task7_2

2025-09-30

```
# Load required packages ----------------------------------------------------
library(data.table)  # fast tab-delimited reader: fread()
library(ggplot2)     # plotting
library(dplyr)       # small helpers (optional)
```

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

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

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

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

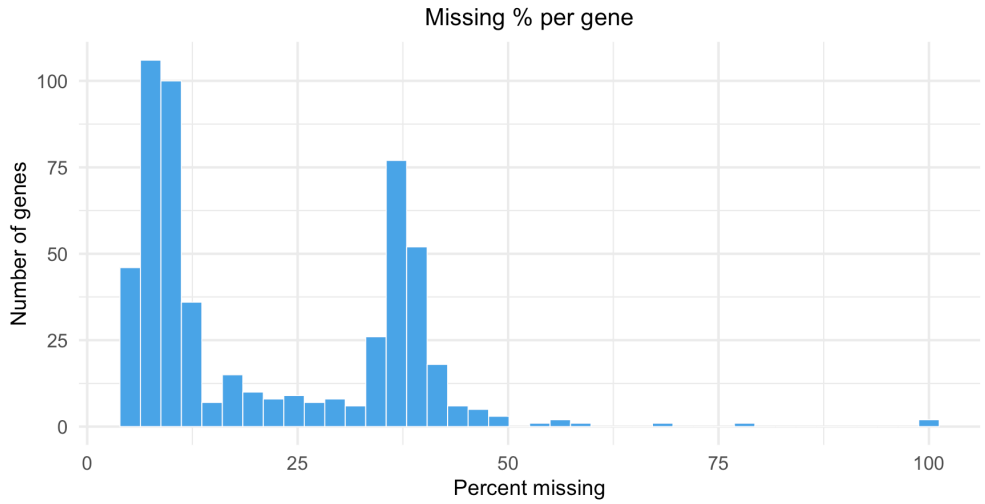
```
# FIXED ABSOLUTE PATH to your file (edit here if it ever changes) ----------------
tab_path <- "/Users/rui/Desktop/研究相关定题材料/KI-SciLifelab/microarray_data.tab"  # <- your pa
th

# Fail early if the file is missing ----------------------------------------------
if (!file.exists(tab_path)) {
  stop("File not found at:\n  ", tab_path,
       "\nTip: check the exact path & filename (case-sensitive).")
}

message("Reading TAB-separated microarray data from: ", tab_path)
```

```
## Reading TAB-separated microarray data from: /Users/rui/Desktop/研究相关定题材料/KI-SciLifelab/
microarray_data.tab
```

```
# Read as tab-separated text; treat common tokens as NA ------------------------
dt <- tryCatch(
  fread(tab_path,
        sep = "\t",
        na.strings = c("", "NA", "NaN", "NULL"),
        quote = "",             # robust when matrix is numeric text
        showProgress = FALSE),
  error = function(e) stop("Reading failed: ", conditionMessage(e))
)

# Build expression matrix:
# - First column = gene IDs (can be missing/duplicated)
# - Remaining columns = numeric expression values
gene_ids <- as.character(dt[[1]])                                  # gene/probe identifiers
X <- as.matrix(as.data.frame(lapply(dt[, -1], function(v) {        # coerce to numeric
  suppressWarnings(as.numeric(v))
})))

# Ensure we have row names (unique gene IDs) -----------------------
if (is.null(gene_ids)) gene_ids <- rep("", nrow(X))                # guard if truly absent
bad <- which(is.na(gene_ids) | !nzchar(gene_ids))                  # empty or NA IDs
if (length(bad)) gene_ids[bad] <- paste0("gene_", bad)             # fill placeholders
rownames(X) <- make.unique(gene_ids)                               # enforce uniqueness

# 2a Quick sanity print ------------------------------------------------
cat("Loaded matrix size: ", nrow(X), " rows (genes) × ", ncol(X), " columns (samples)\n", sep
= "")
```

```
## Loaded matrix size: 553 rows (genes) × 999 columns (samples)
```

```
##2b Count missing values per gene and visualize
# Count NA per row (gene) and percent ------------------------------------------
na_per_gene  <- rowSums(is.na(X))                # how many NAs in each gene
pct_per_gene <- na_per_gene / ncol(X) * 100      # percentage NA per gene

# Wrap into a small data.frame for plotting ------------------------------------
df_miss <- data.frame(
  gene     = rownames(X),
  NA_count = na_per_gene,
  NA_pct   = pct_per_gene,
  row.names = NULL
)

# Histogram of percent missing per gene (compact) ------------------------------
ggplot(df_miss, aes(NA_pct)) +
  geom_histogram(bins = 40, color = "white", linewidth = 0.2, fill = "#4EA5E9") +
  labs(title = "Missing % per gene",
       x = "Percent missing", y = "Number of genes") +
  theme_minimal(base_size = 10) +
  theme(plot.title = element_text(size = 10, hjust = 0.5),
        axis.title  = element_text(size = 9),
        axis.text   = element_text(size = 8))
```




```
## 2c Genes with more than X% missing (X = 10%, 20%, 50%)
# Define thresholds (percent) ----------------------------------------------------
thr <- c(10, 20, 50)

# For each threshold, list genes exceeding it ------------------------------------
genes_over <- lapply(thr, function(t) rownames(X)[pct_per_gene > t])
names(genes_over) <- paste0("over_", thr, "pct")

# Print counts and a few examples ------------------------------------------------
counts <- sapply(genes_over, length)
cat("Counts of genes with >X% missing:\n")
```

```
## Counts of genes with >X% missing:
```

```
for (i in seq_along(thr)) {
  ex <- head(genes_over[[i]], 10)
  cat(sprintf("  > %2d%% : %5d genes. Examples: %s\n",
              thr[i], counts[i],
              if (length(ex)) paste(ifelse(is.na(ex), "NA", ex), collapse = ", ") else "(non
e)"))
}
```

```
##   > 10% :   346 genes. Examples: gene_2, -1.7, gene_7, gene_13, gene_17, gene_18, gene_19,
gene_21, gene_24, 1.951
##   > 20% :   236 genes. Examples: gene_2, gene_7, gene_17, gene_18, gene_19, gene_21, gene_2
4, 1.951, gene_26, 1.333
##   > 50% :     8 genes. Examples: gene_17, gene_57, -2.835, gene_241, gene_250, gene_335, ge
ne_340, gene_541
```

2d) Impute missing values

```
# Define "average expression" per gene as the ROW MEDIAN (robust to outliers)
# If an entire row is NA, keep it as NA.
row_avg <- apply(X, 1, function(v) {
  if (all(is.na(v))) NA_real_ else median(v, na.rm = TRUE)
})

# Copy matrix and replace each NA with that gene's "average expression" ----------
X_imp  <- X
idx_na <- which(is.na(X_imp), arr.ind = TRUE)     # matrix coordinates of NAs
if (nrow(idx_na) > 0) {
  X_imp[idx_na] <- row_avg[idx_na[, 1]]          # fill NA at (i, j) with row_avg[i]
}

# Report rows that are all-NA (cannot be imputed by a median) --------------------
all_na_rows <- which(!is.finite(row_avg))
if (length(all_na_rows) > 0) {
  message(length(all_na_rows), " gene(s) are all-NA; left as NA (no imputation possible).")
}

# Sanity check: NA counts before vs after ----------------------------------------
cat("Total NA BEFORE imputation :", sum(is.na(X)),     "\n")
```

```
## Total NA BEFORE imputation : 117696
```

```
cat("Total NA AFTER  imputation :", sum(is.na(X_imp)), "\n")
```

```
## Total NA AFTER  imputation : 1998
```