METRICAS\_DISTANCIA\_TECNOLOGICA

Daniel José Carreño Quintero

2024-08-21

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

# Carga y Preparación de Paquetes

## Instalación de Paquetes Necesarios

install.packages("readxl")

install.packages("writexl")

install.packages("tidyverse") # Este incluye tidyr y dplyr

install.packages("stringr")

install.packages("openxlsx")

* install.packages( ): Función para instalar paquetes en R. Aquí se instalan varios paquetes necesarios para la manipulación de datos y archivos Excel:
  + readxl: Paquete para leer archivos Excel.
  + writexl: Paquete para escribir archivos Excel.
  + tidyverse: Colección de paquetes para manipulación y visualización de datos, incluye tidyr y dplyr.
  + stringr: Paquete para la manipulación de cadenas de texto.
  + openxlsx: Paquete para leer y escribir archivos Excel sin depender de Java.

## Carga de Paquetes

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

• library(): Función para cargar paquetes ya instalados en la sesión de R, habilitando sus funciones.

# Especificación y Lectura de Datos

## Especificar la Ruta del Archivo

file.choose()

• file.choose(): Abre un cuadro de diálogo para seleccionar el archivo Excel manualmente. (Nota: Esta línea es opcional si ya se conoce la ruta del archivo).

## Definir la Ruta del Archivo

ruta\_excel <- "C:\\Users\\danie\\Downloads\\PERFILES\_TECNOLOGICOS\_EMPRESAS..xlsx"

ruta\_excel: Variable que almacena la ruta del archivo Excel que se va a leer.

## Leer los datos de cada hoja

perfiles1 <- read\_excel(ruta\_excel, sheet = "Almacenamiento\_Rango\_1")  
perfiles2 <- read\_excel(ruta\_excel, sheet = "Almacenamiento\_Rango\_2")  
perfiles3 <- read\_excel(ruta\_excel, sheet = "Almacenamiento\_Rango\_3")

• read\_excel(ruta\_excel, sheet = “Nombre\_Hoja”): Lee los datos de la hoja especificada en el archivo Excel y almacena cada hoja en un DataFrame (perfiles1, perfiles2, perfiles3).

# Procesamiento de Datos

## Verificar si hay NA o Inf en los datos

verificar\_datos\_validos <- function(df) {  
 return(!any(is.na(df)) && all(is.finite(df)))  
}

* verificar\_datos\_validos: Función que verifica si un DataFrame contiene valores NA (faltantes) o Inf (infinito). Devuelve TRUE si todos los datos son finitos y no hay valores faltantes, de lo contrario, devuelve FALSE.
  + is.na(df): Verifica la presencia de valores NA.
  + is.finite(df): Verifica la presencia de valores infinitos.

## Normalizar los datos por fila (proporciones)

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

* normalizar\_datos: Función que normaliza los datos en un DataFrame por fila, dividiendo cada valor por la suma de los valores en esa fila. La columna de la izquierda (normalmente identificador) no se normaliza.
  + apply(df[, -1], 1, function(x) x / sum(x, na.rm = TRUE)): Aplica una función a cada fila del DataFrame, excluyendo la primera columna.
  + t(): Transpone la matriz resultante para mantener la estructura del DataFrame.
  + as.data.frame(): Convierte la matriz transpuesta de nuevo a un DataFrame.

## Normalización de Datos en Cada DataFrame

perfiles1\_normalizado <- cbind(perfiles1[1], normalizar\_datos(perfiles1))  
perfiles2\_normalizado <- cbind(perfiles2[1], normalizar\_datos(perfiles2))  
perfiles3\_normalizado <- cbind(perfiles3[1], normalizar\_datos(perfiles3))

* cbind(perfiles1[1], normalizar\_datos(perfiles1)): Combina la primera columna del DataFrame original (presumiblemente una columna de identificadores) con los datos normalizados para formar un nuevo DataFrame normalizado.

# Cálculo de Distancias

## Función para Calcular la Distancia de Jaffe

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

* distancia\_jaffe: Función que calcula la distancia tecnológica entre dos perfiles utilizando la distancia de Jaffe. La fórmula utilizada mide la disimilitud entre dos vectores normalizados.
  + : Producto interno de los dos vectores.
  + : Producto de las normas (longitudes) de los vectores.

## Función para Calcular la Distancia Euclidiana

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

* distancia\_euclidiana: Función que calcula la distancia euclidiana entre dos vectores. Mide la “distancia recta” entre dos puntos en un espacio multidimensional.
  + : Suma de los cuadrados de las diferencias entre los elementos correspondientes de los vectores.

## Función para Calcular la Distancia de Mínimo Complemento

distancia\_minimo\_complemento <- function(p1, p2) {  
 suma\_pmin <- sum(pmin(p1, p2))  
 return(1 - suma\_pmin)  
}

* distancia\_minimo\_complemento: Función que calcula la distancia de mínimo complemento entre dos vectores, midiendo la cantidad de coincidencia en los elementos más pequeños de cada vector.
  + : Calcula el mínimo entre los elementos correspondientes de los dos vectores.
  + : Suma los valores mínimos.
  + : Resta la suma del mínimo del complemento (distancia).

## # Cálculo y Almacenamiento de Distancias entre Empresas

## Función para calcular distancias entre dos empresas específicas en una hoja

calcular\_distancias\_especificas <- function(perfiles, empresa1, empresa2) {  
 nombres\_empresas <- perfiles$Empresa  
 fila1 <- which(nombres\_empresas == empresa1)  
 fila2 <- which(nombres\_empresas == empresa2)  
   
 if(length(fila1) == 0 || length(fila2) == 0) {  
 stop("Una o ambas empresas no se encuentran en los datos.")  
 }  
   
 p1 <- as.numeric(perfiles[fila1, -1])  
 p2 <- as.numeric(perfiles[fila2, -1])  
   
 dist\_jaffe <- distancia\_jaffe(p1, p2)  
 dist\_euclidiana <- distancia\_euclidiana(p1, p2)  
 dist\_minimo\_complemento <- distancia\_minimo\_complemento(p1, p2)  
   
 return(list(jaffe = dist\_jaffe, euclidiana = dist\_euclidiana, minimo\_complemento = dist\_minimo\_complemento))  
}

* calcular\_distancias\_especificas: Función que calcula las distancias entre dos empresas específicas en un DataFrame de perfiles normalizados.
  + perfiles: DataFrame con perfiles de empresas, incluyendo una columna de identificadores y varias columnas de datos normalizados.
  + empresa1 y empresa2: Nombres de las dos empresas para comparar.
  + nombres\_empresas: Extrae la columna de nombres de las empresas del DataFrame.
  + fila1 y fila2: Encuentra las filas correspondientes a las dos empresas en el DataFrame.
  + as.numeric(perfiles[fila1, -1]): Extrae y convierte a numérico los datos de la fila de la primera empresa (excluyendo la columna de nombres).
  + distancia\_jaffe(p1, p2), distancia\_euclidiana(p1, p2), distancia\_minimo\_complemento(p1, p2): Calcula las tres distancias entre los perfiles de las dos empresas.
  + return(list(…)): Devuelve una lista con las tres distancias calculadas.

## Calcular Distancias entre las Dos Empresas para Cada Perfil

empresa1 <- "STANFORD UNIVERSITY"  
empresa2 <- "THE JOHNS HOPKINS UNIVERSITY"  
  
distancias1 <- calcular\_distancias\_especificas(perfiles1\_normalizado, empresa1, empresa2)  
distancias2 <- calcular\_distancias\_especificas(perfiles2\_normalizado, empresa1, empresa2)  
distancias3 <- calcular\_distancias\_especificas(perfiles3\_normalizado, empresa1, empresa2)

* empresa1 y empresa2: Especifica las dos empresas para las cuales se calcularán las distancias.
* distancias1, distancias2, distancias3: Almacena los resultados de las distancias calculadas entre las empresas para cada uno de los perfiles normalizados (perfiles1\_normalizado, perfiles2\_normalizado, perfiles3\_normalizado).

## Mostrar los Resultados

print(paste("Distancias en almacenamiento\_rango\_1:"))

## [1] "Distancias en almacenamiento\_rango\_1:"

print(distancias1)

## $jaffe  
## [1] 0.0400568  
##   
## $euclidiana  
## [1] 0.1590317  
##   
## $minimo\_complemento  
## [1] 0.1954254

print(paste("Distancias en almacenamiento\_rango\_2:"))

## [1] "Distancias en almacenamiento\_rango\_2:"

print(distancias2)

## $jaffe  
## [1] 0.05352602  
##   
## $euclidiana  
## [1] 0.1771004  
##   
## $minimo\_complemento  
## [1] 0.2092214

print(paste("Distancias en almacenamiento\_rango\_3:"))

## [1] "Distancias en almacenamiento\_rango\_3:"

print(distancias3)

## $jaffe  
## [1] 0.0513613  
##   
## $euclidiana  
## [1] 0.1716264  
##   
## $minimo\_complemento  
## [1] 0.205541

• print(): Muestra en la consola las distancias calculadas para cada perfil. Utiliza paste() para crear mensajes descriptivos que acompañan los resultados.

## Guardar los Resultados en Archivos Excel

resultados <- data.frame(  
 Perfil = c("almacenamiento\_rango\_1", "almacenamiento\_rango\_2", "almacenamiento\_rango\_3"),  
 Jaffe = c(distancias1$jaffe, distancias2$jaffe, distancias3$jaffe),  
 Euclidiana = c(distancias1$euclidiana, distancias2$euclidiana, distancias3$euclidiana),  
 Minimo\_Complemento = c(distancias1$minimo\_complemento, distancias2$minimo\_complemento, distancias3$minimo\_complemento)  
)  
  
write\_xlsx(list("Resultados" = resultados), "DT\_EMPRESAS.xlsx")

* **resultados**: DataFrame que organiza los resultados de las distancias para cada perfil en columnas, con una fila para cada perfil.
  + **Perfil**: Nombres de los perfiles.
  + **Jaffe**, **Euclidiana**, **Minimo\_Complemento**: Valores de las distancias calculadas para cada perfil.
* **write\_xlsx(list("Resultados" = resultados), "DT\_139.1.xlsx")**: Guarda el DataFrame resultados en un archivo Excel llamado "DT\_139.1.xlsx" en la ruta especificada.