NOMBRE COMPLETO: Ing. Hilaría Adima Vásquez Durán

NAO ID: 3033

FECHA: 2 de agosto de 2024

NOMBRE DE LA TRAYECTORIA
EN LA QUE ESTÁS ENROLADO: DATA ANALYST CORE

Título del Reto:
Programación R al elaborar reportes estadísticos

SPRINT 1 DESARROLLO R1

Crea un conjunto de programas en R para explorar los datos de Rossmann Pharma a través de tablas con resúmenes estadísticos y visualizaciones que cumplan con las siguientes directrices:

EJERCICIO 3

Diseña un programa("histogram_avg_sales_customer.R") que construya el histograma de frecuencias de la variable avg_sales_customer como se ha definido antes (sales/customers), junto con la el promedio intervalos que define la regla empírica débil para encontrar el 95% de los datos alrededor del promedio, restringiendo el análisis para España y Francia. El resultado de este script deberá ser una figura denominada "histogram_avg_sales_customer.png".

Desarrollo del ejercicio

Carga las bibliotecas necesarias para la manipulación de datos (tidyverse), la lectura de archivos Excel (readxl) y la manipulación de fechas (lubridate).

```{r}
library(tidyverse)
library(readxl)
library(lubridate)
```

#Establecer El Directorio De Trabajo A La Ruta Especificada En `DATA_PATH`

```
DATA_PATH <- "/Users/Usuario/Desktop/Help ME"
setwd(DATA_PATH)
cat("Directorio de trabajo establecido a:", DATA_PATH, "\n")
```

CARGAR LOS DATOS

```
#Leer los datos de ventas SALES_DATA <- 'sales.csv' sales <-
read.csv2(SALES_DATA, sep=";", header=T) cat("Datos de ventas
leídos correctamente. Número de filas:",

# Leer los datos de tiendas STORES_INFO_DATA <-
'stores_info.xlsx' stores <- read_excel(STORES_INFO_DATA,
```

```
sheet = "data") cat("Datos de tiendas leídos correctamente.
Número de filas:"
```

Leer el archivo stores_info.xlsx que contiene información sobre las tiendas, desde la hoja
llamada "data", e imprime un mensaje confirmando que los datos se leyeron correctamente y muestra el número de filas.

```
# Leer los datos de tiendas
STORES_INFO_DATA <- 'stores_info.xlsx'
stores <- read_excel(STORES_INFO_DATA, sheet = "data")
cat("Datos de tiendas leídos correctamente. Número de filas:"
```

Unión de los datos de ventas (sales) con la información de las tiendas (stores) usando la columna store

```
sales_all <- sales %>% left_join(stores, by=c("store")) cat("Tablas de ventas y tiendas unidas correctamente. Número d
```

```
>
> sales_all <- sales %>% left_join(stores, by=c("store"))
> cat("Tablas de ventas y tiendas unidas correctamente. Número de filas:", nro$
Tablas de ventas y tiendas unidas correctamente. Número de filas: 1017209
>
```

#Convierte la columna date a un objeto de fecha y confirma esta acción con un mensaje.

```
sales_all$date <- ymd(sales_all$date)
cat("Columna de fecha convertida a tipo Date.\n")
```

 # Filtrar los datos para incluir solo los registros de España y Francia, y se imprime un mensaje confirmando el número de filas después del filtrado.

```
sales_all <- sales_all %>% filter(country %in% c("spain", "fra
cat("Datos filtrados para España y Francia. Número de filas:"
```

```
> sales_all$date <- ymd(sales_all$date)
> cat("Columna de fecha convertida a tipo Date.\n")
Columna de fecha convertida a tipo Date.
> sales_all <- sales_all %>% filter(country %in% c("spain", "france"))
> cat("Datos filtrados para España y Francia. Número de filas:", nrow(sales_al$
Datos filtrados para España y Francia. Número de filas: 1008915
>
```

Calcula la variable avg_sales_customer (ventas promedio por cliente) solo para los registros donde el número de clientes es mayor que cero, y luego filtra cualquier valor NA resultante. Se imprime un mensaje confirmando el número de filas válidas.

```
#sales_all <- sales_all %>%   mutate(avg_sales_customer =
ifelse(customers > 0, sales / c
filter(!is.na(avg_sales_customer))
cat("Variable avg_sales_customer calculada y filtrada. Número
```

```
> sales_all <- sales_all %>%
+   mutate(avg_sales_customer = ifelse(customers > 0, sales / customers, NA)) $
+   filter(!is.na(avg_sales_customer))
> cat("Variable avg_sales_customer calculada y filtrada. Número de filas válid$
Variable avg_sales_customer calculada y filtrada. Número de filas válidas: 836130
>
```

Calcular y muestra el promedio y la desviación estándar de `avg_sales_customer` , ignorando los valores `NA` .

```
# mean_avg_sales <- mean(sales_all$avg_sales_customer, na.rm =
T std_dev_avg_sales <- sd(sales_all$avg_sales_customer, na.rm
= cat("Promedio de avg_sales_customer:", mean_avg_sales, "\n")
cat("Desviación estándar de avg_sales_customer:", std_dev_avg_
```

```
>
> mean_avg_sales <- mean(sales_all$avg_sales_customer, na.rm = TRUE)
> std_dev_avg_sales <- sd(sales_all$avg_sales_customer, na.rm = TRUE)
> cat("Promedio de avg_sales_customer:", mean_avg_sales, "\n")
Promedio de avg_sales_customer: 9.545958
> cat("Desviación estándar de avg_sales_customer:", std_dev_avg_sales, "\n")
Desviación estándar de avg_sales_customer: 2.142609
>
```

Calcular y mostrar los límites inferior y superior del intervalo del 95% para `avg_sales_customer` .

```
interval_lower <- mean_avg_sales - 2 * std_dev_avg_sales
interval_upper <- mean_avg_sales + 2 * std_dev_avg_sales
cat("Intervalo inferior (95%):", interval_lower, "\n")
cat("Intervalo superior (95%):", interval_upper, "\n")
```

```
> interval_lower <- mean_avg_sales - 2 * std_dev_avg_sales
> interval_upper <- mean_avg_sales + 2 * std_dev_avg_sales
> cat("Intervalo inferior (95%):", interval_lower, "\n")
Intervalo inferior (95%): 5.260741
> cat("Intervalo superior (95%):", interval_upper, "\n")
Intervalo superior (95%): 13.83118
```

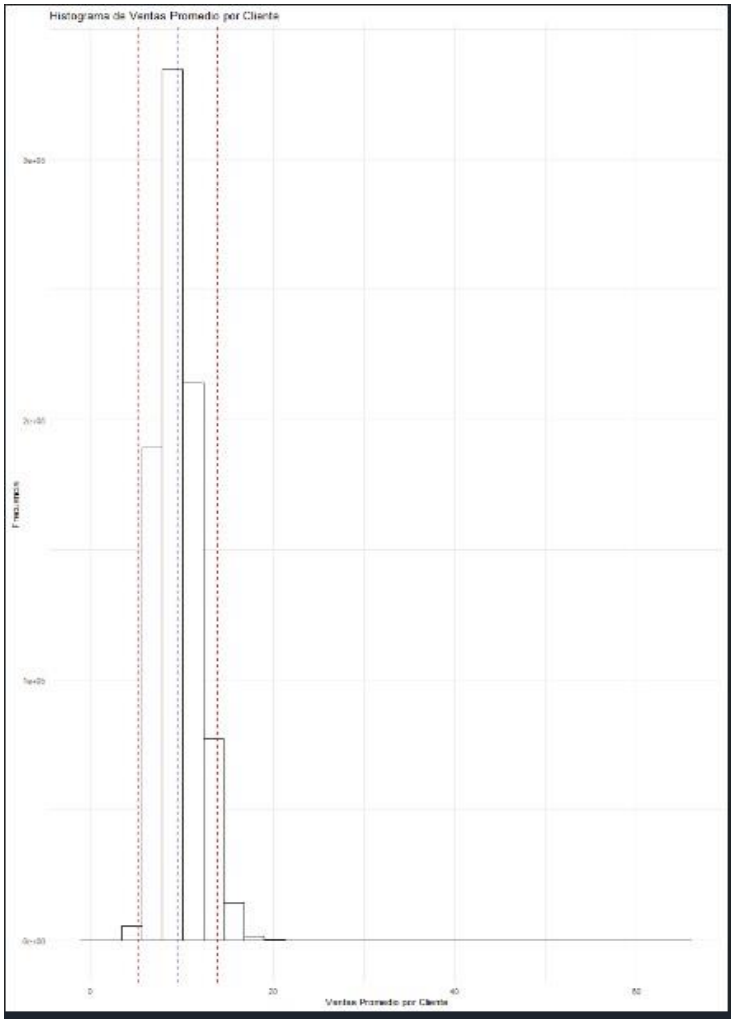
#Crear un histograma de `avg_sales_customer` usando `ggplot2` , añadiendo líneas verticales para el promedio y los límites del intervalo del 95%. Luego, muestra el histograma en la consola y confirma esta acción con un mensaje.

```
histogram_plot <- ggplot(sales_all, aes(x = avg_sales_custome
geom_histogram(color = "black", fill = "white", bins = 30) +
geom_vline(xintercept = mean_avg_sales, linetype = "dashed"
geom_vline(xintercept = interval_lower, linetype = "dashed"
geom_vline(xintercept = interval_upper, linetype = "dashed"
```

```
  labs(title = "Histograma de Ventas Promedio por Cliente",
       x = "Ventas Promedio por Cliente",
       y = "Frecuencia") +
  theme_minimal()

# Mostrar el histograma en la consola
print(histogram_plot)
cat("Histograma creado y mostrado en consola.\n")
```

```
>
> histogram_plot <- ggplot(sales_all, aes(x = avg_sales_customer)) +
+   geom_histogram(color = "black", fill = "white", bins = 30) +
+   geom_vline(xintercept = mean_avg_sales, linetype = "dashed", color = "blue$
+   geom_vline(xintercept = interval_lower, linetype = "dashed", color = "red"$
+   geom_vline(xintercept = interval_upper, linetype = "dashed", color = "red"$
+   labs(title = "Histograma de Ventas Promedio por Cliente",
+        x = "Ventas Promedio por Cliente",
+        y = "Frecuencia") +
+   theme_minimal()
>
> # Mostrar el histograma en la consola
> print(histogram_plot)
> cat("Histograma creado y mostrado en consola.\n")
Histograma creado y mostrado en consola.
>
```

#Histograma como un archivo PNG y se confirma esta acción con un mensaje.

```
ggsave("histogram_avg_sales_customer.png", plot = histogram_pl
cat("La figura 'histogram_avg_sales_customer.png'
```

```
> ggsave("histogram_avg_sales_customer.png", plot = histogram_plot)
Saving 10.1 x 10.1 in image
> cat("La figura 'histogram_avg_sales_customer.png' ha sido guardada exitosame$
La figura 'histogram_avg_sales_customer.png' ha sido guardada exitosamente.
> cat("La figura 'histogram_avg_sales_customer.png' ha sido creada exitosament$
La figura 'histogram_avg_sales_customer.png' ha sido creada exitosamente.
>
```