Data_Simulation_new

Shizhao_Yang

2025-03-16

time+treatment+interaction

```
# ---------------------------------------------------------
# Simulating RNA-seq Data with Main Effects & Interaction
# (Aligned with lmerSeq Paper)
# ---------------------------------------------------------

library(MASS)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

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

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(matrixStats)
```

```
##
## Attaching package: 'matrixStats'
```

```
## The following object is masked from 'package:dplyr':
##
##     count
```

```r
set.seed(571)

# Parameters
n_subjects <- 20       # 10 control, 10 case
n_genes <- 10000       # Total genes
n_de <- 500            # 5% DE genes (interaction only)
base_mean <- 50        # Baseline expression (control, baseline)
sdlog_basemean <- 0.5  # Log-normal spread of baseline expression
mean_effect <- 1.2     # Mean log2 fold-change for DE genes (interaction)
sd_effect <- 0.5       # SD of log2 fold-change
dropout_rate <- 0.1    # 10% dropout
time_points <- c("baseline", "followup")

# Subject random effects (N(0, 1))
subject_re <- rnorm(n_subjects, mean = 0, sd = 1)

# Library size variation (technical noise)
library_sizes <- rlnorm(n_subjects * length(time_points),
                        meanlog = log(1e6),
                        sdlog = 0.2)

# Baseline means (log-normal)
base_means <- rlnorm(n_genes, meanlog = log(base_mean), sdlog = sdlog_basemean)

# Gene-specific dispersion (inverse relationship to mean)
dispersion_vec <- 0.1 + 1 / base_means

# DE effect sizes (interaction term only)
effect_sizes <- rnorm(n_de, mean = mean_effect, sd = sd_effect)

simulate_rnaseq_full_model <- function() {

  # Assign groups
  group_assign <- rep(c("control", "case"), each = n_subjects / 2)

  # Design matrix
  design <- expand.grid(
    subject_id = 1:n_subjects,
    time = time_points
  )
  design$condition <- group_assign[design$subject_id]

  # Align library sizes
  library_sizes_ordered <- library_sizes[order(design$subject_id)]

  # Simulate counts
  counts <- matrix(NA, nrow = nrow(design), ncol = n_genes)

  for (g in 1:n_genes) {
    # Initialize coefficients (β1 = β2 = 0 for all genes)
    beta_g1 <- 0  # Main effect of group (fixed at 0)
    beta_g2 <- 0  # Main effect of time (fixed at 0)
    beta_g3 <- if (g <= n_de) effect_sizes[g] else 0  # Interaction effect

    mu_vals <- numeric(nrow(design))
    for (row_i in 1:nrow(design)) {
      subj_id <- design$subject_id[row_i]
      group <- design$condition[row_i]
      time_num <- ifelse(design$time[row_i] == "followup", 1, 0)

      # Compute log2FC: β1*group + β2*time + β3*group*time
      log2FC <- beta_g1 * (group == "case") +
                beta_g2 * time_num +
                beta_g3 * (group == "case") * time_num

      # Base expression with random effect + library size
      base_expr <- base_means[g] * 2^(log2FC)
      re_factor <- exp(subject_re[subj_id])  # Unscaled random effect
      lib_factor <- library_sizes_ordered[row_i] / median(library_sizes)

      mu_vals[row_i] <- base_expr * re_factor * lib_factor
    }

    # Negative binomial counts
    counts[, g] <- rnbinom(nrow(design),
                           size = 1/dispersion_vec[g],
                           mu = mu_vals)

    # Add dropout
    zeros <- sample(1:nrow(design), floor(dropout_rate * nrow(design)))
    counts[zeros, g] <- 0
  }

  colnames(counts) <- paste0("Gene", 1:n_genes)
  return(list(design = design, counts = counts))
}

# Generate data
rnaseq_data_subject <- simulate_rnaseq_full_model()
```

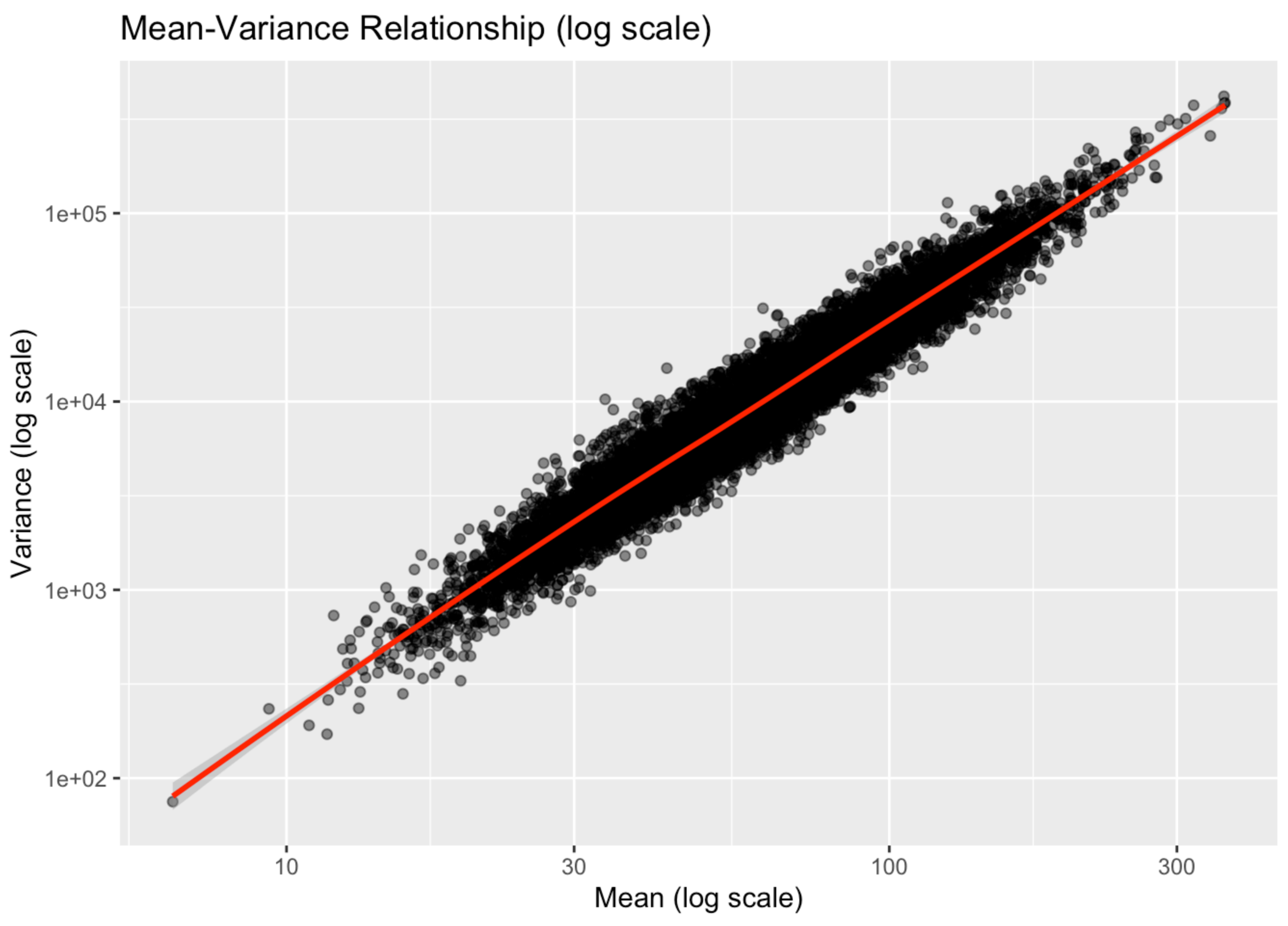
```
library(matrixStats)  # For rowVars or colVars, rowMeans2, etc.
library(ggplot2)

# Let's assume each row is a sample (observation) and each column is a gene
# so we compute the "mean expression" for each gene
gene_means <- colMeans(rnaseq_data_subject$counts)
gene_vars  <- colVars(rnaseq_data_subject$counts)

# Option 2: Using ggplot2 for a nicer visualization
df <- data.frame(mean = gene_means, variance = gene_vars)

ggplot(df, aes(x = mean, y = variance)) +
  geom_point(alpha = 0.5) +  # semi-transparent points
  scale_x_log10() +
  scale_y_log10() +
  geom_smooth(method = "loess", color = "red") +
  labs(title = "Mean-Variance Relationship (log scale)",
       x = "Mean (log scale)",
       y = "Variance (log scale)")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



```
save(rnaseq_data_subject, file = "simulated_rnaseq_data_new.RData")
```