project2 Q1

Coraline Zhu

2025-10-30

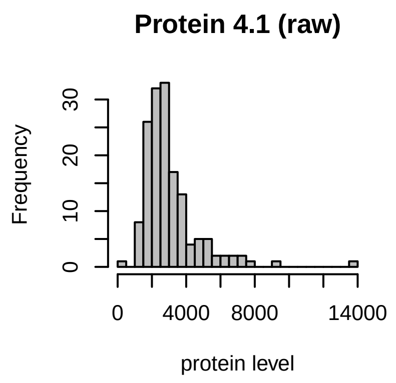
```r
# install.packages(c("tidyverse", "e1071"))  # if needed
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(e1071)
library(readr)

dat <- read.csv("Desktop/biomarker-raw.csv", check.names = FALSE)

# keep only numeric protein columns (drop id/label columns if any)
num_dat <- dat %>% select(where(is.numeric))

# pick a small sample of proteins to visualize (change n if you want)
set.seed(1)
sample_prots <- sample(colnames(num_dat), size = min(6, ncol(num_dat)))

sample_prots
```

```
## [1] "Protein 4.1"
```

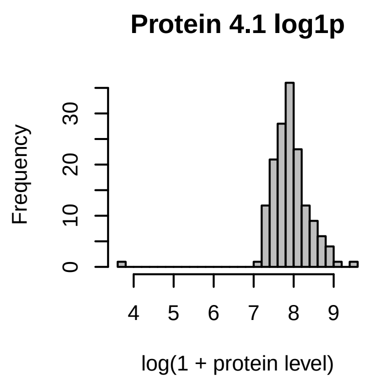
```r
# histograms of raw values
par(mfrow = c(2, 3))
for (v in sample_prots) {
  hist(num_dat[[v]], breaks = 30, main = paste(v, "(raw)"),
       xlab = "protein level", col = "grey")
}
par(mfrow = c(1,1))
```



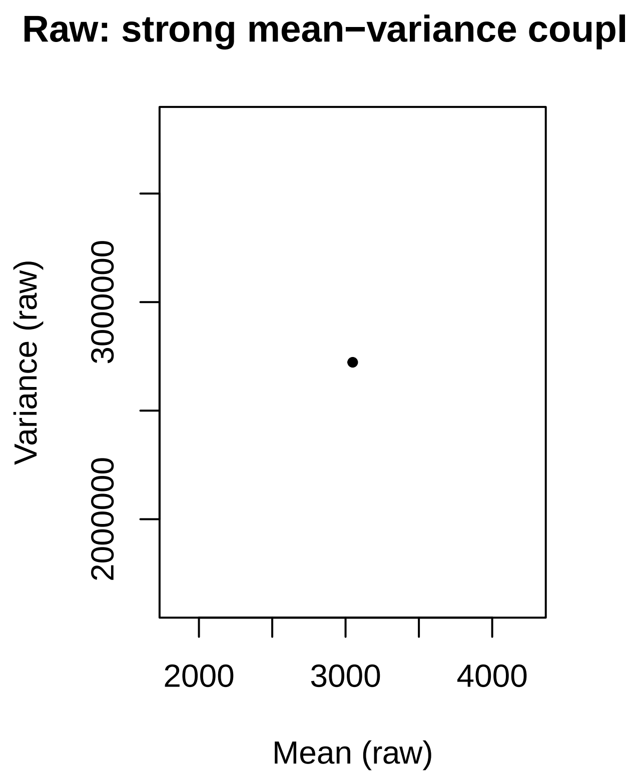
```
# skewness before vs after log1p (log(1+x) works even if zeros are present)
sk_tbl <- tibble(
  protein = sample_prots,
  skew_raw = map_dbl(sample_prots, ~ skewness(num_dat[[.x]], na.rm = TRUE)),
  skew_log = map_dbl(sample_prots, ~ skewness(log1p(num_dat[[.x]]), na.rm = TRUE))
)
sk_tbl
```

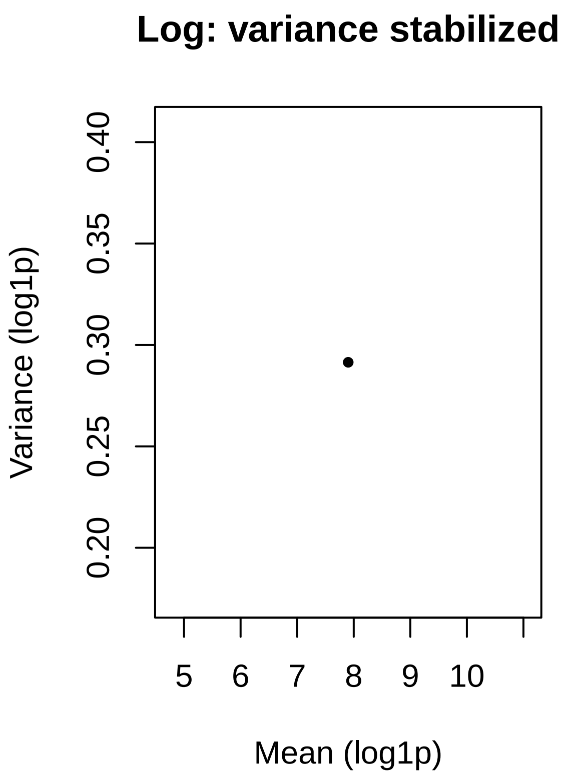
```
## # A tibble: 1 x 3
##   protein     skew_raw skew_log
##   <chr>          <dbl>    <dbl>
## 1 Protein 4.1     2.64    -2.51
```

```
# histograms after log1p transform
par(mfrow = c(2, 3))
for (v in sample_prots) {
  hist(log1p(num_dat[[v]]), breaks = 30, main = paste(v, "log1p"),
       xlab = "log(1 + protein level)", col = "grey")
}
par(mfrow = c(1,1))
```



```
# mean-variance relationship across all proteins: raw vs log
mean_raw <- sapply(num_dat, function(x) mean(x, na.rm = TRUE))
var_raw  <- sapply(num_dat, function(x) var(x,  na.rm = TRUE))

mean_log <- sapply(num_dat, function(x) mean(log1p(x), na.rm = TRUE))
var_log  <- sapply(num_dat, function(x) var(log1p(x),  na.rm = TRUE))

par(mfrow = c(1,2))
plot(mean_raw, var_raw, pch = 19, cex = .6,
     xlab = "Mean (raw)", ylab = "Variance (raw)",
     main = "Raw: strong mean-variance coupling")
plot(mean_log, var_log, pch = 19, cex = .6,
     xlab = "Mean (log1p)", ylab = "Variance (log1p)",
     main = "Log: variance stabilized")
```





```r
par(mfrow = c(1,1))
```

```r
summary_tbl <- num_dat %>%
  summarise(across(all_of(sample_prots), list(
    p50 = ~median(.x, na.rm=TRUE),
    mean = ~mean(.x, na.rm=TRUE),
    p95 = ~quantile(.x, .95, na.rm=TRUE),
    p99 = ~quantile(.x, .99, na.rm=TRUE)
  ))) %>%
  pivot_longer(everything(),
               names_to = c("protein","stat"),
               names_sep = "_",
               values_to = "value") %>%
  pivot_wider(names_from = stat, values_from = value)

summary_tbl
```

```
## # A tibble: 1 x 5
##   protein       p50  mean   p95   p99
##   <chr>       <dbl> <dbl> <dbl> <dbl>
## 1 Protein 4.1 2648. 3049. 6363. 8456.
```

Interpretation (why log-transform?):

Raw protein levels are highly right-skewed. In the raw histograms, most samples sit near the lower end with a long right tail and big outliers. The skewness numbers (skew_raw) are usually large and positive.

After a log transform, distributions look more symmetric. The log1p histograms compress the right tail, and skew_log becomes much closer to 0. This makes many proteins look more normal.

Variance stabilization. In the raw data, proteins with larger means also have much larger variance. On the log scale, that relationship is flatter, so variability is more comparable across proteins. This helps methods that assume roughly constant variance or are sensitive to scale.

Reduces the influence of extreme values. Log compresses very large measurements so a few high-abundance proteins don't dominate model fitting or distance calculations.

Handles zeros safely. Using log1p(x) = log(1+x) avoids issues when some proteins have zero (or near zero) values.

Conclusion:

We log-transform protein levels because the raw values are strongly rightskewed with heavy tails and heteroscedasticity; logging makes them more symmetric, stabilizes variance, and reduces the impact of outliers—yielding features that are easier for downstream models to learn from.