Pertussis Challenge 4.1_+ge

Runqi Zhang

2024-11-21

```
# Load required libraries
suppressPackageStartupMessages({
  library(dplyr)
  library(ggplot2)
  library(tidyr)
  library(tidyverse)
  library(readr)
  library(kableExtra)
  library(glmnet)
  library(here)
  library(knitr)
  library(GSVA)
  library(agua)
  library(tibble)        # For handling row names
  library(impute)
  #source(here("./scripts/codebase.R"))

})
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3

## Warning: package 'tidyr' was built under R version 4.3.3

## Warning: package 'tidyverse' was built under R version 4.3.3

## Warning: package 'readr' was built under R version 4.3.3

## Warning: package 'purrr' was built under R version 4.3.3

## Warning: package 'stringr' was built under R version 4.3.3

## Warning: package 'forcats' was built under R version 4.3.3

## Warning: package 'lubridate' was built under R version 4.3.3

## Warning: package 'kableExtra' was built under R version 4.3.3

## Warning: package 'glmnet' was built under R version 4.3.3
## Warning: package 'here' was built under R version 4.3.3

## Warning: package 'knitr' was built under R version 4.3.3

## Warning: package 'GSVA' was built under R version 4.3.3

## Warning: package 'agua' was built under R version 4.3.3

## Warning: package 'parsnip' was built under R version 4.3.3
```

```r
# Define working directory and start H2O
workDir <- "C:/Users/zhang/Desktop/cmi-pb-3rd-final/Runqi/CMI-PB"
agua::h2o_start()
```

```
## Warning: JAVA not found, H2O may take minutes trying to connect.

## Warning in h2o.clusterInfo():
## Your H2O cluster version is (11 months and 1 day) old. There may be a newer version available.
## Please download and install the latest version from: https://h2o-release.s3.amazonaws.com/h2o/latest_
```

```r
options(readr.show_col_types = FALSE)
```

```r
read_data <- function(year, type = "LD") {
  list(
    pts = read_tsv(file.path(workDir, paste0("data/", year, type, "_subject.tsv"))),
    sample = read_tsv(file.path(workDir, paste0("data/", year, type, "_specimen.tsv"))),
    ge = read_tsv(file.path(workDir, paste0("data/", year, type, "_pbmc_gene_expression.tsv")))
  )
}

# Load datasets for 2020-2023
data2020 <- read_data("2020")
data2021 <- read_data("2021")
data2022 <- read_data("2022")
data2023 <- read_data("2023", type = "BD")
```

```r
# List of base gene IDs (genes of interest)
target_genes <- c(
  #"ENSG00000113525.9", #IL5
  "ENSG00000107485.15", #GATA3
  #"ENSG00000136574.17", #GATA4
  "ENSG00000102145.13", #GATA1
  #"ENSG00000130700.6", #GATA5
  #"ENSG00000141448.8", #GATA6
  "ENSG00000179348.11", #GATA2
  "ENSG00000220201.7", #ZGLP1
  "ENSG00000104447.12", #TRPS1
  "ENSG00000071564.14", #TCF3
  "ENSG00000196628.15", #TCF4
  "ENSG00000140262.17", #TCF12
  #"ENSG00000112499.12", #SLC22A2
  "ENSG00000172216.5", #CEBPB
  "ENSG00000185591.9", #SP1
  "ENSG00000167182.14", #SP2
  "ENSG00000172845.14", #SP3
  "ENSG00000105866.13" #SP4
)
```

to be tested “ENSG00000111537.4”, #IFNG “ENSG00000105829.11”, #BET1 “ENSG00000131196.17”, #NFATC1 “ENSG00000101096.19”, #NFATC2 “ENSG00000072736.18”, #NFATC3 #“ENSG00000100968.13”, #NFATC4 “ENSG00000102908.20”, #NFAT5 “ENSG00000109320.11”, #NFKB1 “ENSG00000077150.18”, #NFKB2 “ENSG00000162924.13”, #REL “ENSG00000100811.12”, #YY1 #“ENSG00000230797.2”, #YY2 #“ENSG00000179059.9”, #ZFP42 “ENSG00000115415.18”, #STAT1 “ENSG00000170581.13”, #STAT2 “ENSG00000168610.14”, #STAT3 “ENSG00000138378.17”, #STAT4 “ENSG00000126561.16”, #STAT5A “ENSG00000166888.11”, #STAT6 “ENSG00000173757.9” #STAT5B

```
# Function to extract day 0 TPM values
extract_day0_tpm <- function(ge_data, sample_data, pts_data, target_genes) {
  ge_data %>%
    filter(versioned_ensembl_gene_id %in% target_genes) %>%
    inner_join(sample_data, by = "specimen_id") %>%
    inner_join(pts_data, by = "subject_id") %>%
    filter(planned_day_relative_to_boost == 0) %>%
    dplyr::select(subject_id, versioned_ensembl_gene_id, tpm) %>%
    pivot_wider(
      names_from = versioned_ensembl_gene_id,
      values_from = tpm,
      names_prefix = "tpm_"
    ) %>%
    replace(is.na(.), 0) %>% # Replace NA with 0
    column_to_rownames("subject_id") # Use subject_id as rownames
}

# Extract Day 0 TPM values for 2021, 2022, and 2023
x21_gene_expr <- extract_day0_tpm(data2021$ge, data2021$sample, data2021$pts, target_genes)
x22_gene_expr <- extract_day0_tpm(data2022$ge, data2022$sample, data2022$pts, target_genes)
x23_gene_expr <- extract_day0_tpm(data2023$ge, data2023$sample, data2023$pts, target_genes)
```

```
# Load data for 2021
pts2021DF <- read_tsv(file.path(workDir, "data/2021LD_subject.tsv"))
sample2021DF <- read_tsv(file.path(workDir, "data/2021LD_specimen.tsv"))
tcp2021DF <- read_tsv(file.path(workDir, "data/2021LD_t_cell_polarization.tsv"))
tca2021DF <- read_tsv(file.path(workDir, "data/2021LD_t_cell_activation.tsv"))

# Load data for 2022
pts2022DF <- read_tsv(file.path(workDir, "data/2022LD_subject.tsv"))
sample2022DF <- read_tsv(file.path(workDir, "data/2022LD_specimen.tsv"))
tcp2022DF <- read_tsv(file.path(workDir, "data/2022LD_t_cell_polarization.tsv"))
tca2022DF <- read_tsv(file.path(workDir, "data/2022LD_t_cell_activation.tsv"))

# Load data for 2023
pts2023DF <- read_tsv(file.path(workDir, "data/2023BD_subject.tsv"))
sample2023DF <- read_tsv(file.path(workDir, "data/2023BD_specimen.tsv"))
tcp2023DF <- read_tsv(file.path(workDir, "data/2023BD_t_cell_polarization.tsv"))
```

```r
tca2023DF <- read_tsv(file.path(workDir, "data/2023BD_t_cell_activation.tsv"))
```

```r
# Function to calculate Th1/Th2 ratio for a given dataset
calculate_th1_th2_ratio <- function(tcpDF, sampleDF, ptsDF) {
  tcpDF %>%
    filter(stimulation == "PT", protein_id %in% c("P01579", "P05113")) %>% # Select IFN-  and IL-5 with
    merge(y = sampleDF, by = "specimen_id", all.x = TRUE) %>%              # Merge sample data
    merge(y = ptsDF, by = "subject_id", all.x = TRUE) %>%                  # Merge patient data
    filter(planned_day_relative_to_boost == 30) %>%                       # Filter for Day 30 post-boos
    dplyr::select(subject_id, protein_id, analyte_counts) %>%             # Select relevant columns
    group_by(subject_id, protein_id) %>%                                  # Group by subject and protei
    summarize(analyte_counts = mean(analyte_counts, na.rm = TRUE), .groups = "drop") %>% # Aggregate du
    pivot_wider(names_from = protein_id, values_from = analyte_counts) %>% # Reshape to wide format
    mutate(Th1_Th2_ratio = P01579 / P05113) %>%                           # Calculate Th1/Th2 ratio
    dplyr::select(subject_id, Th1_Th2_ratio)                              # Select output columns
}

# Calculate Th1/Th2 ratio for 2021 and 2022
y21DF <- calculate_th1_th2_ratio(tcp2021DF, sample2021DF, pts2021DF)
y22DF <- calculate_th1_th2_ratio(tcp2022DF, sample2022DF, pts2022DF)
```

```r
# Process TCP data for a single year
preprocess_tcp <- function(tcpDF, sampleDF, ptsDF, stimulation_filter, proteins, timepoint) {
  tcpDF %>%
    filter(stimulation %in% stimulation_filter, protein_id %in% proteins) %>%
    merge(y = sampleDF, by = "specimen_id", all.x = TRUE) %>%
    merge(y = ptsDF, by = "subject_id", all.x = TRUE) %>%
    filter(planned_day_relative_to_boost == timepoint) %>%
    dplyr::select(subject_id, stimulation, protein_id, analyte_counts) %>%
    pivot_wider(
      names_from = c(stimulation, protein_id),
      values_from = analyte_counts
    ) %>%
    replace(is.na(.), 0) %>%
    column_to_rownames("subject_id") %>%
    setNames(make.names(names(.), unique = TRUE))
}

# Process TCA data for a single year
preprocess_tca <- function(tcaDF, sampleDF, ptsDF, stimulation_filter, timepoint) {
  tcaDF %>%
    filter(stimulation %in% stimulation_filter) %>%
    merge(y = sampleDF, by = "specimen_id", all.x = TRUE) %>%
    merge(y = ptsDF, by = "subject_id", all.x = TRUE) %>%
    filter(planned_day_relative_to_boost == timepoint) %>%
    dplyr::select(subject_id, stimulation, analyte_percentages) %>%
    pivot_wider(
      names_from = stimulation,
      values_from = analyte_percentages
    ) %>%
    replace(is.na(.), 0) %>%
    column_to_rownames("subject_id") %>%
    setNames(make.names(names(.), unique = TRUE))
```

```r
}

# Combine TCP and TCA predictors for a single year
combine_tcp_tca <- function(tcpDF, tcaDF, sampleDF, ptsDF, yDF, tcp_stimulation, tcp_proteins, tca_stimu
  # Process TCP
  tcp_processed <- preprocess_tcp(tcpDF, sampleDF, ptsDF, tcp_stimulation, tcp_proteins, timepoint) %>%
    rownames_to_column(var = "subject_id") # Add row names back as a column

  # Process TCA
  tca_processed <- preprocess_tca(tcaDF, sampleDF, ptsDF, tca_stimulation, timepoint) %>%
    rownames_to_column(var = "subject_id") # Add row names back as a column

  # Merge TCP and TCA predictors
  combined_data <- full_join(
    tcp_processed, tca_processed,
    by = "subject_id" # Merge by subject_id
  ) %>%
    replace(is.na(.), 0) %>% # Replace NA with 0
    column_to_rownames(var = "subject_id") # Convert subject_id back to row names

  # Align with response variable
  combined_data <- combined_data[rownames(combined_data) %in% yDF$subject_id, ]

  return(combined_data)
}


# Define constants
tcp_stimulation <- c("PT") # TCP stimulation
tcp_proteins <- c("P01579", "P05113") # IFN-  and IL-5
tca_stimulation <- c("PT", "PHA", "DMSO", "TT") # TCA stimulation
timepoint <- 0 # Baseline (Day 0)

# Process TCP and TCA data for each year
x21DF <- combine_tcp_tca(tcp2021DF, tca2021DF, sample2021DF, pts2021DF, y21DF, tcp_stimulation, tcp_pro
x22DF <- combine_tcp_tca(tcp2022DF, tca2022DF, sample2022DF, pts2022DF, y22DF, tcp_stimulation, tcp_pro

# Process TCP and TCA data for 2023
x23DF <- combine_tcp_tca(
  tcp2023DF,
  tca2023DF,
  sample2023DF,
  pts2023DF,
  data.frame(subject_id = pts2023DF$subject_id), # Correct alignment with 2023 subjects
  tcp_stimulation,
  tcp_proteins,
  tca_stimulation,
  timepoint
)

# Combine predictors (TCP/TCA data) with Day 0 gene expression
combine_predictors <- function(predictor_data, gene_expr_data) {
  # Add rownames as a column for merging
  predictor_data <- predictor_data %>% rownames_to_column("subject_id")
  gene_expr_data <- gene_expr_data %>% rownames_to_column("subject_id")

  # Merge on subject_id to ensure alignment
  combined_data <- predictor_data %>%
    full_join(gene_expr_data, by = "subject_id") %>%
    replace(is.na(.), 0) %>% # Replace NA with 0
    column_to_rownames("subject_id") # Restore rownames

  return(combined_data)
}

# Combine datasets for each year
x21DF <- combine_predictors(x21DF, x21_gene_expr)
x22DF <- combine_predictors(x22DF, x22_gene_expr)
x23DF <- combine_predictors(x23DF, x23_gene_expr)

# Ensure consistent columns across datasets
common_cols <- Reduce(intersect, list(colnames(x21DF), colnames(x22DF), colnames(x23DF)))
x21DF <- x21DF[, common_cols]
x22DF <- x22DF[, common_cols]
x23DF <- x23DF[, common_cols]

# Match x21DF rows to y21DF subject IDs
x21DF <- x21DF %>%
  rownames_to_column("subject_id") %>%  # Convert rownames to a column for matching
  filter(subject_id %in% y21DF$subject_id) %>%  # Filter rows to include only those in y21DF
  arrange(match(subject_id, y21DF$subject_id)) %>%  # Reorder rows to match y21DF order
  column_to_rownames("subject_id")  # Restore subject_id as rownames

# Match x22DF rows to y22DF subject IDs
x22DF <- x22DF %>%
  rownames_to_column("subject_id") %>%  # Convert rownames to a column for matching
  filter(subject_id %in% y22DF$subject_id) %>%  # Filter rows to include only those in y22DF
  arrange(match(subject_id, y22DF$subject_id)) %>%  # Reorder rows to match y22DF order
  column_to_rownames("subject_id")  # Restore subject_id as rownames
```

```
# Combine training data and response variable
trainDF <- rbind(x21DF, x22DF) %>%
  as.data.frame() %>%
  mutate(Th1_Th2_ratio = c(y21DF$Th1_Th2_ratio, y22DF$Th1_Th2_ratio))

# Impute missing values if needed
train_matrix <- as.matrix(trainDF) # Convert to matrix
imputed_matrix <- impute.knn(train_matrix)$data
trainDF <- as.data.frame(imputed_matrix) # Convert back to dataframe

trainDF$Th1_Th2_ratio <- log1p(trainDF$Th1_Th2_ratio)

# Train regression model
set.seed(3)
auto_fit <- auto_ml() %>%
  set_engine("h2o", max_runtime_secs = 5) %>%
  set_mode("regression") %>%
  fit(Th1_Th2_ratio ~ ., data = trainDF)

# Validate model on training data
train_predictions <- predict(auto_fit, new_data = trainDF)$.pred

# Calculate correlations
pearson_cor <- cor(train_predictions, trainDF$Th1_Th2_ratio, method = "pearson")
spearman_cor <- cor(train_predictions, trainDF$Th1_Th2_ratio, method = "spearman")

# Print correlations
cat("Pearson Correlation: ", pearson_cor, "\n")
```

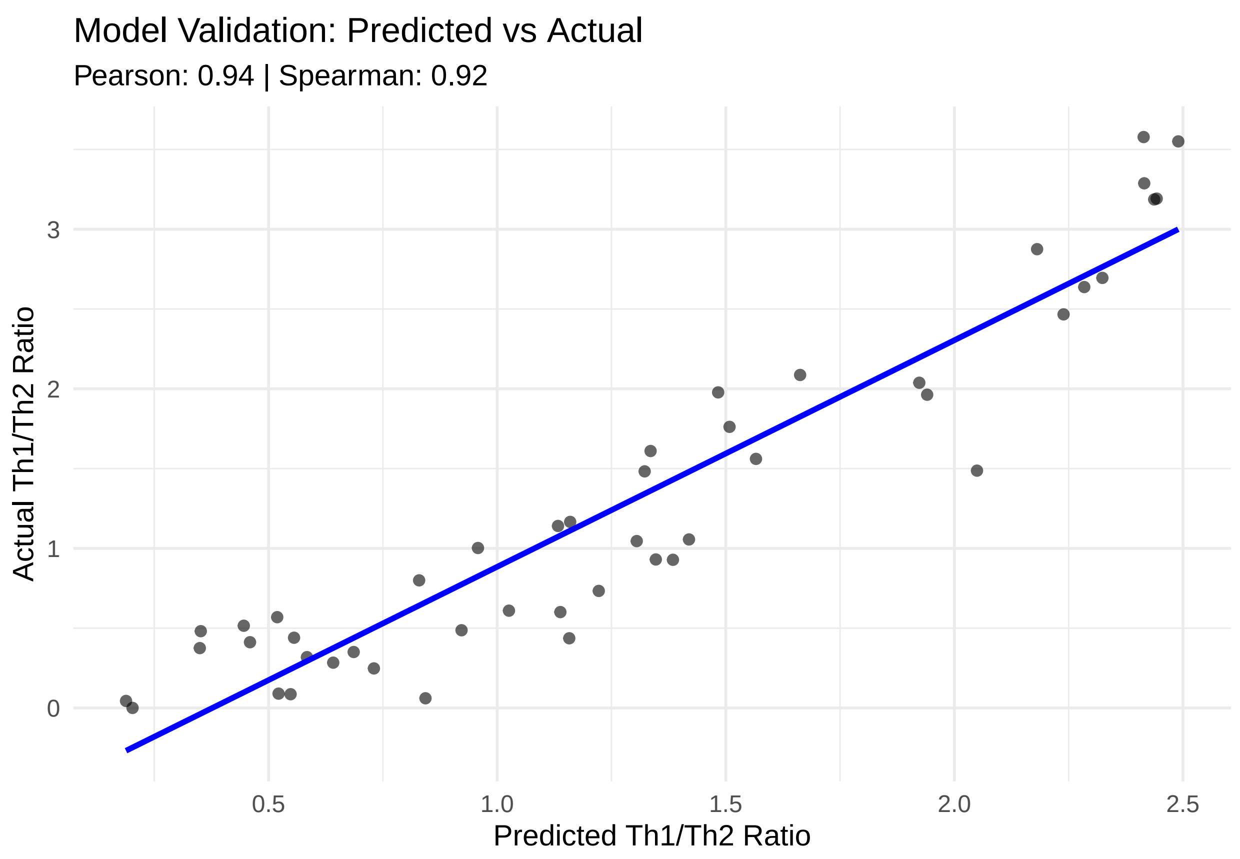
```
## Pearson Correlation:  0.9412563
```

```
cat("Spearman Correlation: ", spearman_cor, "\n")
```

```
## Spearman Correlation:  0.9232809
```

```
# Plot validation results
ggplot(data = data.frame(
  Predicted = train_predictions,
  Actual = trainDF$Th1_Th2_ratio
), aes(x = Predicted, y = Actual)) +
  geom_point(alpha = 0.6) +
  geom_smooth(method = "lm", color = "blue", se = FALSE) +
  labs(
    title = "Model Validation: Predicted vs Actual",
    subtitle = paste0("Pearson: ", round(pearson_cor, 2), " | Spearman: ", round(spearman_cor, 2)),
    x = "Predicted Th1/Th2 Ratio",
    y = "Actual Th1/Th2 Ratio"
  ) +
  theme_minimal()
```

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



```r
# Predict and rank for 2023
yhat <- predict(auto_fit, new_data = x23DF)$.pred
rhat <- rank(-1 * yhat, ties.method = "first")

# Print rankings
print(cbind(rownames(x23DF), rhat))
```

```
##             rhat
##  [1,] "119" "37"
##  [2,] "120" "3"
##  [3,] "121" "34"
##  [4,] "122" "2"
##  [5,] "123" "8"
##  [6,] "124" "18"
##  [7,] "125" "4"
##  [8,] "126" "44"
##  [9,] "127" "52"
## [10,] "128" "20"
## [11,] "129" "39"
## [12,] "130" "28"
## [13,] "132" "47"
## [14,] "133" "14"
## [15,] "134" "13"
## [16,] "135" "23"
## [17,] "136" "12"
## [18,] "137" "16"
## [19,] "138" "7"
## [20,] "139" "19"
## [21,] "140" "25"
## [22,] "141" "29"
## [23,] "142" "21"
## [24,] "143" "5"
## [25,] "144" "11"
## [26,] "145" "31"
## [27,] "146" "27"
## [28,] "147" "43"
## [29,] "149" "10"
## [30,] "150" "49"
## [31,] "151" "1"
## [32,] "152" "42"
## [33,] "153" "36"
## [34,] "154" "32"
## [35,] "156" "15"
## [36,] "157" "6"
## [37,] "158" "40"
## [38,] "159" "30"
## [39,] "160" "54"
## [40,] "161" "35"
## [41,] "162" "17"
## [42,] "163" "45"
## [43,] "164" "46"
## [44,] "165" "33"
## [45,] "166" "48"
## [46,] "167" "38"
## [47,] "168" "9"
## [48,] "169" "51"
## [49,] "170" "50"
## [50,] "171" "24"
## [51,] "172" "53"
## [52,] "131" "26"
## [53,] "148" "22"
## [54,] "155" "41"
```

```
# Save the rankings into a revised submission template
data <- read_tsv(file.path(workDir, "3rdChallengeSubmissionTemplate_revised.tsv"))

ranking_df <- data.frame(
  SubjectID = as.numeric(rownames(x23DF)),
  "4.1) IFNG/IL5-Polarization-D30-Rank" = rhat,
  check.names = FALSE
)

data <- data %>%
  mutate(
    `4.1) IFNG/IL5-Polarization-D30-Rank` = ifelse(
      SubjectID %in% ranking_df$SubjectID,
      ranking_df$`4.1) IFNG/IL5-Polarization-D30-Rank`[match(SubjectID, ranking_df$SubjectID)],
      `4.1) IFNG/IL5-Polarization-D30-Rank`
    )
```

```r
  )

# Write updated data back to file
write_tsv(data, file.path(workDir, "3rdChallengeSubmissionTemplate_revised.tsv"))
```

```r
# End H2O session and display session info
agua::h2o_end()
sessionInfo()
```

```
## R version 4.3.1 (2023-06-16 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 11 x64 (build 22631)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=English_United States.utf8
## [2] LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.utf8
##
## time zone: America/Los_Angeles
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] impute_1.76.0    agua_0.1.4       parsnip_1.2.1    GSVA_1.50.5
##  [5] knitr_1.49       here_1.0.1       glmnet_4.1-8     Matrix_1.5-4.1
##  [9] kableExtra_1.4.0 lubridate_1.9.3  forcats_1.0.0    stringr_1.5.1
## [13] purrr_1.0.2      readr_2.1.5      tibble_3.2.1     tidyverse_2.0.0
## [17] tidyr_1.3.1      ggplot2_3.5.1    dplyr_1.1.3
##
## loaded via a namespace (and not attached):
##   [1] jsonlite_1.8.9              rstudioapi_0.17.1
##   [3] shape_1.4.6.1               magrittr_2.0.3
##   [5] farver_2.1.2                rmarkdown_2.29
##   [7] zlibbioc_1.46.0             vctrs_0.6.3
##   [9] memoise_2.0.1               DelayedMatrixStats_1.24.0
##  [11] RCurl_1.98-1.16             htmltools_0.5.8.1
##  [13] S4Arrays_1.2.1              dials_1.3.0
##  [15] curl_6.0.1                  Rhdf5lib_1.24.2
##  [17] SparseArray_1.2.4           rhdf5_2.46.1
##  [19] parallelly_1.39.0           cachem_1.1.0
##  [21] lifecycle_1.0.4             iterators_1.0.14
##  [23] pkgconfig_2.0.3             rsvd_1.0.5
##  [25] R6_2.5.1                    fastmap_1.2.0
##  [27] future_1.34.0               GenomeInfoDbData_1.2.11
##  [29] MatrixGenerics_1.14.0       tune_1.2.1
##  [31] digest_0.6.37               colorspace_2.1-0
##  [33] furrr_0.3.1                 AnnotationDbi_1.64.1
##  [35] S4Vectors_0.38.2            rprojroot_2.0.4
##  [37] irlba_2.3.5.1               GenomicRanges_1.54.1
##  [39] RSQLite_2.3.7               beachmat_2.18.1
##  [41] labeling_0.4.3              yardstick_1.3.1
##  [43] fansi_1.0.4                 timechange_0.3.0
##  [45] mgcv_1.8-42                 httr_1.4.7
##  [47] abind_1.4-8                 compiler_4.3.1
##  [49] bit64_4.5.2                 withr_3.0.2
##  [51] BiocParallel_1.34.2         DBI_1.2.3
##  [53] HDF5Array_1.30.1            lava_1.8.0
##  [55] MASS_7.3-60                 DelayedArray_0.28.0
##  [57] tools_4.3.1                 future.apply_1.11.3
##  [59] nnet_7.3-19                 glue_1.6.2
##  [61] nlme_3.1-162                h2o_3.44.0.3
##  [63] rhdf5filters_1.14.1         grid_4.3.1
##  [65] generics_0.1.3              recipes_1.1.0
##  [67] gtable_0.3.6                tzdb_0.4.0
##  [69] class_7.3-22                rsample_1.2.1
##  [71] data.table_1.16.2           hms_1.1.3
##  [73] BiocSingular_1.18.0         ScaledMatrix_1.10.0
##  [75] xml2_1.3.6                  utf8_1.2.3
##  [77] XVector_0.40.0              BiocGenerics_0.48.1
##  [79] foreach_1.5.2               pillar_1.9.0
##  [81] vroom_1.6.5                 splines_4.3.1
##  [83] lhs_1.2.0                   lattice_0.21-8
##  [85] survival_3.5-5              bit_4.5.0
##  [87] annotate_1.80.0             tidyselect_1.2.1
##  [89] SingleCellExperiment_1.24.0 Biostrings_2.70.3
##  [91] IRanges_2.34.1              SummarizedExperiment_1.32.0
##  [93] svglite_2.1.3               stats4_4.3.1
##  [95] xfun_0.48                   Biobase_2.62.0
##  [97] hardhat_1.4.0               timeDate_4041.110
##  [99] matrixStats_1.4.1           stringi_1.8.4
## [101] DiceDesign_1.10             yaml_2.3.10
## [103] workflows_1.1.4             evaluate_1.0.1
## [105] codetools_0.2-19            graph_1.80.0
## [107] cli_3.6.1                   rpart_4.1.19
## [109] xtable_1.8-4                systemfonts_1.1.0
## [111] munsell_0.5.1               Rcpp_1.0.11
## [113] GenomeInfoDb_1.38.8         globals_0.16.3
## [115] png_0.1-8                   XML_3.99-0.17
## [117] parallel_4.3.1              gower_1.0.1
## [119] blob_1.2.4                  sparseMatrixStats_1.14.0
## [121] bitops_1.0-9                listenv_0.9.1
## [123] GPfit_1.0-8                 viridisLite_0.4.2
## [125] GSEABase_1.64.0             ipred_0.9-15
## [127] prodlim_2024.06.25          scales_1.3.0
## [129] crayon_1.5.3                rlang_1.1.1
## [131] KEGGREST_1.42.0
```