METRICS\_TECHNOLOGICAL\_DISTANCE

Daniel Jose Carreno Quintero

2024-08-21

options ( repos = c ( CRAN = "https://cloud.r-project.org/" ))

# Loading and Preparing Packages

## Installing Required Packages

install.packages ( "readxl" )

install.packages ( "writexl" )

install.packages ( "tidyverse" ) # This includes tidyr and dplyr

install.packages ( "stringr" )

install.packages ( "openxlsx" )

* install.packages( ): Function to install packages in R. Several packages necessary for data manipulation and Excel files are installed here:
  + readxl: Package for reading Excel files.
  + writexl: Package for writing Excel files.
  + tidyverse: Collection of packages for data manipulation and visualization, including tidyr and dplyr.
  + stringr: Package for text string manipulation.
  + openxlsx: Package to read and write Excel files without depending on Java.

## Loading Packages

library (readxl)   
library (writexl)   
library (tidyr)   
library (dplyr)   
library (stringr)   
library (openxlsx)

• library(): Function to load packages already installed in the R session, enabling their functions.

# Specifying and Reading Data

## Specify the File Path

file.choose ()

• file.choose(): Opens a dialog box to select the Excel file manually. (Note: This line is optional if the file path is already known.)

## Define File Path

excel\_path <- "C: \\ Users \\ danie \\ Downloads \\ PERFILES\_TECNOLOGICOS\_EMPRESAS..xlsx"

excel\_path: Variable that stores the path of the Excel file to be read.

## Read the data on each sheet

profiles1 <- read\_excel (excel\_path, sheet = "Storage\_Range\_1" )   
profiles2 <- read\_excel (excel\_path, sheet = "Storage\_Range\_2" )   
profiles3 <- read\_excel (excel\_path, sheet = "Storage\_Range\_3" )

• read\_excel(excel\_path, sheet = “Sheet\_Name”): Reads the data from the specified sheet in the Excel file and stores each sheet in a DataFrame (profiles1, profiles2, profiles3).

# Data Processing

## Check for NA or Inf in the data

check\_valid\_data <- function (df) {  
 return ( ! any ( is.na (df)) && all ( is.finite (df)))   
}

* check\_valid\_data: Function that checks if a DataFrame contains NA (missing) or Inf (infinite) values. Returns TRUE if all data is finite and there are no missing values, otherwise returns FALSE.
  + is.na(df): Checks for the presence of NA values.
  + is.finite(df): Checks for the presence of infinite values.

## Normalize data by row (proportions)

normalize\_data <- function (df) {  
 return ( as.data.frame ( t ( apply (df[, - 1 ], 1 , function (x) x / sum (x, na.rm = TRUE )))))   
}

* normalize\_data: Function that normalizes the data in a DataFrame by row, dividing each value by the sum of the values in that row. The leftmost column (usually identifier) is not normalized.
  + apply(df[, -1], 1, function(x) x / sum(x, na.rm = TRUE)): Applies a function to each row of the DataFrame, excluding the first column.
  + t(): Transposes the resulting matrix to maintain the structure of the DataFrame.
  + as.data.frame(): Converts the transposed matrix back to a DataFrame.

## Data Normalization in Each DataFrame

profiles1\_normalized <- cbind (profiles1[ 1 ], normalize\_data (profiles1))   
profiles2\_normalized <- cbind (profiles2[ 1 ], normalize\_data (profiles2))   
profiles3\_normalized <- cbind (profiles3[ 1 ], normalize\_data (profiles3))

* cbind(profiles1[1], normalize\_data(profiles1)): Combines the first column of the original DataFrame (presumably an ID column) with the normalized data to form a new normalized DataFrame.

# Distance Calculation

## Function to calculate Jaffe distance

jaffe\_distance <- function (p1, p2) {  
 return ( 1 - ( sum (p1 \* p2) / sqrt ( sum ( p1 ^ 2 ) \* sum (p2 ^ 2 ))))   
}

* jaffe\_distance: Function that calculates the technological distance between two profiles using the Jaffe distance. The formula used measures the dissimilarity between two normalized vectors.
  + : Inner product of the two vectors.
  + : Product of the norms (lengths) of the vectors.

## Function to calculate Euclidean distance

euclidean\_distance <- function (p1, p2) {  
 return ( sqrt ( sum ((p1 - p2) ^ 2 )))   
}

* euclidean\_distance: Function that calculates the Euclidean distance between two vectors. It measures the “straight distance” between two points in a multidimensional space.
  + : Sum of the squares of the differences between the corresponding elements of the vectors.

## Function to Calculate the Minimum Complement Distance

min\_complement\_distance <- function (p1, p2) {   
pmin\_sum <- sum ( pmin (p1, p2))  
 return ( 1 - sum\_pmin)   
}

* min\_complement\_distance: Function that calculates the minimum complement distance between two vectors, by measuring the amount of coincidence in the smallest elements of each vector.
  + : Calculates the minimum between the corresponding elements of the two vectors.
  + : Add the minimum values.
  + : Subtract the sum from the minimum of the complement (distance).

## # Calculation and Storage of Distances between Companies

## Function to calculate distances between two specific companies in a sheet

calculate\_specific\_distances <- function (profiles, company1, company2) {   
company\_names <- profiles $ Company   
row1 <- which (company\_names == company1)   
row2 <- which (company\_names == company2)  
   
 if ( length (row1) == 0 || length (row2) == 0 ) {  
 stop ( "One or both companies are not in the data." )   
}  
   
p1 <- as.numeric (profiles[row1, - 1 ])   
p2 <- as.numeric (profiles[row2, - 1 ])  
   
dist\_jaffe <- jaffe\_distance (p1, p2)   
dist\_euclidean <- euclidean\_distance (p1, p2)   
dist\_min\_complement <- min\_complement\_distance (p1, p2)  
   
 return ( list ( jaffe = dist\_jaffe, euclidean = dist\_euclidean, min\_complement = dist\_min\_complement))   
}

* calculate\_specific\_distances: Function that calculates the distances between two specific companies in a DataFrame of normalized profiles.
  + profiles: DataFrame with company profiles, including a column of identifiers and several columns of normalized data.
  + company1 and company2: Names of the two companies to compare.
  + company\_names: Extracts the company names column from the DataFrame.
  + row1 and row2: Find the rows corresponding to the two companies in the DataFrame.
  + as.numeric(profiles[row1, -1]): Extracts and converts to numeric the data from the first company row (excluding the names column).
  + dist\_jaffe (p1, p2), dist\_euclidean (p1, p2), dist\_min\_complement (p1, p2): Calculates the three distances between the profiles of the two companies.
  + return(list(…)): Returns a list with the three calculated distances.

## Calculate Distances Between Two Companies for Each Profile

company1 <- "STANFORD UNIVERSITY"   
company2 <- "THE JOHNS HOPKINS UNIVERSITY"   
  
distances1 <- calculate\_specific\_distances (profiles1\_normalized, company1, company2)   
distances2 <- calculate\_specific\_distances (profiles2\_normalized, company1, company2)   
distances3 <- calculate\_specific\_distances (profiles3\_normalized, company1, company2)

* company1 and company2: Specifies the two companies for which distances will be calculated.
* distances1, distances2, distances3: Stores the results of the distances calculated between companies for each of the normalized profiles (profiles1\_normalized, profiles2\_normalized, profiles3\_normalized).

## Show Results

print ( paste ( "Distances in storage\_range\_1:" ))

## [1] "Distances in storage\_range\_1:"

print (distances1)

## $jaffe   
## [1] 0.0400568   
##   
## $euclidean   
## [1] 0.1590317   
##   
## $minimum\_complement   
## [1] 0.1954254

print ( paste ( "Distances in storage\_range\_2:" ))

## [1] "Distances in storage\_range\_2:"

print (distances2)

## $jaffe   
## [1] 0.05352602   
##   
## $euclidean   
## [1] 0.1771004   
##   
## $minimum\_complement   
## [1] 0.2092214

print ( paste ( "Distances in storage\_range\_3:" ))

## [1] "Storage distances\_range\_3:"

print (distances3)

## $jaffe   
## [1] 0.0513613   
##   
## $euclidean   
## [1] 0.1716264   
##   
## $minimum\_complement   
## [1] 0.205541

• print(): Displays the calculated distances for each profile on the console. Use paste() to create descriptive messages that accompany the results.

## Save the Results in Excel Files

results <- data.frame (  
 Profile = c ( "storage\_range\_1" , "storage\_range\_2" , "storage\_range\_3" ),  
 Jaffe = c (distances1 $ jaffe, distances2 $ jaffe, distances3 $ jaffe),  
 Euclidean = c (distances1 $ euclidean, distances2 $ euclidean, distances3 $ euclidean),  
 Min\_Complement = c (distances1 $ min\_complement, distances2 $ min\_complement, distances3 $ min\_complement)   
)   
  
write\_xlsx ( list ( "Results" = results), "DT\_EMPRESAS.xlsx" )

* **results** : DataFrame that organizes the distance results for each profile into columns, with one row for each profile.
  + **Profile** : Names of the profiles.
  + **Jaffe** , **Euclidean** , **Min\_Complement** : Values of the distances calculated for each profile.
* **write\_xlsx(list("Results" = results), "DT\_139.1.xlsx")** : Saves the DataFrame results to an Excel file named "DT\_139.1.xlsx" at the specified path.