Principle Component Analysis of Nasal Epithelium

2024-10-4

R Markdown

This is an R Markdown document.

```
#install.packages("limma")
library(limma)
library(dplyr)
```

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

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

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

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ forcats   1.0.0     ✔ readr     2.1.5
## ✔ ggplot2   3.5.1     ✔ stringr   1.5.1
## ✔ lubridate 1.9.3     ✔ tibble    3.2.1
## ✔ purrr     1.0.2     ✔ tidyr     1.3.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(GEOquery)
```

```
## Loading required package: Biobase
## Loading required package: BiocGenerics
##
## Attaching package: 'BiocGenerics'
##
## The following objects are masked from 'package:lubridate':
##
##     intersect, setdiff, union
##
## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union
##
## The following object is masked from 'package:limma':
##
##     plotMA
##
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
##
## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which.max, which.min
##
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
##
## Setting options('download.file.method.GEOquery'='auto')
## Setting options('GEOquery.inmemory.gpl'=FALSE)
```

```
library(pheatmap)


file_path <- "~/Downloads/GDS3309_full.soft"
raw_data <- readLines(file_path)
head(raw_data, 100)
```

```
start_line <- grep("!dataset_table_begin", raw_data)
my_data <- read.delim(file_path, skip = start_line, header = TRUE)

expression_data <- my_data %>% select(starts_with("GSM"))

identifier_column <- my_data["IDENTIFIER"]


# Normalize the data
normalized_data <- normalizeBetweenArrays(as.matrix(expression_data))

expression_df <- cbind(identifier_column, expression_data)

# Sample GSM code status mapping as a vector
gsm_status <- c(
  "GSM227868" = "Never Smoker",
  "GSM227870" = "Never Smoker",
  "GSM227871" = "Never Smoker",
  "GSM227874" = "Never Smoker",
  "GSM227876" = "Never Smoker",
  "GSM227877" = "Never Smoker",
  "GSM227878" = "Never Smoker",
  "GSM227880" = "Never Smoker",
  "GSM227869" = "Current Smoker",
  "GSM227872" = "Current Smoker",
  "GSM227873" = "Current Smoker",
  "GSM227875" = "Current Smoker",
  "GSM227879" = "Current Smoker",
  "GSM227881" = "Current Smoker",
  "GSM227882" = "Current Smoker"
)

# Ensure that the GSM codes in expression_df match the order of gsm_status
gsm_codes <- colnames(expression_df) # Exclude the 'IDENTIFIER' column

# Create a new row of smoker status matching the GSM codes
smoker_status <- sapply(gsm_codes, function(gsm) gsm_status[gsm])

# Add the smoker status as a new row to the dataframe
expression_df_with_status <- rbind(smoker_status, expression_df)


# Reshaping data for plotting purposes
long_df <- expression_df %>%
    pivot_longer(cols = starts_with("GSM"),
                 names_to = "Sample",
                 values_to = "Expression")

#Decided to take top 25 genes for better visualization purposes
#Steps to take top 25 genes:
#One, Calculate the mean expression for each gene
top_genes <- long_df %>%
```

```
    group_by(IDENTIFIER) %>%
    summarize(MeanExpression = mean(Expression, na.rm = TRUE)) %>%
    top_n(25, MeanExpression) %>%
    pull(IDENTIFIER)

#Two, Filter the long dataframe to include only the top genes
filtered_long_df <- long_df %>%
    filter(IDENTIFIER %in% top_genes)

#Three,plot with filtered data
#Scatter plot with geom_point

ggplot(filtered_long_df, aes(x = IDENTIFIER, y = Expression, color = Sample)) +
  geom_point() +
  scale_x_discrete(expand = expansion(mult = c(0.001, 0.01))) +
  theme(axis.text.x = element_text(angle = 90,
                                   vjust = 0.5,
                                   hjust = 1,
                                   margin = margin(t = 15))) +
  labs(title = "Top 25 Genes by Expression Across Samples",
       x = "Gene Names",
       y = "Expression Levels") +
  coord_flip() + # flip the coordinates
  theme_minimal()
```

```
#Alternative visualization: Table
#I've commented this out as I prefer the scatter plot
#top_25_genes <- filtered_long_df %>%
#  select(IDENTIFIER, Sample, Expression) %>%
#  arrange(IDENTIFIER, Sample)
#print(top_25_genes)
#save to CSV
#write.csv(top_25_genes_table, "top_25_genes_expression_table.csv", row.names = FALSE)



#Principle Component Analysis

# Scale the expression data
expression_data_scaled <- scale(expression_data)

# Replace infinite values with 0
expression_data_scaled[is.infinite(expression_data_scaled)] <- 0

# Replace missing (NA) values with 0
expression_data_scaled[is.na(expression_data_scaled)] <- 0

# Perform PCA
pca_result <- prcomp(expression_data_scaled, center = TRUE, scale. = TRUE)

# Print PCA summary to check the result
summary(pca_result)
```

```
## Importance of components:
##                           PC1     PC2    PC3     PC4    PC5     PC6     PC7
## Standard deviation     3.7379 0.67714 0.3529 0.33913 0.2511 0.22946 0.22106
## Proportion of Variance 0.9315 0.03057 0.0083 0.00767 0.0042 0.00351 0.00326
## Cumulative Proportion  0.9315 0.96202 0.9703 0.97799 0.9822 0.98570 0.98896
##                            PC8     PC9    PC10    PC11    PC12   PC13    PC14
## Standard deviation     0.20233 0.18110 0.15753 0.14637 0.12975 0.1098 0.09324
## Proportion of Variance 0.00273 0.00219 0.00165 0.00143 0.00112 0.0008 0.00058
## Cumulative Proportion  0.99169 0.99387 0.99553 0.99696 0.99808 0.9989 0.99946
##                           PC15
## Standard deviation     0.08979
## Proportion of Variance 0.00054
## Cumulative Proportion  1.00000
```

```
# Create a data frame with PCA results
pca_df <- as.data.frame(pca_result$x)



colnames(pca_df)[1:15] <- colnames(expression_data)


#pca_df <- pca_df %>% select(-Sample)


gsm_samples <- colnames(pca_df)[1:15]

# Create a new row corresponding to SmokerStatus using the gsm_status mapping
smoker_status_row <- gsm_status[gsm_samples]

# We use rbind to add the SmokerStatus row to the top of the dataframe
pca_df <- rbind(SmokerStatus = smoker_status_row, pca_df)

#pca_df <- pca_df %>% slice(-22280)

#print(pca_df)


# The SmokerStatus is the first row, so let's separate it from the rest of the PCA data
smoker_status <- pca_df[1, 1:15]  # Extract the SmokerStatus row (first row)
pca_data <- pca_df[-1, 1:15]


smoker_status <- as.vector(as.matrix(pca_df[1, 1:15]))  # Convert first row to a vector

# Convert smoker_status into a factor
smoker_status <- as.factor(smoker_status)

# Remove the SmokerStatus row from pca_df to keep only PCA data for plotting
pca_data <- pca_df[-1, 1:15]

# Convert PCA data into numeric for plotting
pca_data <- as.data.frame(lapply(pca_data, as.numeric))

# Transpose the PCA data so that samples are in rows and PCs in columns
pca_plot_df <- as.data.frame(t(pca_data))

# Add SmokerStatus as a column
pca_plot_df$SmokerStatus <- smoker_status

# Plot PCA with ggplot2, using PC1 and PC2, and color points by SmokerStatus
ggplot(pca_plot_df, aes(x = V1, y = V2, color = SmokerStatus)) +
  geom_point(size = 3) +
  labs(title = "PCA of Gene Expression Data: PC1 vs PC2",
       x = "Principal Component 1",
       y = "Principal Component 2") +
```

```
  theme_minimal() +
  scale_color_manual(values = c("Never Smoker" = "blue", "Current Smoker" = "red"))
```