# install.packages("RPostgres")

library(RPostgres)

library(dplyr) # Para manipulación de datos

# Configuración de la conexión a la base de datos

db\_host <- "localhost"

db\_port <- 5432

db\_name <- "metodos\_numericos"

db\_user <- "tu\_usuario\_de\_postgres"

db\_password <- "tu\_contraseña"

# Establecer la conexión

con <- dbConnect(RPostgres::Postgres(),

host = db\_host,

port = db\_port,

dbname = db\_name,

user = db\_user,

password = db\_password)

# Verificar la conexión

if (dbIsValid(con)) {

message("Conexión a la base de datos exitosa!")

} else {

stop("Fallo al conectar a la base de datos.")

}

# Ejemplo de lectura de datos (una vez que estén cargados)

# df\_ventas <- dbReadTable(con, "ventas\_mensuales")

# df\_marcas <- dbReadTable(con, "marcas")

# No olvides desconectar al final de tu script

# dbDisconnect(con)

# Suponiendo que ya tienes tus datos cargados en un dataframe llamado 'df\_ventas\_completo'

# que contenga ventas, marcas, paises, etc.

# Marcas más vendidas (ejemplo usando datos simulados o cargados)

# Si no tienes datos, puedes simular algunos para probar:

# df\_ventas\_completo <- data.frame(

# marca\_nombre = sample(c("Toyota", "Volkswagen", "Ford", "Honda", "Hyundai"), 100, replace = TRUE),

# cantidad\_vendida = sample(100:1000, 100, replace = TRUE),

# anio = sample(2020:2024, 100, replace = TRUE),

# pais\_nombre = sample(c("EE. UU.", "China", "Alemania"), 100, replace = TRUE)

# )

# Marcas más vendidas por año (ejemplo)

marcas\_top <- df\_ventas\_completo %>%

group\_by(marca\_nombre, anio) %>%

summarise(ventas\_totales = sum(cantidad\_vendida, na.rm = TRUE)) %>%

arrange(anio, desc(ventas\_totales))

print(head(marcas\_top))

# Medidas de tendencia central y dispersión para 'cantidad\_vendida'

summary(df\_ventas\_completo$cantidad\_vendida)

mean(df\_ventas\_completo$cantidad\_vendida)

median(df\_ventas\_completo$cantidad\_vendida)

sd(df\_ventas\_completo$cantidad\_vendida)

# Moda (requiere función o paquete)

# install.packages("DescTools")

# library(DescTools)

# Mode(df\_ventas\_completo$cantidad\_vendida)

# Correlación (ejemplo hipotético si tuvieras PIB en el mismo dataframe)

# cor(df\_ventas\_completo$cantidad\_vendida, df\_ventas\_completo$PIB)

# Regresión (ejemplo hipotético)

# modelo\_regresion <- lm(cantidad\_vendida ~ PIB + anio, data = df\_ventas\_completo)

# summary(modelo\_regresion)

# Test de normalidad (si aplica a tus datos de ventas)

# shapiro.test(df\_ventas\_completo$cantidad\_vendida) # Para n < 5000

# Histograma

hist(df\_ventas\_completo$cantidad\_vendida, main = "Histograma de Cantidad Vendida")

# Q-Q plot para normalidad

qqnorm(df\_ventas\_completo$cantidad\_vendida)

qqline(df\_ventas\_completo$cantidad\_vendida, col = "blue")

# Si necesitaras modelar con Poisson (ej. si modelas conteos de ventas por concesionario en un día)

# plot(dpois(0:10, lambda = 3), type = "h", main = "Distribución de Poisson (lambda=3)")

library(ggplot2)

# Ventas totales por marca

ventas\_por\_marca <- df\_ventas\_completo %>%

group\_by(marca\_nombre) %>%

summarise(total\_ventas = sum(cantidad\_vendida, na.rm = TRUE)) %>%

arrange(desc(total\_ventas))

ggplot(ventas\_por\_marca, aes(x = reorder(marca\_nombre, -total\_ventas), y = total\_ventas)) +

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

labs(title = "Ventas Totales por Marca", x = "Marca", y = "Cantidad Vendida") +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))

ggplot(ventas\_por\_marca, aes(x = "", y = total\_ventas, fill = marca\_nombre)) +

geom\_bar(stat = "identity", width = 1) +

coord\_polar("y", start = 0) +

labs(title = "Distribución de Ventas por Marca") +

theme\_void() +

geom\_text(aes(label = scales::percent(total\_ventas / sum(total\_ventas))),

position = position\_stack(vjust = 0.5)) +

guides(fill = guide\_legend(title = "Marca"))

# Necesitas un dataframe con PIB y ventas por país/año

# Ejemplo: df\_pib\_ventas <- data.frame(pib = rnorm(50, 100, 20), ventas = rnorm(50, 50000, 10000))

# ggplot(df\_pib\_ventas, aes(x = pib, y = ventas)) +

# geom\_point() +

# geom\_smooth(method = "lm", col = "red") +

# labs(title = "Ventas vs. PIB", x = "PIB", y = "Cantidad Vendida") +

# theme\_minimal()

# Esto a menudo se muestra en gráficos de barras con barras de error

# Ejemplo:

# df\_resumen\_ventas\_mes\_marca <- df\_ventas\_completo %>%

# group\_by(marca\_nombre, mes) %>%

# summarise(

# media\_ventas = mean(cantidad\_vendida, na.rm = TRUE),

# sd\_ventas = sd(cantidad\_vendida, na.rm = TRUE)

# )

#

# ggplot(df\_resumen\_ventas\_mes\_marca, aes(x = as.factor(mes), y = media\_ventas, fill = marca\_nombre)) +

# geom\_bar(stat = "identity", position = position\_dodge()) +

# geom\_errorbar(aes(ymin = media\_ventas - sd\_ventas, ymax = media\_ventas + sd\_ventas),

# position = position\_dodge(0.9), width = 0.25) +

# labs(title = "Ventas Mensuales Promedio por Marca con Desviación Estándar",

# x = "Mes", y = "Ventas Promedio") +

# theme\_minimal()

# install.packages(c("caret", "randomForest", "forecast", "prophet"))

library(caret)

library(randomForest)

library(forecast) # Para modelos de series de tiempo

# library(prophet) # Si decides usar Prophet

# Ejemplo de preparación de datos (simplificado)

# df\_modelo <- df\_ventas\_completo %>%

# filter(anio <= 2024) %>% # Usar datos hasta 2024 para entrenar

# mutate(

# mes\_factor = as.factor(mes),

# marca\_factor = as.factor(marca\_nombre),

# pais\_factor = as.factor(pais\_nombre)

# # Si tuvieras PIB, lo incluirías aquí

# )

#

# # Crear dummies para factores si no usas un modelo que los maneje automáticamente

# dummy\_vars <- dummyVars(~ mes\_factor + marca\_factor + pais\_factor, data = df\_modelo)

# df\_dummies <- data.frame(predict(dummy\_vars, newdata = df\_modelo))

# df\_modelo\_final <- bind\_cols(df\_modelo, df\_dummies) %>%

# select(-c(mes\_factor, marca\_factor, pais\_factor)) # Eliminar las originales

# Dividir datos (ejemplo)

# set.seed(123)

# index <- createDataPartition(df\_modelo\_final$cantidad\_vendida, p = 0.8, list = FALSE)

# train\_data <- df\_modelo\_final[index, ]

# test\_data <- df\_modelo\_final[-index, ]

# Ejemplo de modelo Random Forest (muy simplificado)

# rf\_model <- randomForest(cantidad\_vendida ~ anio + mes + marca\_factor + pais\_factor,

# data = train\_data,

# ntree = 500)

# print(rf\_model)

#

# # Evaluar el modelo

# predictions <- predict(rf\_model, newdata = test\_data)

# RMSE <- sqrt(mean((predictions - test\_data$cantidad\_vendida)^2))

# print(paste("RMSE del modelo:", RMSE))

Crear el modelo de Marchin Learning

# Ejemplo de creación de datos para predicción (hipotético)

# fechas\_futuras <- expand.grid(

# anio = 2025:2026,

# mes = 1:12,

# marca\_nombre = unique(df\_ventas\_completo$marca\_nombre), # Todas las marcas conocidas

# pais\_nombre = unique(df\_ventas\_completo$pais\_nombre)

# ) %>%

# # Asegúrate de que todas las columnas que el modelo espera estén presentes

# mutate(

# mes\_factor = as.factor(mes),

# marca\_factor = as.factor(marca\_nombre),

# pais\_factor = as.factor(pais\_nombre)

# )

#

# # Si el modelo usa dummies, crea las dummies para fechas\_futuras

# dummy\_vars\_futuras <- dummyVars(~ mes\_factor + marca\_factor + pais\_factor, data = fechas\_futuras)

# df\_dummies\_futuras <- data.frame(predict(dummy\_vars\_futuras, newdata = fechas\_futuras))

# fechas\_futuras\_final <- bind\_cols(fechas\_futuras, df\_dummies\_futuras) %>%

# select(-c(mes\_factor, marca\_factor, pais\_factor))

#

# # Realizar predicciones

# predicciones\_2025\_2026 <- predict(rf\_model, newdata = fechas\_futuras\_final)

#

# # Añadir predicciones al dataframe

# fechas\_futuras\_final$ventas\_predichas <- predicciones\_2025\_2026

#

# # Encontrar las 4 marcas más vendidas para 2025 y 2026

# top\_marcas\_2025\_2026 <- fechas\_futuras\_final %>%

# group\_by(anio, marca\_nombre) %>%

# summarise(ventas\_totales\_predichas = sum(ventas\_predichas, na.rm = TRUE)) %>%

# arrange(anio, desc(ventas\_totales\_predichas)) %>%

# group\_by(anio) %>%

# slice\_head(n = 4) # Obtener las top 4 por año

#

# print(top\_marcas\_2025\_2026)