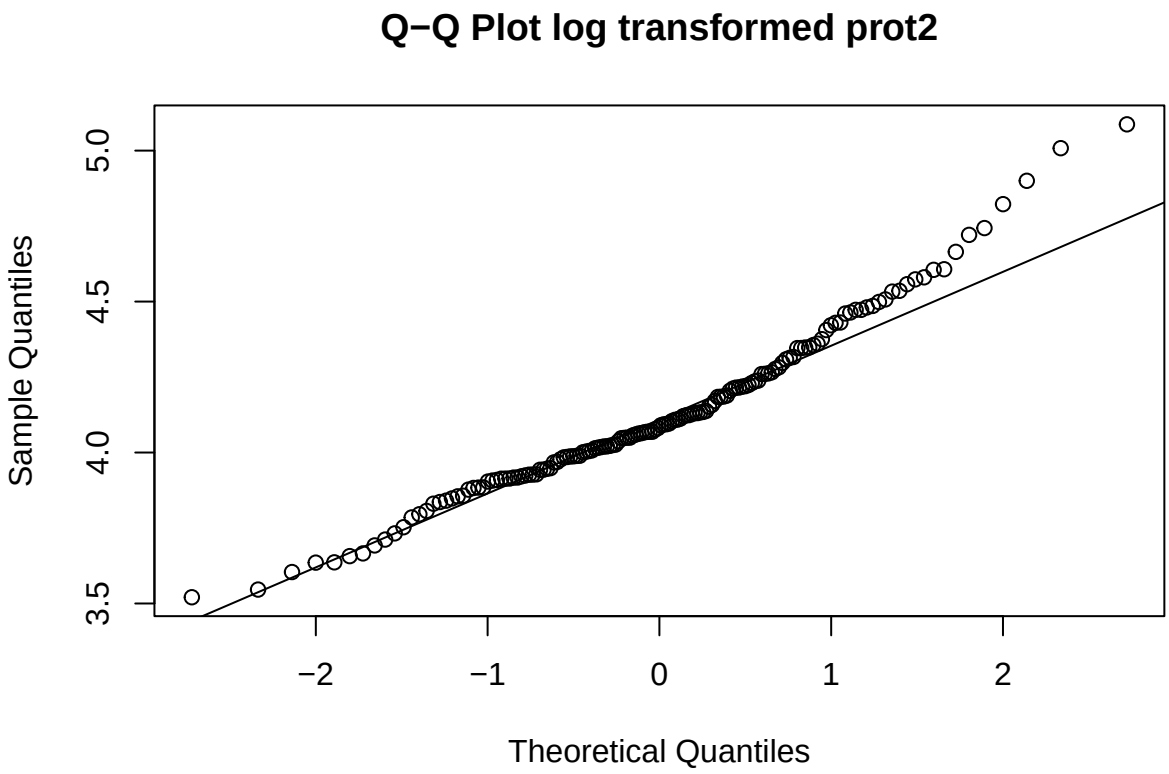
As we can see, after log transforming, the right-skewed `prot2` now appears more symmetric and of a much smaller and more readable scale.

We can check more rigorously to see how normality differs between the raw and log transformed `prot2` with a QQ-Plot.

```
# normality check for raw data
qqnorm(prot2, main = 'Q-Q Plot prot2')
qqline(prot2)
```




```
# normality check for log transformed data
qqnorm(log10(prot2), main = 'Q-Q Plot log transformed prot2')
qqline(log10(prot2))
```



After transforming `prot2`, the protein values appear slightly more normal, reducing the influential outliers. Overall, log transforming the protein levels acted as a way to reduce the large scale of the values, and make the data more symmetric.

2. Temporarily remove the outlier trimming from preprocessing and do some exploratory analysis of the outlying values. Are there specific subjects (not values) that seem to be outliers? If so, are outliers more frequent in one group or the other?

——————————-ANNA ADD SECTION——————————-

3. Repeat the analysis but carry out the entire selection procedure on a training partition – in other words, set aside some testing data at the very beginning and don't use it until you are evaluating accuracy at the very end

Choose a larger number (more than ten) of top predictive proteins using each selection method

Use a fuzzy intersection instead of a hard intersection to combine the sets of top predictive proteins across selection methods

——————————-Nathan/Minu SECTION——————————-

4. Find an alternative panel that achieves improved classification accuracy

Benchmark your results against the in-class analysis.

Goal: Explore alternative feature selection

```
# Load libraries
library(tidyverse)
library(caret)
library(glmnet)
library(pROC)
library(yardstick)
```

```
# 1. Load processed data
load(here('data', 'biomarker-clean.RData'))

# Rename
biomarker <- biomarker_clean

# Convert group to factor (TD = control group)
biomarker$group <- factor(biomarker$group, levels = c("TD", "ASD"))
```

```
# 2. Train/Test Split (80% train, 20% test)

set.seed(123)
train_idx <- createDataPartition(biomarker$group,
                                 p = 0.8, list = FALSE)

train <- biomarker[train_idx, ]
test <- biomarker[-train_idx, ]

# Prepare matrices for glmnet
x_train <- as.matrix(train %>% select(-group, -ados))
y_train <- train$group

x_test <- as.matrix(test %>% select(-group, -ados))
y_test <- test$group
```

```
# 3. LASSO Classification and Feature Selection

set.seed(123)
lasso_fit <- cv.glmnet(
  x_train, y_train,
  alpha = 1, # LASSO penalty
  family = "binomial"
)

# Extract non-zero coefficients (The Alternative Panel)
lasso_coef <- coef(lasso_fit, s = "lambda.min")
lasso_features <- rownames(lasso_coef)[lasso_coef[, 1] != 0]
lasso_features <- lasso_features[lasso_features != "(Intercept)"]

cat("\nQ4 Alternative Panel (Pure LASSO features):\n")
```

```
##
## Q4 Alternative Panel (Pure LASSO features):
```

```
print(lasso_features)
```

```
##  [1] "IgA"                          "CD59"
##  [3] "FAM3D"                        "FSTL1"
##  [5] "CXCL16, soluble"              "Macrophage mannose receptor"
##  [7] "P-Cadherin"                   "Protein S"
##  [9] "IGFBP-1"                      "LAG-1"
## [11] "Kallikrein 11"                "Met"
## [13] "Growth hormone receptor"      "ESAM"
## [15] "Siglec-3"                     "FCN1"
## [17] "HGFA"                         "a2-Macroglobulin"
## [19] "MAPK2"                        "IL-6 sRa"
## [21] "ENPP7"                        "ENTP5"
## [23] "MFGM"                         "PCSK7"
## [25] "PERL"                         "ITI heavy chain H4"
## [27] "Calcineurin"                  "IgD"
## [29] "DERM"                         "hnRNP K"
## [31] "ILT-4"                        "RELT"
## [33] "SIG14"                        "TWEAKR"
## [35] "PPID"                         "PSMA"
## [37] "SRCN1"                        "NRP1"
## [39] "Epo"                          "GDNF"
## [41] "14-3-3 protein zeta/delta"    "ANK2"
## [43] "a-Synuclein"                  "CSRP3"
```

```
# 4. Evaluation of LASSO Classifier on Test Set (The Answer)

# Predict probabilities on the held-out test set
pred_prob_lasso <- predict(lasso_fit, newx = x_test,
                           s = "lambda.min", type = "response")
pred_class_lasso <- factor(ifelse(pred_prob_lasso > 0.5,
                                  "ASD", "TD"), levels = c("TD", "ASD"))

# Calculate AUROC
roc_lasso <- roc(y_test, as.numeric(pred_prob_lasso),
                 levels = c("TD", "ASD"))
```

```
## Setting direction: controls < cases
```

```
auroc_final <- auc(roc_lasso)

# Get confusion matrix object
cm <- confusionMatrix(pred_class_lasso, y_test, positive = "ASD")

# Print all metrics
cat("AUC ROC (LASSO classifier) on Test Set: ", auroc_final, "\n")
```

```
## AUC ROC (LASSO classifier) on Test Set:  0.8577778
```

```r
cat("Test Set Accuracy (LASSO classifier): ", cm$overall['Accuracy'], "\n")
```

```
## Test Set Accuracy (LASSO classifier):  0.7666667
```

```r
cat("Sensitivity (LASSO classifier): ", cm$byClass['Sensitivity'], "\n")
```

```
## Sensitivity (LASSO classifier):  0.8
```

```r
cat("Specificity (LASSO classifier): ", cm$byClass['Specificity'], "\n")
```

```
## Specificity (LASSO classifier):  0.7333333
```

We chose to find an alternative panel that achieves improved classification accuracy by utilizing the pure LASSO (Least Absolute Shrinkage and Selection Operator) penalized regression model.

The original analysis used the intersection of three methods (multiple t-test, Random Forest, and LASSO) followed by an unregularized Logistic Regression. For this alternative approach, we used the LASSO method alone to perform both feature selection and classification. The LASSO model was trained on the 80% training partition, and the optimal penalty ($\lambda_{\min}$) automatically selected the feature panel.

Results:

1. Alternative Panel: The LASSO model selected a panel of 44 proteins listed above under the Alternate Panel.
2. Test Set Performance: The LASSO classifier achieved a Test Set AUC ROC (Area Under the Receiver Operating Characteristic curve) of 0.858, accuracy of 0.767, sensitivity of 0.800, and specificity of 0.733.

Comparison to the benchmark:

$$\text{roc auc} = 0.883 \text{ ; accuracy} = 0.774 \text{ ; sensitivity} = 0.812 \text{ ; specificity} = 0.733$$

Metrics are similar to the baseline. Panel is more compelex though (compared to panel of 5 proteins in the baseline).