Quality Control for bulk RNA-seq data

Ximing Ran

2025-05-15

Contents

```
library(tibble)
library(tidyr)
library(dplyr)
library(rtracklayer)
```

```
# load function from local files
```

1. Read the count data

In this section, we will read the raw count data from the neuron_bulkRNA folder.

```
sample_df <- read.csv(here::here("data", "neuron_bulkRNA",
                                  "sample_info.csv"),
                        header = TRUE)
colnames(sample_df) <- c("Raw", "SampleID", "Condition")


condition_list <- sample_df$Condition

sample_list <- sample_df$Raw
sample_ID_list <- sample_df$SampleID
ref_sample <- paste0(sample_list[1], "_final_gene_with_names.csv")
counts_ref <- read.csv(here::here("data", "neuron_bulkRNA",
                                  "raw",ref_sample),
                        header = TRUE)

counts <- tibble()
for (i in 1:length(sample_list)) {
  sample_raw <- sample_list[i]
  gene_file <- here::here("data", "neuron_bulkRNA","raw",
                          paste0(sample_raw, "_final_gene_with_names.csv"))
  gene <- read.csv(gene_file, header = TRUE)

  if (i == 1) {
    counts <- gene[, c("GeneID", "GeneName")]
    counts <- counts %>% mutate(!!sample_ID_list[i] := gene$Count)
  } else {
    temp_counts <- gene[, c("GeneID", "Count")]
    colnames(temp_counts)[2] <- sample_ID_list[i]
    counts <- merge(counts, temp_counts, by = "GeneID")
  }
}

# replace all - in column names with _
colnames(counts) <- gsub("-", "_", colnames(counts))
write.csv(counts,here::here("data", "neuron_bulkRNA",
                            "neuron_bulkRNA_counts_raw.csv"),
          row.names = FALSE)
```

2. Map the gene name.

In this section, we will map the gene name to the gene ID using the GTF file. Since some gene may have different gene name, we will check if the gene name is unique. If not, we will find the gene that has more than one gene name. And merge their counts. The gene annotation comes from the the file `Homo_sapiens.GRCh38.106.gtf`. The final table will be stored in `results/01-QC/synaptosomes_bulkRNA_counts_cleaned.csv`.

```
# read the gtf file
gtf_data <- import(here::here("data", "ref","Homo_sapiens.GRCh38.106.gtf"))

gtf_df <- as.data.frame(gtf_data)

gtf_genes <- gtf_df %>%
  filter(type == "gene") %>%
  select(gene_id, gene_name)
colnames(gtf_genes) <- c("GeneID","GeneName")

# check the annotation format
head(gtf_genes)
```

```
##            GeneID GeneName
## 1 ENSG00000186827  TNFRSF4
## 2 ENSG00000186891 TNFRSF18
## 3 ENSG00000160072   ATAD3B
## 4 ENSG00000260179     <NA>
## 5 ENSG00000234396     <NA>
## 6 ENSG00000225972 MTND1P23
```

```
gtf_gene <- na.omit(gtf_genes)

# check if the gene is unique
unique_genename <- length(unique(gtf_genes$GeneName)) == nrow(gtf_genes)
print(paste("GeneName is unique:", unique_genename))
```

```
## [1] "GeneName is unique: FALSE"
```

```
# Find the genes that appear more than once
duplicate_genes <- gtf_genes$GeneName[duplicated(gtf_genes$GeneName)]
duplicate_genes <- unique(duplicate_genes)

# Correct sprintf statement
```

```r
print(sprintf("There are %d genes with duplicate gene names",
              length(duplicate_genes)))
```

```
## [1] "There are 68 genes with duplicate gene names"
```

```r
# Merge counts data with GTF information
merged_data <- counts %>%
  left_join(gtf_genes, by = "GeneID")%>%
  select(-GeneName.x,-GeneID)

names(merged_data)[names(merged_data) == "GeneName.y"] <- "gene"

# find the same gene
aggregated_data <- merged_data %>%
  group_by(gene) %>%
  summarise(across(everything(), \(x) sum(x, na.rm = TRUE)))



# Using sprintf (Recommended for better formatting)
print(sprintf("The shape of the count matrix is: %d x %d",
              dim(aggregated_data)[1], dim(aggregated_data)[2]))
```

```
## [1] "The shape of the count matrix is: 39853 x 17"
```

```r
# clean the NA in gene
aggregated_data <- aggregated_data[complete.cases(aggregated_data), ]
# check if here is any NA in gene
print(sprintf("There are %d genes with NA gene name",
              sum(is.na(aggregated_data$gene))))
```

```
## [1] "There are 0 genes with NA gene name"
```

```r
write.csv(aggregated_data, here::here("data", "neuron_bulkRNA",
                                      "neuron_bulkRNA_counts_cleaned.csv"),
          row.names = FALSE)
```

3. make gene refernce

ALso, we will make a reference table for the gene length. The reference table will be stored in `results/01-QC/gene_lengths.csv`.

```r
# Filter for rows with gene information only
genes_df <- gtf_df %>%
  filter(type == "gene") %>%
  select(seqnames, start, end, gene_id, gene_name)


genes_df <- genes_df %>%
  mutate(gene_length = end - start + 1) %>%
  select(gene_id, gene_name, gene_length)

# Save to CSV
write.csv(genes_df,here::here("data", "ref", "gene_lengths.csv"),
          row.names = FALSE)
```

```r
sessionInfo()
```

```
## R version 4.4.0 (2024-04-24)
## Platform: aarch64-apple-darwin20
## Running under: macOS 15.4
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  LAPACK v
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/New_York
## tzcode source: internal
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] rtracklayer_1.64.0   GenomicRanges_1.56.2 GenomeInfoDb_1.40.1
##  [4] IRanges_2.38.1       S4Vectors_0.42.1     BiocGenerics_0.50.0
##  [7] knitr_1.50           lubridate_1.9.4      forcats_1.0.0
## [10] stringr_1.5.1        dplyr_1.1.4          purrr_1.0.4
## [13] readr_2.1.5          tidyr_1.3.1          tibble_3.2.1
## [16] ggplot2_3.5.2        tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] SummarizedExperiment_1.34.0 gtable_0.3.6
##  [3] rjson_0.2.23                xfun_0.52
##  [5] lattice_0.22-7              Biobase_2.64.0
##  [7] tzdb_0.5.0                  vctrs_0.6.5
##  [9] tools_4.4.0                 bitops_1.0-9
## [11] generics_0.1.4              curl_6.2.2
## [13] parallel_4.4.0              pkgconfig_2.0.3
## [15] Matrix_1.7-3                RColorBrewer_1.1-3
## [17] lifecycle_1.0.4             GenomeInfoDbData_1.2.12
## [19] compiler_4.4.0              farver_2.1.2
## [21] Rsamtools_2.20.0            Biostrings_2.72.1
## [23] codetools_0.2-20            htmltools_0.5.8.1
## [25] RCurl_1.98-1.17             yaml_2.3.10
## [27] pillar_1.10.2               crayon_1.5.3
## [29] BiocParallel_1.38.0         DelayedArray_0.30.1
## [31] abind_1.4-8                 tidyselect_1.2.1
## [33] digest_0.6.37               stringi_1.8.7
## [35] restfulr_0.0.15             rprojroot_2.0.4
## [37] fastmap_1.2.0               grid_4.4.0
## [39] here_1.0.1                  SparseArray_1.4.8
## [41] cli_3.6.5                   magrittr_2.0.3
## [43] S4Arrays_1.4.1              dichromat_2.0-0.1
## [45] XML_3.99-0.18               withr_3.0.2
## [47] scales_1.4.0                UCSC.utils_1.0.0
## [49] timechange_0.3.0            rmarkdown_2.29
## [51] XVector_0.44.0             httr_1.4.7
## [53] matrixStats_1.5.0          hms_1.1.3
## [55] evaluate_1.0.3             BiocIO_1.14.0
## [57] rlang_1.1.6                glue_1.8.0
## [59] rstudioapi_0.17.1          jsonlite_2.0.0
## [61] R6_2.6.1                   MatrixGenerics_1.16.0
## [63] GenomicAlignments_1.40.0   zlibbioc_1.50.0
```