FA-582 Assignment 2 Code

Naveen Mathews Renji - 20016323

Problem 1

```
# Load required libraries
library(httr)
library(rvest)
library(tidyverse)

# Define the URL
url <- "https://en.wikipedia.org/wiki/List_of_S%26P_500_companies"

# Fetch the webpage using httr
response <- GET(url)

# Parse the webpage content
webpage <- read_html(content(response, "text"))

# Extract the table (Note: This code is for educational purposes only)
sp500_table <- html_nodes(webpage, "table")[[1]] %>% html_table()

# Print a preview of the table
print(head(sp500_table))

# Perform Exploratory Data Analysis (EDA)
# Summary statistics
summary(sp500_table)

str(sp500_table)
#formatting Date
sp500_table$`Date added` <- as.Date(sp500_table$`Date added`, format="%Y-%m-%d")
str(sp500_table)
#formatting founded
# Extract the first set of four digits as the year
sp500_table$Founded_cleaned <- stringr::str_extract(sp500_table$Founded, "\\d{4}")

# Convert the cleaned "Founded" column to numeric
sp500_table$Founded_numeric <- as.numeric(sp500_table$Founded_cleaned)

#Distribution of Companies by Founding Year
ggplot(sp500_table, aes(x=Founded_numeric)) +
  geom_histogram(binwidth=10, aes(fill=..count..)) +
```

```
  ggtitle("Distribution of Companies by Founding Year") +
  xlab("Founded Year") +
  ylab("Number of Companies")

# Create a histogram of the 'Date added' column, ignoring NA values
ggplot(data = sp500_table, aes(x = `Date added`)) +
  geom_histogram(binwidth = 365.25, fill = 'blue', alpha = 0.7) +
  scale_x_date(breaks = "10 years", date_labels = "%Y") +
  labs(title = "Companies Added to S&P 500 Over the Years",
       x = "Year",
       y = "Number of Companies Added")


# checking companies adding when sp was founded in 1957
selected_companies <- sp500_table[sp500_table$`Date added` == "1957-03-04", ]
# Print the companies
print(selected_companies$Security)


#Years Taken to Get Added to S&P 500 by Year Founded

sp500_table$Year_added <- as.integer(format(as.Date(sp500_table$`Date added`), "%Y"))
sp500_table$Years_to_add <- sp500_table$Year_added - sp500_table$Founded_numeric


library(ggplot2)
ggplot(sp500_table, aes(x=Years_to_add)) +
  geom_histogram(binwidth=5, fill="blue", alpha=0.7) +
  ggtitle("Time to Get Added to S&P 500") +
  xlab("Years from Founded to Added") +
  ylab("Number of Companies")


#Number of Companies by GICS Sector
ggplot(sp500_table, aes(x=`GICS Sector`)) +
  geom_bar(aes(fill=`GICS Sector`), position='dodge') +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Number of Companies by GICS Sector") +
  xlab("GICS Sector") +
  ylab("Number of Companies")

#Top 10 Most Popular Sub-Industries
# Count the occurrences of each sub-industry
sub_industry_counts <- table(sp500_table$`GICS Sub-Industry`)

# Sort in descending order and take the top 5 sub-industries
top_10_sub_industries <- names(sort(sub_industry_counts, decreasing=TRUE)[1:10])
```

```
# Filter the data frame to only include these top 10 sub-industries
filtered_sp500_table <- sp500_table[sp500_table$`GICS Sub-Industry` %in%
top_10_sub_industries, ]

# Plot using ggplot2
ggplot(filtered_sp500_table, aes(x=`GICS Sub-Industry`)) +
  geom_bar(aes(fill=`GICS Sub-Industry`), position='dodge') +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Top 10 Most Popular Sub-Industries") +
  xlab("GICS Sub-Industry") +
  ylab("Number of Companies")

#Top 10 Company Headquarters Locations
# Count the occurrences of each headquarters location
hq_counts <- table(sp500_table$`Headquarters Location`)

# Sort in descending order and take the top 10 headquarters locations
top_10_hq <- names(sort(hq_counts, decreasing=TRUE)[1:10])

# Filter the data frame to only include these top 10 headquarters locations
filtered_sp500_table_hq <- sp500_table[sp500_table$`Headquarters Location` %in%
top_10_hq, ]

# Plot using ggplot2
ggplot(filtered_sp500_table_hq, aes(x=`Headquarters Location`)) +
  geom_bar(aes(fill=`Headquarters Location`), position='dodge') +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Top 10 Company Headquarters Locations") +
  xlab("Headquarters Location") +
  ylab("Number of Companies")


#Summary Statistics for the entire data
summary(sp500_table)
```

Problem 2

1. L- Norm and Minkowski

```
# Load the required libraries
library(readr)
library(dplyr)
library(reshape2)
```

```
library(tidyr)

# Load the data from the CSV files
fundamentals_df <- read_csv("fundamentals.csv")

quantitative_cols <- c("After Tax ROE", "Cash Ratio", "Current Ratio", "Operating Margin",
"Pre-Tax Margin",
                      "Pre-Tax ROE", "Profit Margin", "Quick Ratio", "Total Assets", "Total
Liabilities", "Earnings Per Share")

# Filter Data for Specific Year (2013)
fundamentals_2013 <- filter(fundamentals_df, substr(`Period Ending`, 1, 4) == "2013")


fundamentals_2013 <- fundamentals_2013 %>%
  rowwise() %>%
  mutate(missing_or_zero = sum(across(all_of(quantitative_cols), ~is.na(.) | . == 0), na.rm =
TRUE)) %>%
  ungroup()

# Sorting the DataFrame based on the missing_or_zero column and then picking the first
100 tickers.
top_100_tickers <- fundamentals_2013 %>%
  arrange(missing_or_zero) %>%
  head(100) %>%
  pull('Ticker Symbol')

# Selecting rows for the top 100 tickers
fundamentals_2013_subset <- filter(fundamentals_2013, `Ticker Symbol` %in%
top_100_tickers)

# selected columns
selected_columns <- c("Ticker Symbol", "After Tax ROE", "Cash Ratio", "Current Ratio",
"Operating Margin", "Pre-Tax Margin", "Pre-Tax ROE", "Profit Margin", "Quick Ratio", "Total
Assets", "Total Liabilities", "Earnings Per Share")
fundamentals_2013_subset_selected <- dplyr::select(fundamentals_2013_subset,
all_of(selected_columns))

# Function for Lp-norm
lpnorm <- function(x, y, p) {
  return (sum(abs(x - y)^p)^(1 / p))
}

# Function for weighted Minkowski distance
weights <- c(0.0941, 0.0941, 0.0941, 0.0824, 0.0706, 0.0706, 0.1059, 0.1059, 0.0941,
0.0588) #calculation explained in the report
weighted_minkowski <- function(x, y, p) {
  return (sum(weights * abs(x - y)^p)^(1 / p))
```

```
}

# Calculate distances and sort and return as data frame
calc_and_sort_distances <- function(df, dist_func, p = NULL) {
  dist_list <- list()
  for (i in 1:(nrow(df) - 1)) {
    for (j in (i + 1):nrow(df)) {
      ticker1 <- as.character(df[i, 'Ticker Symbol'])
      ticker2 <- as.character(df[j, 'Ticker Symbol'])
      pair <- paste(ticker1, ticker2, sep = "-")
      if (is.null(p)) {
        dist <- dist_func(df[i, -1], df[j, -1])
      } else {
        dist <- dist_func(df[i, -1], df[j, -1], p)
      }
      dist_list[[pair]] <- dist
    }
  }
  sorted_list <- sort(unlist(dist_list))
  sorted_df <- data.frame(Ticker_Pair = names(sorted_list), Distance = sorted_list)
  return (sorted_df)
}

# Lp-norm calculations
for (p in c(1, 2, 3, 10)) {
  sorted_distances_df <- calc_and_sort_distances(fundamentals_2013_subset_selected,
lpnorm, p)
  print(paste("Top 10 and Bottom 10 for Lp-norm with p =", p))
  print("Top 10:")
  print(head(sorted_distances_df, 10))
  print("Bottom 10:")
  print(tail(sorted_distances_df, 10))
}

# Minkowski distance calculations
sorted_distances_minkowski_df <-
calc_and_sort_distances(fundamentals_2013_subset_selected, weighted_minkowski, 2)
print("Top 10 and Bottom 10 for Weighted Minkowski Distance")
print("Top 10:")
print(head(sorted_distances_minkowski_df, 10))
print("Bottom 10:")
print(tail(sorted_distances_minkowski_df, 10))
```

2. Match based distance

```
# Load the required libraries
```

```
library(readr)
library(dplyr)

# Load the data from the CSV files
fundamentals_df <- read_csv("fundamentals.csv")
securities_df <- read_csv("securities.csv")
print("Structure of fundamentals_df:")
str(fundamentals_df)

# Filter Data for Specific Year (2013)
fundamentals_2013 <- filter(fundamentals_df, substr(`Period Ending`, 1, 4) == "2013")
quantitative_cols <- c("After Tax ROE", "Cash Ratio", "Current Ratio", "Operating Margin",
"Pre-Tax Margin",
                      "Pre-Tax ROE", "Profit Margin", "Quick Ratio", "Total Assets", "Total
Liabilities", "Earnings Per Share")

fundamentals_2013 <- fundamentals_2013 %>%
  rowwise() %>%
  mutate(missing_or_zero = sum(across(all_of(quantitative_cols), ~is.na(.) | . == 0), na.rm =
TRUE)) %>%
  ungroup()

# Sorting the DataFrame based on the missing_or_zero column and then picking the first
100 tickers.
top_100_tickers <- fundamentals_2013 %>%
  arrange(missing_or_zero) %>%
  head(100) %>%
  pull('Ticker Symbol')

fundamentals_2013_subset <- filter(fundamentals_2013, `Ticker Symbol` %in%
top_100_tickers)

selected_columns <- c("Ticker Symbol", "After Tax ROE", "Cash Ratio", "Current Ratio",
"Operating Margin", "Pre-Tax Margin", "Pre-Tax ROE", "Profit Margin", "Quick Ratio", "Total
Assets", "Total Liabilities", "Earnings Per Share")
fundamentals_2013_subset_selected <- dplyr::select(fundamentals_2013_subset,
all_of(selected_columns))

bucketize <- function(column, n_buckets) {
  breaks <- quantile(column, probs = seq(0, 1, length.out = n_buckets + 1))
  cut(column, breaks = breaks, labels = FALSE, include.lowest = TRUE)
}

match_score <- function(row1, row2) {
  sum(row1 == row2)
}

# Bucketize each column, 3 buckets
```

```
bucketized_data <- as.data.frame(lapply(fundamentals_2013_subset_selected[-1],
bucketize, n_buckets=3))

# match-based similarity matrix
n <- nrow(bucketized_data)
match_similarity_matrix <- matrix(0, n, n)
rownames(match_similarity_matrix) <- fundamentals_2013_subset_selected$`Ticker
Symbol`
colnames(match_similarity_matrix) <- fundamentals_2013_subset_selected$`Ticker Symbol`

for (i in 1:(n - 1)) {
  for (j in (i + 1):n) {
    score <- match_score(bucketized_data[i,], bucketized_data[j,])
    match_similarity_matrix[i, j] <- score
    match_similarity_matrix[j, i] <- score
  }
}

match_similarity_df <- as.data.frame(as.table(match_similarity_matrix))

# Filter out the diagonal and duplicate entries
match_similarity_df <- match_similarity_df %>% filter(Var1 != Var2)

# Sort by similarity score (Freq) in descending order for top 10 pairs
sorted_df_top <- match_similarity_df %>% arrange(desc(Freq))
print("Top 10 most similar pairs based on match score:")
print(head(sorted_df_top, 10))

# Sort by similarity score (Freq) in ascending order for bottom 10 pairs
sorted_df_bottom <- match_similarity_df %>% arrange(Freq)
print("Bottom 10 least similar pairs based on match score:")
print(head(sorted_df_bottom, 10))
```

3. Mahalanobis Distance

```
# Load necessary libraries
library(readr)
library(dplyr)
library(MASS)
library(stats)

# Load the data from the CSV files
fundamentals_df <- read_csv("fundamentals.csv")

quantitative_cols <- c("After Tax ROE", "Cash Ratio", "Current Ratio", "Operating Margin",
"Pre-Tax Margin",
                       "Pre-Tax ROE", "Profit Margin", "Quick Ratio", "Total Assets", "Total
Liabilities", "Earnings Per Share")
```

```
# Filter Data for Specific Year (2013)
fundamentals_2013 <- filter(fundamentals_df, substr(`Period Ending`, 1, 4) == "2013")


fundamentals_2013 <- fundamentals_2013 %>%
  rowwise() %>%
  mutate(missing_or_zero = sum(across(all_of(quantitative_cols), ~is.na(.) | . == 0), na.rm =
TRUE)) %>%
  ungroup()

# Sorting the DataFrame based on the missing_or_zero column and then picking the first
100 tickers.
top_100_tickers <- fundamentals_2013 %>%
  arrange(missing_or_zero) %>%
  head(100) %>%
  pull('Ticker Symbol')

fundamentals_2013_subset <- filter(fundamentals_2013, `Ticker Symbol` %in%
top_100_tickers)

selected_columns <- c("Ticker Symbol", "After Tax ROE", "Cash Ratio", "Current Ratio",
"Operating Margin", "Pre-Tax Margin", "Pre-Tax ROE", "Profit Margin", "Quick Ratio", "Total
Assets", "Total Liabilities", "Earnings Per Share")
fundamentals_2013_subset_selected <- dplyr::select(fundamentals_2013_subset,
all_of(selected_columns))

# Standardize the data (mean = 0, sd = 1)
# Exclude the 'Ticker Symbol' column when standardizing
fundamentals_2013_subset_selected_std <- fundamentals_2013_subset_selected
fundamentals_2013_subset_selected_std[-1] <-
scale(fundamentals_2013_subset_selected[-1])

# Compute the covariance matrix and its inverse
cov_matrix <- cov(fundamentals_2013_subset_selected_std[-1])
inv_cov_matrix <- solve(cov_matrix)

#  Mahalanobis distance
mahalanobis_distance <- function(x, y, inv_cov_matrix) {
  diff <- matrix(x - y, ncol = 1)  # Convert difference to column vector
  dist <- sqrt(t(diff) %*% inv_cov_matrix %*% diff)
  return(dist)
}

# Matrix to store Mahalanobis distances
mahalanobis_matrix <- matrix(NA,
                             nrow = nrow(fundamentals_2013_subset_selected_std),
                             ncol = nrow(fundamentals_2013_subset_selected_std))
```

```
rownames(mahalanobis_matrix) <- fundamentals_2013_subset_selected_std$`Ticker
Symbol`
colnames(mahalanobis_matrix) <- fundamentals_2013_subset_selected_std$`Ticker
Symbol`

# Compute Mahalanobis distance for each pair
for (i in 1:nrow(fundamentals_2013_subset_selected_std)) {
  for (j in 1:nrow(fundamentals_2013_subset_selected_std)) {
    x <- as.numeric(fundamentals_2013_subset_selected_std[i, -1])
    y <- as.numeric(fundamentals_2013_subset_selected_std[j, -1])
    mahalanobis_matrix[i, j] <- mahalanobis_distance(x, y, inv_cov_matrix)
  }
}
mahalanobis_df <- as.data.frame(as.table(mahalanobis_matrix))

sorted_mahalanobis <- mahalanobis_df %>% arrange(Freq)
top_10_mahalanobis <- head(sorted_mahalanobis, 10)
bottom_10_mahalanobis <- tail(sorted_mahalanobis, 10)

print("Top 10 Mahalanobis distances:")
print(top_10_mahalanobis)
print("Bottom 10 Mahalanobis distances:")
print(bottom_10_mahalanobis)
```

4. Categorical (Overlap, Inverse Frequency and Goodall)

```
library(dplyr)
library(readr)
library(tidyr)

# Read the data
securities_df <- read_csv("securities.csv")
categorical_cols <- c("GICS Sector", "GICS Sub Industry", "Address of Headquarters")

# Preprocess the data
securities_df <- securities_df %>%
  rowwise() %>%
  mutate(missing_or_zero = sum(across(all_of(categorical_cols), ~is.na(.) | . == 0), na.rm =
TRUE)) %>%
  ungroup()

# Select top 100 tickers
top_100_tickers <- securities_df %>%
  arrange(missing_or_zero) %>%
  head(100) %>%
  pull('Ticker symbol')
```

```
securities_subset <- filter(securities_df, `Ticker symbol` %in% top_100_tickers)

# Convert all categorical columns to character type
securities_subset <- securities_subset %>%
  mutate(across(all_of(categorical_cols), as.character))

# overlap similarity
overlap_similarity <- function(x, y) {
  if (is.na(x) || is.na(y)) return(0)
  if (x == y) return(1)
  return(0)
}

# inverse frequency similarity
inverse_frequency_similarity <- function(x, y, p) {
  if (is.na(x) || is.na(y)) return(0)
  if (x == y) return(1 / (p^2))
  return(0)
}

#Goodall similarity
goodall_similarity <- function(x, y, p) {
  if (is.na(x) || is.na(y)) return(0)
  if (x == y) return(1 - (p^2))
  return(0)
}

# Calculate pk(x) for each categorical column
pk_values <- lapply(categorical_cols, function(col) {
  table(securities_subset[[col]]) / nrow(securities_subset)
})

# empty matrix to store similarity scores
n <- nrow(securities_subset)
tickers <- as.character(securities_subset$`Ticker symbol`)

overlap_matrix <- matrix(0, n, n, dimnames=list(tickers, tickers))
inverse_frequency_matrix <- matrix(0, n, n, dimnames=list(tickers, tickers))
goodall_matrix <- matrix(0, n, n, dimnames=list(tickers, tickers))

# Calculate similarity for each pair of tickers
for (i in 1:n) {
  for (j in 1:n) {
    if (i == j) next # Skip diagonal values
    for (col in 1:length(categorical_cols)) {
      xi <- securities_subset[i, categorical_cols[col]]
      yi <- securities_subset[j, categorical_cols[col]]
      p <- pk_values[[col]][as.character(xi)]
```

```
      overlap_matrix[i, j] <- overlap_matrix[i, j] + overlap_similarity(xi, yi)
      inverse_frequency_matrix[i, j] <- inverse_frequency_matrix[i, j] +
inverse_frequency_similarity(xi, yi, p)
      goodall_matrix[i, j] <- goodall_matrix[i, j] + goodall_similarity(xi, yi, p)
    }
  }
}

# Convert matrices to data frames for easier sorting and viewing
overlap_df <- as.data.frame(as.table(overlap_matrix))
inverse_frequency_df <- as.data.frame(as.table(inverse_frequency_matrix))
goodall_df <- as.data.frame(as.table(goodall_matrix))

# Print top 10 and bottom 10 similarities, excluding diagonal and duplicate pairs
print("Top 10 Overlap Similarities")
print(head(overlap_df %>% filter(as.character(Var1) != as.character(Var2) &
as.character(Var1) < as.character(Var2)) %>% arrange(desc(Freq)), 10))

print("Bottom 10 Overlap Similarities")
print(head(overlap_df %>% filter(as.character(Var1) != as.character(Var2) &
as.character(Var1) < as.character(Var2)) %>% arrange(Freq), 10))

print("Top 10 Inverse Frequency Similarities")
print(head(inverse_frequency_df %>% filter(as.character(Var1) != as.character(Var2) &
as.character(Var1) < as.character(Var2)) %>% arrange(desc(Freq)), 10))

print("Bottom 10 Inverse Frequency Similarities")
print(head(inverse_frequency_df %>% filter(as.character(Var1) != as.character(Var2) &
as.character(Var1) < as.character(Var2)) %>% arrange(Freq), 10))

print("Top 10 Goodall Similarities")
print(head(goodall_df %>% filter(as.character(Var1) != as.character(Var2) &
as.character(Var1) < as.character(Var2)) %>% arrange(desc(Freq)), 10))

print("Bottom 10 Goodall Similarities")
print(head(goodall_df %>% filter(as.character(Var1) != as.character(Var2) &
as.character(Var1) < as.character(Var2)) %>% arrange(Freq), 10))
```

5. Overall Similarity

```
# Load the required libraries
library(readr)
library(dplyr)
library(reshape2)

# Load the data from the CSV files
fundamentals_df <- read_csv("fundamentals.csv")
```

```
securities_df <- read_csv("securities.csv")

# Filter Data for Specific Year (2013)
fundamentals_2013 <- filter(fundamentals_df, substr(`Period Ending`, 1, 4) == "2013")
quantitative_cols <- c("After Tax ROE", "Cash Ratio", "Current Ratio", "Operating Margin",
"Pre-Tax Margin",
                      "Pre-Tax ROE", "Profit Margin", "Quick Ratio", "Total Assets", "Total
Liabilities", "Earnings Per Share")

# Select tickers with least missing or zero values in quantitative columns
fundamentals_2013 <- fundamentals_2013 %>%
  rowwise() %>%
  mutate(missing_or_zero = sum(across(all_of(quantitative_cols), ~is.na(.) | . == 0), na.rm =
TRUE)) %>%
  ungroup()

top_100_tickers <- fundamentals_2013 %>%
  arrange(missing_or_zero) %>%
  head(100) %>%
  pull('Ticker Symbol')

fundamentals_2013_subset <- filter(fundamentals_2013, `Ticker Symbol` %in%
top_100_tickers)
securities_subset <- filter(securities_df, `Ticker symbol` %in% top_100_tickers)

selected_columns <- c("Ticker Symbol", "After Tax ROE", "Cash Ratio", "Current Ratio",
"Operating Margin", "Pre-Tax Margin", "Pre-Tax ROE", "Profit Margin", "Quick Ratio", "Total
Assets", "Total Liabilities", "Earnings Per Share")
fundamentals_2013_subset_selected <- dplyr::select(fundamentals_2013_subset,
all_of(selected_columns))

# Merge the two data frames
merged_df <- merge(fundamentals_2013_subset_selected, securities_subset, by.x = "Ticker
Symbol", by.y = "Ticker symbol")

df_numeric <- merged_df[, c("After Tax ROE", "Cash Ratio", "Current Ratio", "Operating
Margin", "Pre-Tax Margin",
                           "Pre-Tax ROE", "Profit Margin", "Quick Ratio", "Total Assets", "Total
Liabilities", "Earnings Per Share")]
df_categorical <- merged_df[, c("Security", "SEC filings", "GICS Sector","GICS Sub
Industry", "Address of Headquarters")]

lambda <- 0.8

#  categorical similarity (Overlap)
calc_cat_sim <- function(x, y) {
  return(ifelse(all(x == y), 1, 0))
}
```

```
# numerical similarity (Euclidean distance)
calc_num_sim <- function(x, y) {
  return(sum((x - y)^2))
}

num_tickers <- nrow(merged_df)
overall_sim_matrix <- matrix(0, nrow=num_tickers, ncol=num_tickers)
overall_sim_df <- data.frame(Var1 = character(), Var2 = character(), value = numeric())

ticker_symbols <- merged_df$`Ticker Symbol`

# overall similarity
for (i in 1:(num_tickers - 1)) {
  for (j in (i + 1):num_tickers) {
    if (i == j) {
      overall_sim_matrix[i, j] <- 1 #similarity score of 1 when tickers are  same
    } else {
      num_sim <- calc_num_sim(df_numeric[i, ], df_numeric[j, ])
      cat_sim <- calc_cat_sim(df_categorical[i, ], df_categorical[j, ])
      overall_sim <- lambda * num_sim + (1 - lambda) * cat_sim
      overall_sim_df <- rbind(overall_sim_df, data.frame(Var1 = ticker_symbols[i], Var2 =
ticker_symbols[j], value = overall_sim))
    }
  }
}

ticker_symbols <- merged_df$`Ticker Symbol`
rownames(overall_sim_matrix) <- ticker_symbols
colnames(overall_sim_matrix) <- ticker_symbols

# Rank the similarities
overall_sim_df$rank <- rank(-overall_sim_df$value)
top_10_sim <- overall_sim_df[order(-overall_sim_df$value), ][1:10,]
bottom_10_sim <- overall_sim_df[order(overall_sim_df$value), ][1:10,]
similarity_results <- (list(top_10 = top_10_sim, bottom_10 = bottom_10_sim))

print("Top 10 similarities:")
print(similarity_results$top_10)

print("Bottom 10 similarities:")
print(similarity_results$bottom_10)
```

6. Overall Normalised Similarity

```
# Load the required libraries
library(readr)
library(dplyr)
```

```
library(reshape2)

# Load the data from the CSV files
fundamentals_df <- read_csv("fundamentals.csv")
securities_df <- read_csv("securities.csv")

# Filter Data for Specific Year (2013)
fundamentals_2013 <- filter(fundamentals_df, substr(`Period Ending`, 1, 4) == "2013")
quantitative_cols <- c("After Tax ROE", "Cash Ratio", "Current Ratio", "Operating Margin",
"Pre-Tax Margin",
                      "Pre-Tax ROE", "Profit Margin", "Quick Ratio", "Total Assets", "Total
Liabilities", "Earnings Per Share")

# Select tickers with least missing or zero values in quantitative columns
fundamentals_2013 <- fundamentals_2013 %>%
  rowwise() %>%
  mutate(missing_or_zero = sum(across(all_of(quantitative_cols), ~is.na(.) | . == 0), na.rm =
TRUE)) %>%
  ungroup()

top_100_tickers <- fundamentals_2013 %>%
  arrange(missing_or_zero) %>%
  head(100) %>%
  pull('Ticker Symbol')

fundamentals_2013_subset <- filter(fundamentals_2013, `Ticker Symbol` %in%
top_100_tickers)
securities_subset <- filter(securities_df, `Ticker symbol` %in% top_100_tickers)

selected_columns <- c("Ticker Symbol", "After Tax ROE", "Cash Ratio", "Current Ratio",
"Operating Margin", "Pre-Tax Margin", "Pre-Tax ROE", "Profit Margin", "Quick Ratio", "Total
Assets", "Total Liabilities", "Earnings Per Share")
fundamentals_2013_subset_selected <- dplyr::select(fundamentals_2013_subset,
all_of(selected_columns))

# Merge the two data frames
merged_df <- merge(fundamentals_2013_subset_selected, securities_subset, by.x = "Ticker
Symbol", by.y = "Ticker symbol")

df_numeric <- merged_df[, c("After Tax ROE", "Cash Ratio", "Current Ratio", "Operating
Margin", "Pre-Tax Margin",
                           "Pre-Tax ROE", "Profit Margin", "Quick Ratio", "Total Assets", "Total
Liabilities", "Earnings Per Share")]
df_categorical <- merged_df[, c("Security", "SEC filings", "GICS Sector","GICS Sub
Industry", "Address of Headquarters")]

# Define the function
compute_normalized_similarity <- function(df_numeric, df_categorical, lambda) {
```

```
  # numerical similarity (Euclidean distance)
  calc_num_sim <- function(x, y, sigma_n) {
    return(sum(((x - y)^2) / sigma_n^2))
  }

  # categorical similarity (Overlap similarity)
  calc_cat_sim <- function(x, y, sigma_c) {
    return(sum(x == y) / sigma_c)
  }

  num_tickers <- nrow(df_numeric)
  ticker_symbols <- merged_df$`Ticker Symbol`
  overall_sim_df <- data.frame(Var1 = character(), Var2 = character(), value = numeric())
  overall_sim_matrix <- matrix(0, nrow=num_tickers, ncol=num_tickers)
  rownames(overall_sim_matrix) <- ticker_symbols
  colnames(overall_sim_matrix) <- ticker_symbols

  # Standard deviations for normalization
  sigma_n <- apply(df_numeric, 2, sd, na.rm = TRUE)  # column-wise standard deviation
  sigma_c <- length(unique(as.vector(as.matrix(df_categorical))))  # unique categories for
normalization

  # Calculate overall similarity
  for (i in 1:(num_tickers - 1)) {
    for (j in (i + 1):num_tickers) {
      num_sim <- calc_num_sim(df_numeric[i, ], df_numeric[j, ], sigma_n)
      cat_sim <- calc_cat_sim(df_categorical[i, ], df_categorical[j, ], sigma_c)
      overall_sim <- lambda * num_sim + (1 - lambda) * cat_sim
      overall_sim_df <- rbind(overall_sim_df, data.frame(Var1 = ticker_symbols[i], Var2 =
ticker_symbols[j], value = overall_sim))
    }
  }

  # Rank the similarities and return top and bottom 10
  overall_sim_df$rank <- rank(-overall_sim_df$value)
  top_10_sim <- overall_sim_df[order(-overall_sim_df$value), ][1:10,]
  bottom_10_sim <- overall_sim_df[order(overall_sim_df$value), ][1:10,]
  return(list(top_10 = top_10_sim, bottom_10 = bottom_10_sim))
}

lambda <- 0.1

similarity_results <- compute_normalized_similarity(df_numeric, df_categorical, lambda)
print("Top 10 similarities:")
print(similarity_results$top_10)
print("Bottom 10 similarities:")
print(similarity_results$bottom_10)
```