Gene module analysis using optimized parameters

Mikayla, Mustafa, Yibo

2023-05-07

Table of Contents

[Introduction 1](#_Toc134452954)

[InferCNV based on immune cells and reference genome. 1](#_Toc134452955)

[Calculate genomic instability score for each cell 7](#_Toc134452956)

[Calculate the variance of cancer cell’s genomic instability score as results 9](#_Toc134452957)

[Visualization 11](#_Toc134452958)

# Introduction

This report is totally original without taking code from the github repo of the original paper. As this report also includes all the output for your references, we have also provided a table of content so that you might skip some of very tedious output and go straight to some code.

# InferCNV based on immune cells and reference genome.

library(infercnv)

library(Matrix)

# List of patient IDs

patient\_ids <- c("CID3586", "CID3838", "CID3921", "CID3941", "CID3946", "CID3948", "CID3963", "CID4040", "CID4066", "CID4067", "CID4290A", "CID4398", "CID4461", "CID4463", "CID4465", "CID4471", "CID4495", "CID4513", "CID4515", "CID4523", "CID4530N", "CID4535", "CID44041", "CID44971", "CID44991", "CID45171")

# Function to create annotations file and generate raw count matrix

create\_annotations\_file\_and\_generate\_matrix <- function(metadata\_csv, count\_matrix, barcodes\_tsv, genes\_tsv) {

metadata <- read.csv(metadata\_csv)

annotations <- data.frame(metadata[,1], metadata$celltype\_major)

colnames(annotations) <- c("", "cell\_group")

# Read sparse count matrix

sparse\_count\_matrix <- readMM(count\_matrix)

colnames(sparse\_count\_matrix) <- read.table(barcodes\_tsv, col.names = c("barcode"))$barcode

rownames(sparse\_count\_matrix) <- read.table(genes\_tsv, col.names = c("gene"))$gene

# Convert sparse matrix to dense matrix

dense\_count\_matrix <- as.matrix(sparse\_count\_matrix)

return(list(annotations = annotations, dense\_count\_matrix = dense\_count\_matrix))

}

calculate\_genome\_instability\_score <- function(expr\_data, reference\_indices, observation\_indices) {

# Get reference and observation data

references <- expr\_data[, unlist(reference\_indices)]

observations <- expr\_data[, unlist(observation\_indices)]

# Calculate changes

changes <- observations - rowMeans(references)

# Scale changes between -1 and 1

scaled\_changes <- 2 \* (changes - min(changes)) / (max(changes) - min(changes)) - 1

# Calculate instability scores

instability\_scores <- rowMeans(scaled\_changes^2)

return(instability\_scores)

}

# Loop over patient IDs

genome\_instability\_scores\_list <- list()

for (patient\_id in patient\_ids) {

# Define file paths

metadata\_csv <- paste0("./GSE176078\_RAW/", patient\_id, "/metadata.csv")

count\_matrix <- paste0("./GSE176078\_RAW/", patient\_id, "/count\_matrix\_sparse.mtx")

barcodes\_tsv <- paste0("./GSE176078\_RAW/", patient\_id, "/count\_matrix\_barcodes.tsv")

genes\_tsv <- paste0("./GSE176078\_RAW/", patient\_id, "/count\_matrix\_genes.tsv")

# Create annotations file and generate dense count matrix

results <- create\_annotations\_file\_and\_generate\_matrix(metadata\_csv, count\_matrix, barcodes\_tsv, genes\_tsv)

annotations <- results$annotations

dense\_count\_matrix <- results$dense\_count\_matrix

annotations\_file <- paste0("./GSE176078\_RAW/", patient\_id, "/", patient\_id, "\_annotations.txt")

write.table(annotations, file = annotations\_file, sep = "\t", row.names = FALSE, col.names = FALSE, quote = FALSE)

tryCatch({

# Create inferCNV object with custom parameters

infercnv\_obj = CreateInfercnvObject(raw\_counts\_matrix = dense\_count\_matrix,

annotations\_file = annotations\_file,

delim = "\t",

gene\_order\_file = "gene\_order\_file.txt",

ref\_group\_names = c("T-cells", "B-cells", "Endothelial"))

# Set scipen option before running inferCNV

options(scipen = 100)

# Filter genes with a mean count of less than 0.1 across all cells

filtered\_counts\_matrix = dense\_count\_matrix[rowMeans(dense\_count\_matrix) >= 0.1, ]

# Run inferCNV with custom parameters

infercnv\_obj = infercnv::run(infercnv\_obj,

cutoff = 1.3, # dynamic threshold of 1.3 standard deviations from the mean

out\_dir = paste0("infercnv\_output\_", patient\_id),

cluster\_by\_groups = TRUE,

denoise = TRUE,

HMM = TRUE,

analysis\_mode = "subclusters",

window\_length = 100) # 100 gene sliding window

},

error = function(e) {

cat("Error processing patient\_id:", patient\_id, "with 'T-cells', 'B-cells', and 'Endothelial' as reference\n")

cat("Error message:", conditionMessage(e), "\n")

cat("Re-initializing with 'T-cells' and 'B-cells' as reference\n")

tryCatch({

infercnv\_obj = CreateInfercnvObject(raw\_counts\_matrix = dense\_count\_matrix,

annotations\_file = annotations\_file,

delim = "\t",

gene\_order\_file = "gene\_order\_file.txt",

ref\_group\_names = c("T-cells", "B-cells"))

# Set scipen option before running inferCNV

options(scipen = 100)

# Filter genes with a mean count of less than 0.1 across all cells

filtered\_counts\_matrix = dense\_count\_matrix[rowMeans(dense\_count\_matrix) >= 0.1, ]

# Run inferCNV with custom parameters

infercnv\_obj = infercnv::run(infercnv\_obj,

cutoff = 1.3, # dynamic threshold of 1.3 standard deviations from the mean

out\_dir = paste0("infercnv\_output\_", patient\_id),

cluster\_by\_groups = TRUE,

denoise = TRUE,

HMM = TRUE,

analysis\_mode = "subclusters",

window\_length = 100) # 100 gene sliding window

}, error = function(e) {

cat("Error processing patient\_id:", patient\_id, "with 'T-cells' and 'B-cells' as reference\n")

cat("Error message:", conditionMessage(e), "\n")

cat("Re-initializing with 'T-cells' as reference\n")

tryCatch({

infercnv\_obj = CreateInfercnvObject(raw\_counts\_matrix = dense\_count\_matrix,

annotations\_file = annotations\_file,

delim = "\t",

gene\_order\_file = "gene\_order\_file.txt",

ref\_group\_names = c("T-cells"))

# Set scipen option before running inferCNV

options(scipen = 100)

# Filter genes with a mean count of less than 0.1 across all cells

filtered\_counts\_matrix = dense\_count\_matrix[rowMeans(dense\_count\_matrix) >= 0.1, ]

# Run inferCNV with custom parameters

infercnv\_obj = infercnv::run(infercnv\_obj,

cutoff = 1.3, # dynamic threshold of 1.3 standard deviations from the mean

out\_dir = paste0("infercnv\_output\_", patient\_id),

cluster\_by\_groups = TRUE,

denoise = TRUE,

HMM = TRUE,

analysis\_mode = "subclusters",

window\_length = 100) # 100 gene sliding window

}, error = function(e) {

cat("Error processing patient\_id:", patient\_id, "with 'T-cells' as reference\n")

cat("Error message:", conditionMessage(e), "\n")

})

})

})

# Calculate genome instability scores

genome\_instability\_scores <- calculate\_genome\_instability\_score(infercnv\_obj@expr.data,

infercnv\_obj@reference\_grouped\_cell\_indices,

infercnv\_obj@observation\_grouped\_cell\_indices)

genome\_instability\_scores\_list[[patient\_id]] <- genome\_instability\_scores

}

# Calculate genomic instability score for each cell

library(Matrix)

# Function to calculate genomic instability score

genomic\_instability\_score <- function(count\_matrix, genomic\_scores) {

squared\_genomic\_scores <- genomic\_scores^2

weighted\_counts <- count\_matrix \* squared\_genomic\_scores

rowMeans(weighted\_counts)

}

# Main implementation

patient\_ids <- c("CID3586", "CID3838", "CID3921", "CID3941", "CID3946", "CID3948", "CID3963", "CID4040", "CID4066", "CID4067", "CID4290A", "CID4398", "CID4461", "CID4463", "CID4465", "CID4471", "CID4495", "CID4513", "CID4515", "CID4523", "CID4530N", "CID4535", "CID44041", "CID44971", "CID44991", "CID45171")

instability\_scores\_all\_patients <- list()

for (i in 1:length(patient\_ids)) {

patient\_id <- patient\_ids[i]

cat("Dealing with", i, "of", length(patient\_ids), "patients:", patient\_id, "\n")

count\_matrix\_barcodes\_path <- paste0("./GSE176078\_RAW/", patient\_id, "/count\_matrix\_barcodes.tsv")

count\_matrix\_genes\_path <- paste0("./GSE176078\_RAW/", patient\_id, "/count\_matrix\_genes.tsv")

count\_matrix\_sparse\_path <- paste0("./GSE176078\_RAW/", patient\_id, "/count\_matrix\_sparse.mtx")

count\_matrix <- readMM(file = count\_matrix\_sparse\_path)

barcodes <- read.delim(count\_matrix\_barcodes\_path, header = FALSE, col.names = c("barcode"))

genes <- read.delim(count\_matrix\_genes\_path, header = FALSE, col.names = c("gene"))

colnames(count\_matrix) <- barcodes$barcode

rownames(count\_matrix) <- genes$gene

# Get the genomic scores for the current patient

genomic\_scores <- genome\_instability\_scores\_list[[patient\_id]]

# Subset count\_matrix to have only the genes in genomic\_scores

count\_matrix <- count\_matrix[rownames(count\_matrix) %in% names(genomic\_scores), ]

# Sort genomic\_scores to match the order of genes in count\_matrix

genomic\_scores <- genomic\_scores[rownames(count\_matrix)]

# Calculate genomic instability scores

instability\_scores <- genomic\_instability\_score(t(count\_matrix), genomic\_scores)

# Store the results in the list

instability\_scores\_df <- data.frame(instability\_score = instability\_scores)

rownames(instability\_scores\_df) <- colnames(count\_matrix)

instability\_scores\_all\_patients[[patient\_id]] <- instability\_scores\_df

}

# Calculate the variance of cancer cell’s genomic instability score as results

# Initialize a list to store the variance for each patient

variance\_list <- list()

# Loop through each patient

for (patient\_id in patient\_ids) {

# Define file paths for the patient's data

count\_matrix\_barcodes\_path <- paste0("./GSE176078\_RAW/", patient\_id, "/count\_matrix\_barcodes.tsv")

count\_matrix\_genes\_path <- paste0("./GSE176078\_RAW/", patient\_id, "/count\_matrix\_genes.tsv")

count\_matrix\_sparse\_path <- paste0("./GSE176078\_RAW/", patient\_id, "/count\_matrix\_sparse.mtx")

metadata\_path <- paste0("./GSE176078\_RAW/", patient\_id, "/metadata.csv")

# Load count matrix and metadata

count\_matrix <- readMM(file = count\_matrix\_sparse\_path)

barcodes <- read.delim(count\_matrix\_barcodes\_path, header = FALSE, col.names = c("barcode"))

genes <- read.delim(count\_matrix\_genes\_path, header = FALSE, col.names = c("gene"))

metadata <- read.csv(metadata\_path, row.names = 1)

# Filter cells based on the celltype\_major column

cells\_to\_keep <- metadata$celltype\_major %in% c("Cancer Epithelial", "CAFs")

# Set the colnames and rownames of the count matrix

colnames(count\_matrix) <- barcodes$barcode

rownames(count\_matrix) <- genes$gene

# Get the indices of the cells to keep

cells\_to\_keep\_indices <- which(cells\_to\_keep)

# Subset the count matrix using the cells\_to\_keep\_indices

count\_matrix <- count\_matrix[, cells\_to\_keep\_indices]

# Subset the metadata using cells\_to\_keep

metadata <- metadata[cells\_to\_keep, , drop = FALSE]

# Get the genomic instability scores for all cells in the current patient

all\_instability\_scores <- instability\_scores\_all\_patients[[patient\_id]]

# Subset the genomic instability scores using cells\_to\_keep\_indices

cancer\_instability\_scores <- all\_instability\_scores[cells\_to\_keep\_indices, , drop = FALSE]

# Calculate the variance of the cancer cell genomic instability scores for the current patient

variance <- var(cancer\_instability\_scores$instability\_score)

# Store the variance in the variance\_list

variance\_list[[patient\_id]] <- variance

}

# Combine the variance\_list into a data frame

variance\_df <- data.frame(variance = unlist(variance\_list))

rownames(variance\_df) <- patient\_ids

# Visualization

library(ggplot2)

# Convert the variance data frame into a format suitable for ggplot2

variance\_df$patient\_index <- 1:nrow(variance\_df)

# Create a function to display a subset of patient IDs

display\_subset\_labels <- function(patient\_ids, step = 2) {

sapply(1:length(patient\_ids), function(i) {

if (i %% step == 0) {

return(patient\_ids[i])

} else {

return("")

}

})

}

# Create the bar plot

ggplot(variance\_df, aes(x = patient\_index, y = variance)) +

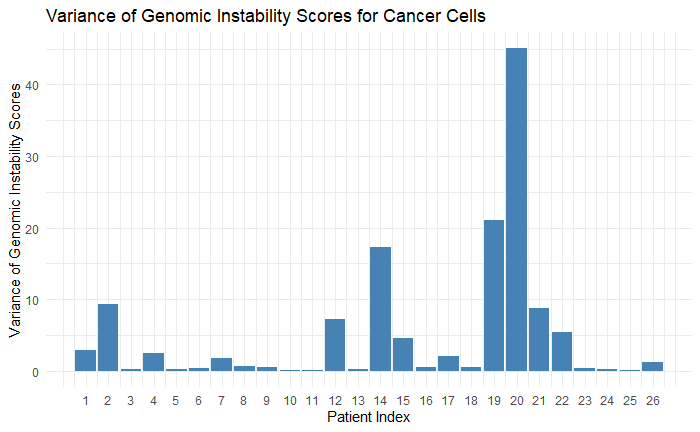
geom\_bar(stat = "identity", fill = "steelblue") +

scale\_x\_continuous(breaks = 1:length(patient\_ids), labels = 1:length(patient\_ids)) +

theme(axis.text.x = element\_text(angle = 90, hjust = 1, vjust = 0.5)) +

labs(x = "Patient Index", y = "Variance of Genomic Instability Scores", title = "Variance of Genomic Instability Scores for Cancer Cells") +

theme\_minimal()



# Store the result

# Convert the variance\_list to a data frame

variance\_df <- data.frame(

patient\_id = patient\_ids,

variance = variance\_list

)

# Write the data frame to a CSV file

write.csv(variance\_df, file = "CNV\_variance.csv", row.names = FALSE)