Loading Libraries

```r
library(dplyr)
library(caret)
library(randomForest)
library(pROC)
library(ggplot2)
library(glmnet)
library(ROSE)
library(biomaRt)
```

Loading Raw Counts Data and Normalization

The raw RNA-seq count data is loaded into the environment. The dataset consists of the **top 200 differentially expressed genes** across **1,224 samples**, including primary tumor (TP) and matched normal tissue (NT) samples. The expression values are currently in raw count format and require normalization prior to downstream analysis. The gene expression matrix was normalized using **z-score scaling**, whereby each gene's expression values were centered and scaled to have a mean of zero and a standard deviation of one.

```r
# This is a samples x genes dataframe
ml_data <- read.csv("raw_counts_degs_all_samples.csv", check.names = FALSE)
rownames(ml_data) <- ml_data[, 1]
ml_data <- ml_data[, -1]

head(ml_data[, 1:5])
```

```
##                                type ENSG00000099953 ENSG00000119927
## TCGA.BH.A18H.01A.11R.A12D.07   TP            7602             462
## TCGA.E2.A14P.01A.31R.A12D.07   TP            9053            2278
## TCGA.AN.A04A.01A.21R.A034.07   TP           75586             838
## TCGA.AQ.A0Y5.01A.11R.A14M.07   TP           43220            1208
## TCGA.A8.A07I.01A.11R.A00Z.07   TP           78859            2216
## TCGA.E9.A1R4.01A.21R.A14D.07   TP           15722             966
##                                ENSG00000117650 ENSG00000230838
## TCGA.BH.A18H.01A.11R.A12D.07               637             131
## TCGA.E2.A14P.01A.31R.A12D.07              1505             149
## TCGA.AN.A04A.01A.21R.A034.07               845             604
## TCGA.AQ.A0Y5.01A.11R.A14M.07               320            1112
## TCGA.A8.A07I.01A.11R.A00Z.07              2526             939
## TCGA.E9.A1R4.01A.21R.A14D.07               775             232
```

```r
# Normalization first
ml_data <- as.data.frame(t(ml_data)) # needs genes x samples dataframe
ml_labels <- as.vector(unlist(ml_data[1, ])) # Storing the labels separately
ml_counts <- ml_data[-1, ] # Counts matrix
ml_counts <- as.data.frame(
  apply(ml_counts, 2, function(x) as.numeric(trimws(x))),
  row.names = rownames(ml_counts)
) # Converting to numeric
```

```r
ml_counts <- scale(ml_counts) # z-score scaling

ml_data <- as.data.frame(t(ml_counts))
ml_data$type <- as.factor((ml_labels))
ml_data <- ml_data %>% dplyr::select(type, everything())
```

Stratified Train-Test Split (70:30) and Data Balancing

The normalized gene expression data were partitioned into training and testing sets using a 70:30 split. Due to class imbalance between primary tumor (TP) and normal tissue (NT) samples, **Synthetic Minority Over-sampling Technique (SMOTE)** was applied to the training set to ensure a more balanced class distribution and reduces model bias during classification.

```r
set.seed(345)
# Creating the index to split
trainIndex <- createDataPartition(
  ml_data$type,
  p = 0.7,
  list = FALSE
)
trainData <- ml_data[trainIndex, ]
testData  <- ml_data[-trainIndex, ]

# Data sample split before balancing
cat(
  "Sample Distribution Before Balancing (Training):\n",
  paste(capture.output(table(trainData$type)), collapse = "\n"),
  "\n\n",
  "Sample Distribution for Test Data:\n",
  paste(capture.output(table(testData$type)), collapse = "\n"),
  "\n\n"
)
```

```
## Sample Distribution Before Balancing (Training):
##
##  NT  TP
##  80 778
##
##  Sample Distribution for Test Data:
##
##  NT  TP
##  33 333
```

```r
# SMOTE for Data Balancing on training data only
set.seed(199)
trainData_bal <- ROSE(type ~ ., data = trainData, N = 2000, p = 0.5)$data

cat(
  "Sample Distribution After Balancing (Training):\n",
  paste(capture.output(table(trainData_bal$type)), collapse = "\n"),
  "\n"
)
```

```
## Sample Distribution After Balancing (Training):
##
##   TP   NT
## 1023  977
```

Machine Learning Models

Two machine learning models, namely **Least Absolute Shrinkage and Selection Operator (LASSO)** and **Random Forest**, were applied to the processed dataset for tumor (TP) versus normal (NT) classification. Each model was independently trained and evaluated on the generated training and testing sets. Following model implementation, the top 10 potential biomarkers were extracted from each model.

OPTION 1 : LASSO - Regularized Regression

In this approach, LASSO regression was applied to the training dataset with **5-fold cross-validation** to identify the optimal regularization parameter (λ). The model was fit along the regularization path, and **cross-validation error** and **coefficient convergence curves** were plotted. Model performance was subsequently evaluated on the testing dataset, and the **top 10 potential diagnostic biomarkers** were selected based on the absolute magnitude of their non-zero coefficients.

```r
# Defining the Training variables
x_lasso <- as.matrix(trainData_bal[, -1])
y_lasso <- as.factor(trainData_bal[, 1])

# Defining the Testing variables
x_test <- as.matrix(testData[, -1])
y_test <- as.factor(testData[, 1])

y_lasso_num <- ifelse(y_lasso == "TP", 1, 0) # Convert the labels to numeric for glmnet

cat("x_lasso:", dim(x_lasso), "-", ncol(x_lasso), "genes,", nrow(x_lasso), "samples\n")
```

```
## x_lasso: 2000 200 - 200 genes, 2000 samples
```

```r
cat("y_lasso:", length(y_lasso), "samples\n")
```

```
## y_lasso: 2000 samples
```

```r
set.seed(345)
# Running 5-fold cross-validation LASSO
cv_lasso <- cv.glmnet(
  x = x_lasso,
  y = y_lasso_num,
  alpha = 1,
  family = "binomial", # Two classes (TP and NT)
  nfolds = 5,
  parallel = FALSE)

# Obtaining the best lambdas that can be used to predict in the testing dataset

cat("Best lambda (min):", cv_lasso$lambda.min) #minimum cross-validation error
```

```
## Best lambda (min): 0.0004332688
```

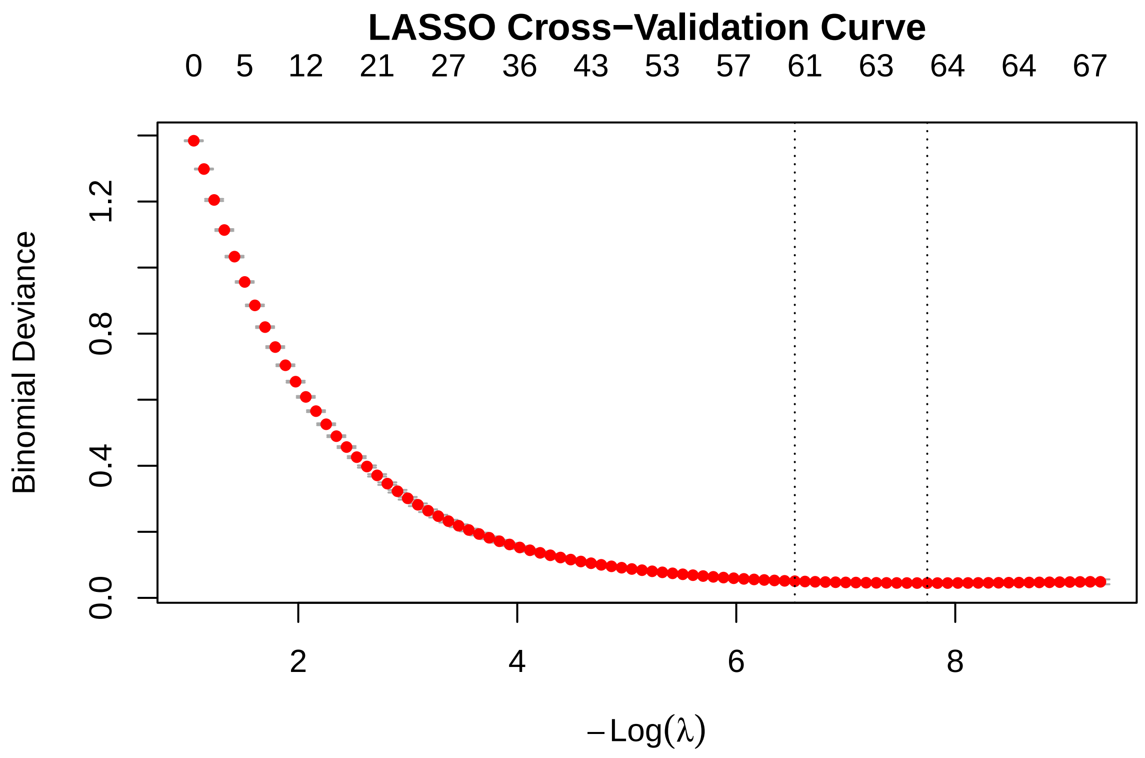
```r
cat("Best lambda (1se):", cv_lasso$lambda.1se) # 1 standard error away from minimum
```

```
## Best lambda (1se): 0.001452145
```

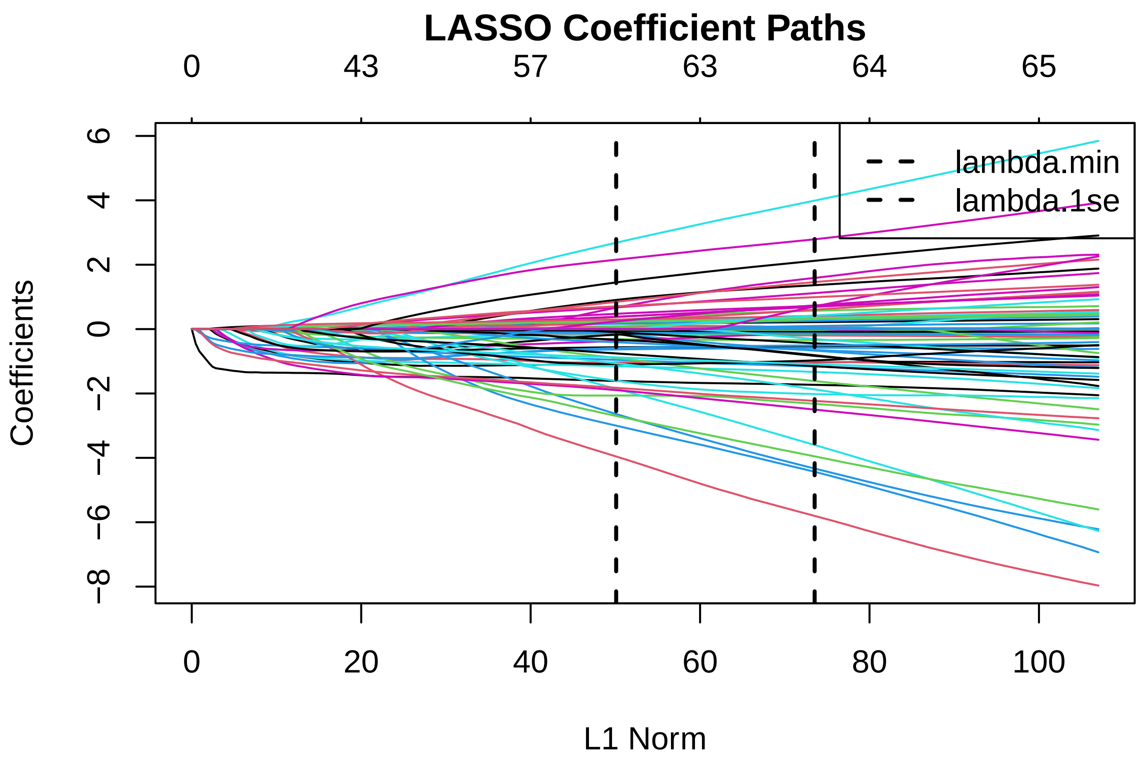
```r
# Fitting the lasso model to a path
lasso_model <- glmnet(
  x = x_lasso,
  y = y_lasso_num,
  alpha = 1,
  family = "binomial"
)

## PLOTS ##
par(mar = c(5, 5, 5, 2)) # plotting both in the same panel

## Plot_1: Cross-validation curve
plot(cv_lasso, main = "LASSO Cross-Validation Curve")
```



```r
## Plot_2: Convergence curve (coefficient paths vs L1 norm)
plot(lasso_model, xvar = "norm", label = FALSE,
     main = "LASSO Coefficient Paths")
abline(v = sum(abs(coef(lasso_model, s = cv_lasso$lambda.min)[-1])),
       col = "black", lty = 2, lwd = 2)
abline(v = sum(abs(coef(lasso_model, s = cv_lasso$lambda.1se)[-1])),
       col = "black", lty = 2, lwd = 2)
legend("topright", legend = c("lambda.min", "lambda.1se"),
       col = c("black", "black"), lty = 2, lwd = 2)
```



```
# Reset plotting parameters
par(mfrow = c(1, 1))

# Model validation tests
y_test_num <- ifelse(y_test == "TP", 1, 0) # Convert to numeric for glmnet prediction comparison

# Predict probabilities
lasso_pred_prob <- predict(
  cv_lasso,
  newx = x_test,
  s = "lambda.1se",
  type = "class",
  type.measure = "auc"
)

# Convert probabilities to class labels
lasso_pred_class <- ifelse(lasso_pred_prob > 0.5, "TP", "NT")
lasso_pred_class <- as.factor(lasso_pred_class)
```

```r
# Validation Metrics
# Confusion Matrix
cm <- confusionMatrix(
  lasso_pred_class,
  y_test,
  positive = "TP"
)
print(cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NT  TP
##         NT  33  10
##         TP   0 323
##
##                Accuracy : 0.9727
##                  95% CI : (0.9503, 0.9868)
##     No Information Rate : 0.9098
##     P-Value [Acc > NIR] : 1.26e-06
##
##                   Kappa : 0.8535
##
##  Mcnemar's Test P-Value : 0.004427
##
##             Sensitivity : 0.9700
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.7674
##              Prevalence : 0.9098
##          Detection Rate : 0.8825
##    Detection Prevalence : 0.8825
##       Balanced Accuracy : 0.9850
##
##        'Positive' Class : TP
##
```

```r
# ROC
roc_obj <- roc(
  y_test,
  as.numeric(lasso_pred_prob),
  levels = c("NT", "TP"),
  direction = "<"
)
# F1 score and AUC
f1_score <- cm$byClass["F1"]
cat("F1 Score for LASSO =", round(f1_score, 4), "\n")
```

```
## F1 Score for LASSO = 0.9848
```

```r
auc_value <- auc(roc_obj)
cat("Test AUC for LASSO =", auc_value, "\n")
```

```
## Test AUC for LASSO = 0.984985
```

```r
# Obtaining Biomarkers
# Get non-zero coefficients (selected features)
lasso_coef <- coef(cv_lasso, s = "lambda.1se")
selected_features <- which(lasso_coef[-1,] != 0)
cat("Number of features selected:", length(selected_features))
```

```
## Number of features selected: 61
```

```r
if(!is.null(colnames(x_lasso))) {
  selected_gene_names <- colnames(x_lasso)[selected_features]
}

lasso_biomarkers <- selected_gene_names[1:10]
print(lasso_biomarkers)
```

```
##  [1] "ENSG00000099953" "ENSG00000119927" "ENSG00000022267" "ENSG00000060718"
##  [5] "ENSG00000079308" "ENSG00000203805" "ENSG00000133800" "ENSG00000105974"
##  [9] "ENSG00000095637" "ENSG00000134962"
```

```r
# Getting the Gene Names for the Ensembl IDs
annotate_genes <- function(biomarkers, model){
  # Biomart Connection
  mart <- useEnsembl(
    biomart = "genes",
    dataset = "hsapiens_gene_ensembl"
  )

  # Gene Annotations
  gene_annotation <- getBM(
    attributes = c("ensembl_gene_id", "hgnc_symbol"),
    filters = "ensembl_gene_id",
    values = biomarkers,
    mart = mart
  )

  # Create annotation dataframe correctly
  ann_df <- data.frame(
    ensembl_gene_id = biomarkers,
    stringsAsFactors = FALSE
  ) %>%
    left_join(gene_annotation, by = "ensembl_gene_id")

  # Removing unannotated genes
  ann_df <- ann_df %>%
    filter(hgnc_symbol != "" & !is.na(hgnc_symbol))

  # Get top 10 biomarkers
  top10_biomarkers <- ann_df$hgnc_symbol
  cat("Top 10", model, "Biomarkers:\n")
  print(top10_biomarkers)
  cat("\n")

  # Return the annotated dataframe
  return(ann_df)
}

lasso_biomarkers_df <- annotate_genes(lasso_biomarkers, "LASSO")
```

```
## Top 10 LASSO Biomarkers:
##  [1] "MMP11"   "GPAM"    "FHL1"    "COL11A1" "TNS1"    "PLPP4"   "LYVE1"
##  [8] "CAV1"    "SORBS1"  "KLB"
```

OPTION 2: RANDOM FOREST

In this approach, a Random Forest classifier was trained on the previously generated training dataset. A **5-fold cross-validation** strategy was implemented and **hyperparameter tuning** was performed for mtry and ntree to improve model performance, minimize overfitting and improve generalization. The optimized model was validated on the independent testing dataset. Finally, the **top 10 potential diagnostic biomarkers** were identified based on **ranked variable importance scores** derived from the model.

```r
# Defining the Training variables
x_rf <- as.matrix(trainData_bal[, -1])
y_rf <- as.factor(trainData_bal[, 1])

# Defining the Testing variables
x_test <- as.matrix(testData[, -1])
y_test <- as.factor(testData[, 1])

set.seed(345)
# Training Model using 5-fold CV
ctrl <- trainControl(
  method = "cv",
  number = 5,
  classProbs = TRUE,
  summaryFunction = twoClassSummary,
  savePredictions = "final"
)

# Performing Hyperparameter Tuning (mtry and ntree)
## mtry
mtry_grid <- expand.grid(
  mtry = seq(2, floor(sqrt(ncol(x_rf))), by = 1)
) # Testing mtry values from 2 up to √(number of features)
## Obtaining best mtry value for our training set
rf_tuned_mtry <- train(
  x = x_rf,
  y = y_rf,
  method = "rf",
  metric = "ROC",
  trControl = ctrl,
  tuneGrid = mtry_grid,
  ntree = 500,
  importance = TRUE
)

best_mtry <- rf_tuned_mtry$bestTune$mtry
cat("Best mtry =", best_mtry, "\n")
```

```
## Best mtry = 2
```

```r
## ntree
ntree_values <- c(100, 200, 300, 500, 700) # Different values for the number of trees in the Random For
results_ntree <- data.frame()
### Running a loop on the ntree_values to and calculating the ROC for each
for (nt in ntree_values) {
  set.seed(123)
  model <- train(
    x = x_rf,
    y = y_rf,
    method = "rf",
    metric = "ROC",
    trControl = ctrl,
    tuneGrid = data.frame(mtry = best_mtry),
    ntree = nt
  )
  results_ntree <- rbind(results_ntree,
                         data.frame(ntree = nt,
                                    ROC = max(model$results$ROC)))
}

best_ntree <- results_ntree$ntree[which.max(results_ntree$ROC)]
cat("Best ntree =", best_ntree, "\n")
```

```
## Best ntree = 100
```

```r
# Using the above found parameters, tuning our final RF model
set.seed(345)

final_rf <- randomForest(
  x = x_rf,
  y = y_rf,
  mtry = best_mtry,
  ntree = best_ntree,
  importance = TRUE
)

# Predicting on Test Data
pred_class <- predict(final_rf, x_test)

# Evaluation of Model Performance
## Confusion matrix
cm <- confusionMatrix(pred_class, y_test)

## ROC + AUC + F1
```

```r
pred_prob <- predict(final_rf, x_test, type = "prob")[,2]
roc_obj <- roc(y_test, pred_prob)

f1_score <- cm$byClass["F1"]
cat("F1 Score for RF =", round(f1_score, 4), "\n")
```

```
## F1 Score for RF = 0.825
```

```r
auc_value <- auc(roc_obj)
cat("Test AUC for RF =", auc_value, "\n")
```

```
## Test AUC for RF = 0.99818
```

```r
# Obtaining Biomarkers
## Extract Variable Importance
var_imp <- varImp(final_rf, scale = TRUE)
var_imp_df <- as.data.frame(var_imp$TP)
var_imp_df$Gene <- rownames(var_imp)

# Use TP column (same as NT)
var_imp_df$Overall <- var_imp_df$`var_imp$TP`

# Keep only Gene + Overall
var_imp_df <- var_imp_df  %>% dplyr::select(Gene, Overall)
head(var_imp_df)
```

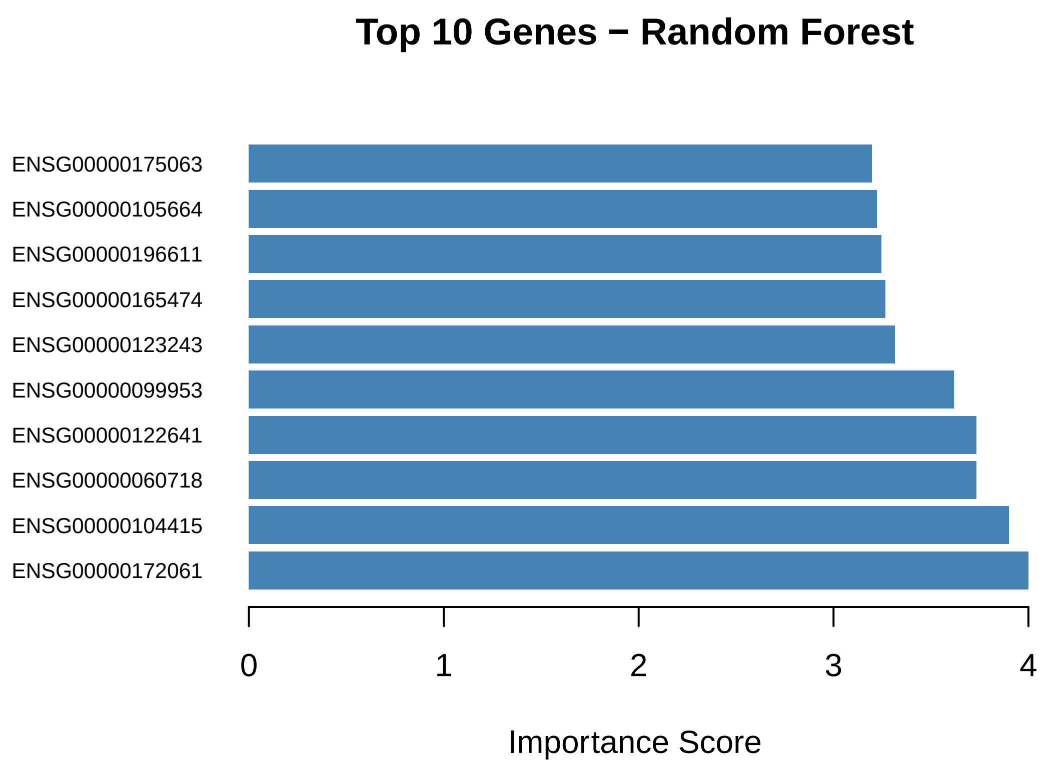
```
##              Gene  Overall
## 1 ENSG00000099953 3.617764
## 2 ENSG00000119927 2.726231
## 3 ENSG00000117650 1.951398
## 4 ENSG00000230838 2.057463
## 5 ENSG00000022267 2.611405
## 6 ENSG00000060718 3.733674
```

```r
## Arrange by importance (ROC)
var_imp_df <- var_imp_df %>%
  arrange(desc(Overall))

# Plot 1: VarImp plot for top 10 features
top10 <- var_imp_df %>%
  dplyr::slice(1:10)

par(mar = c(5, 10, 5, 2))

barplot(height = top10$Overall,
        names.arg = top10$Gene,
        main = "Top 10 Genes - Random Forest",
        xlab = "Importance Score",
        col = "steelblue",
        las = 1,  # Horizontal labels
        horiz = TRUE,  # Horizontal bars
        cex.names = 0.7,  # Smaller font for gene names
        border = NA)
```



```r
## Select the Top 10 features based on descending importance
rf_biomarkers <- top10[, 1]

## Using the function to annotate genes from LASSO section
rf_biomarkers_df <- annotate_genes(rf_biomarkers, "Random Forest")
```

```
## Top 10 Random Forest Biomarkers:
##  [1] "LRRC15"  "CCN4"    "COL11A1" "INHBA"   "MMP11"   "ITIH5"   "GJB2"
##  [8] "MMP1"    "COMP"    "UBE2C"
```

```r
sessionInfo()
```

```
## R version 4.5.2 (2025-10-31)
## Platform: x86_64-pc-linux-gnu
## Running under: Ubuntu 22.04.4 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.10.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0  LAPACK version 3.10.0
##
## locale:
##  [1] LC_CTYPE=en_IN.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_IN.UTF-8        LC_COLLATE=en_IN.UTF-8
##  [5] LC_MONETARY=en_IN.UTF-8    LC_MESSAGES=en_IN.UTF-8
##  [7] LC_PAPER=en_IN.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_IN.UTF-8 LC_IDENTIFICATION=C
##
## time zone: Asia/Kolkata
## tzcode source: system (glibc)
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] biomaRt_2.66.0       ROSE_0.0-4           glmnet_4.1-10
##  [4] Matrix_1.7-4         pROC_1.19.0.1        randomForest_4.7-1.2
##  [7] caret_7.0-1          lattice_0.22-5       ggplot2_4.0.2
## [10] dplyr_1.2.0
##
## loaded via a namespace (and not attached):
##  [1] DBI_1.2.3            httr2_1.2.2          rlang_1.1.7
##  [4] magrittr_2.0.4       otel_0.2.0           e1071_1.7-17
##  [7] compiler_4.5.2       RSQLite_2.4.6        png_0.1-8
## [10] vctrs_0.7.1          reshape2_1.4.5       stringr_1.6.0
## [13] pkgconfig_2.0.3      shape_1.4.6.1        crayon_1.5.3
## [16] fastmap_1.2.0        dbplyr_2.5.2         XVector_0.50.0
## [19] rmarkdown_2.30       prodlim_2025.04.28   purrr_1.2.1
## [22] bit_4.6.0            xfun_0.56            cachem_1.1.0
## [25] progress_1.2.3       recipes_1.3.1        blob_1.3.0
## [28] parallel_4.5.2       prettyunits_1.2.0    R6_2.6.1
## [31] stringi_1.8.7        RColorBrewer_1.1-3   parallelly_1.46.1
## [34] rpart_4.1.24         lubridate_1.9.5      Rcpp_1.1.1
## [37] Seqinfo_1.0.0        iterators_1.0.14     knitr_1.51
## [40] future.apply_1.20.1  IRanges_2.44.0       splines_4.5.2
## [43] nnet_7.3-20          timechange_0.4.0     tidyselect_1.2.1
## [46] rstudioapi_0.18.0    yaml_2.3.12          timeDate_4052.112
## [49] codetools_0.2-19     curl_7.0.0           listenv_0.10.0
## [52] tibble_3.3.1         plyr_1.8.9           Biobase_2.70.0
## [55] withr_3.0.2          KEGGREST_1.50.0      S7_0.2.1
## [58] evaluate_1.0.5       future_1.69.0        survival_3.8-3
## [61] proxy_0.4-29         BiocFileCache_3.0.0  xml2_1.5.2
## [64] Biostrings_2.78.0    pillar_1.11.1        filelock_1.0.3
## [67] foreach_1.5.2        stats4_4.5.2         generics_0.1.4
## [70] hms_1.1.4            S4Vectors_0.48.0     scales_1.4.0
## [73] globals_0.19.0       class_7.3-23         glue_1.8.0
## [76] tools_4.5.2          data.table_1.18.2.1  ModelMetrics_1.2.2.2
## [79] gower_1.0.2          grid_4.5.2           ipred_0.9-15
## [82] AnnotationDbi_1.72.0 nlme_3.1-168         cli_3.6.5
## [85] rappdirs_0.3.4       lava_1.8.2           gtable_0.3.6
## [88] digest_0.6.39        BiocGenerics_0.56.0  farver_2.1.2
## [91] memoise_2.0.1        htmltools_0.5.9      lifecycle_1.0.5
## [94] hardhat_1.4.2        httr_1.4.8           bit64_4.6.0-1
## [97] MASS_7.3-65
```