



Instituto Superior Universitario Tecnológico del Azuay
Tecnología Superior en Big Data

**Proceso de tratamiento de los datos en un conjunto de
datos de ventas de llantas**

Integrantes:

Nube Gutierrez
Eduardo Mendieta

Materia:

Introducción a Big Data

Docente:

MSc. Ing. Carmen Tacuri Vintimilla

Ciclo:

Primer Ciclo

Fecha:

18 de agosto de 2024

Periodo Académico:

Abril 2024 - Agosto 2024

Índice

1.	Introducción	2
2.	Objetivos	3
2.1.	Objetivo general	3
2.2.	Objetivos específicos	3
3.	Paso a paso	4
3.1.	Adquisición y preparación	4
3.2.	Organización y limpieza	5
3.3.	Transformación y visualización	9
3.4.	11
4.	Conclusiones	12
5.	Bibliografía	12

Proceso de tratamiento de los datos en un conjunto de datos de ventas de llantas

1. Introducción

2. Objetivos

2.1. Objetivo general

2.2. Objetivos específicos

-

3. Paso a paso

Para la realización del proceso de tratamiento de datos, hemos elegido un conjunto de datos sobre ventas de llantas proporcionado en un archivo Excel y la herramienta R para su análisis, preparación y visualización. R es un lenguaje de programación y un entorno de software especializado en el análisis estadístico y la visualización de datos. Su amplia gama de paquetes y funciones permite transformar datos mediante técnicas avanzadas de manipulación y modelado, facilitando la limpieza, integración y análisis. Además, R destaca en la visualización de datos, ofreciendo herramientas para crear gráficos y visualizaciones interactivas y personalizables que ayudan a interpretar y comunicar resultados de manera efectiva.

A continuación, se detallan cada uno de los pasos realizados para completar este proceso:

3.1. Adquisición y preparación

- El set de datos es el siguiente:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	IdCliente	NombreCliente	Fecha	Empleado	Referencia	Descripcion	CodigoFamilia	Familia	Cantidad	Ventas	Localidad	Sede	Area
2	C1070956433	PINTO JOHN	9/1/2016	24	LL-2R-0038	PILOT STREET 1	101	LLANTA	1	129310	Puente aranc	Calle 13	35
3	C1032372566	CESAR CRUZ	9/1/2016	14	LL-TCA-0133	TOURING 175/7	101	LLANTA	2	181034	Suba	Suba	32
4	C2865810	PINEDA JAIME	9/1/2016	12	LL-TCA-0019	ENERGY XM2 1	101	LLANTA	1	193095	Usaquen	Santa ana	33
5	C11210285	VILLALBA RODRIGU	9/1/2016	24	LL-TCA-0009	ENERGY XM2 1	101	LLANTA	2	262069	Puente aranc	Calle 13	35
6	C80755314	MOLINA RONALD	9/1/2016	14	LL-TCA-0132	NO UTILIZAR	101	LLANTA	4	362069	Suba	Suba	32
7	C21111935	URQUIJO MERCEDE	9/1/2016	14	LL-TCA-0022	ENERGY XM2 1	101	LLANTA	4	720690	Suba	Suba	32
8	C79295843	GOMEZ FRANCISCO	9/1/2016	12	LL-TCA-0100	PRIMACY 3 ZP 2	101	LLANTA	2	737931	Usaquen	Santa ana	33
9	C79827395	BALLEN CARLOS	9/1/2016	24	LL-PLB-0013	ST250 215/75R1	101	LLANTA	2	931034	Puente aranc	Calle 13	35
10	C860002175	SCHLUMBERGER SU	9/1/2016	7	LL-TCC-0020	ALL TERRAIN T/	101	LLANTA	2	950015	Engativa	Ventas exterr	22
11	C19091837	URIBE GABRIEL	9/1/2016	12	LL-TCA-0114	PRIMACY 3 205	101	LLANTA	4	1167069	Usaquen	Santa ana	33
12	C800035276	TRANSPORTES MON	9/1/2016	2	LL-PLC-0074	X MULTI Z 295/	101	LLANTA	1	1324483	Engativa	Ventas exterr	22
13	C900069712	MONTEJO HEAVY LI	9/1/2016	2	LL-PLC-0074	X MULTI Z 295/	101	LLANTA	1	1324483	Engativa	Ventas exterr	22
14	C900069712	MONTEJO HEAVY LI	9/1/2016	2	LL-PLC-0083	XTE2 215/75R1	101	LLANTA	2	1429173	Engativa	Ventas exterr	22

Figura 1: Archivo Excel con los datos

- Iniciamos creando un script en R en la misma carpeta en la que se encuentra el conjunto de datos. Una vez creado el script, Utilizamos la función `read_excel()` de la biblioteca `readxl` para cargar los datos en el entorno de R.

```
datos ← read_excel('ds-ventallantas-original.xlsx')
View(datos)
```

Figura 2: Cargando los datos en R

	IdCliente	NombreCliente	Fecha	Empleado	Referencia	Descripcion	CodigoFamilia	Familia	Cantidad
1	C107096433	PENITO JOHN	2016-09-01	24	UL-2R-0030	PILOT STREET 150/90-17	101	LLANTA	1,000
2	C1032372566	CEGAR CRUZ	2016-09-01	14	UL-TCA-0133	TOURING 175/70R13	101	LLANTA	2,000
3	C2865810	PINEDA JAIME	2016-09-01	12	UL-TCA-0019	ENERGY XM2 185/65R14	101	LLANTA	1,000
4	C11210285	VILLALBA RODRIGUEZ FABIO	2016-09-01	24	UL-TCA-0009	ENERGY XM2 185/65R13	101	LLANTA	2,000
5	C80795314	MOLINA RONALD	2016-09-01	14	UL-TCA-0132	NO UTILIZAR	101	LLANTA	4,000
6	C21111935	URQUIJO MERCEDES	2016-09-01	14	UL-TCA-0022	ENERGY XM2 185/70R14	101	LLANTA	4,000
7	C79295843	GOMEZ FRANCISCO	2016-09-01	12	UL-TCA-0100	PRIMACY 3 2P 205/55R16	101	LLANTA	2,000
8	C7927395	BALLEN CARLOS	2016-09-01	24	UL-PLB-0013	ST250 215/75R17,5 NO UTILIZAR REEM LL-PLB-0040	101	LLANTA	2,000
9	C860002175	SCHUMBERGER SURENO S A	2016-09-01	7	UL-TCC-0020	ALL TERRAIN T/A K02 265/70R16	101	LLANTA	2,000
10	C19091837	URIBE GABRIEL	2016-09-01	12	UL-TCA-0114	PRIMACY 3 205/60R16	101	LLANTA	4,000
11	C800035276	TRANSPORTES MONTEJO SAS	2016-09-01	2	UL-PLC-0074	X MULTI 2 295/60R22,5	101	LLANTA	1,000
12	C900069712	MONTEJO HEAVY LIFT SA	2016-09-01	2	UL-PLC-0074	X MULTI 2 295/60R22,5	101	LLANTA	1,000
13	C900069712	MONTEJO HEAVY LIFT SA	2016-09-01	2	UL-PLC-0083	XTE2 215/75R17,5	101	LLANTA	2,000
14	C900069712	MONTEJO HEAVY LIFT SA	2016-09-01	2	UL-PLC-0084	XTE2 235/75R17,5	101	LLANTA	2,000

Figura 3: Visualización de los datos en R

3.2. Organización y limpieza

- Usando la función `nrow()` podemos observar el conjunto de datos tiene 127266 filas. Aplicamos un filtro para observar si existen filas que esten completamente en blanco y como resultado existen 11 filas. Para eliminar dichas filas creamos una nueva variable que almacene los datos limpios y aplicando nuevamente el filtro extraemos unicamente la filas que contienen datos, quedando un total de 127255 filas.

```
num_filas <- nrow(datos)
print(paste('El dataset a limpiar tiene:', num_filas, 'filas.'))
# El dataset a limpiar tiene: 127266 filas.

filas_blanco <- sum(apply(datos, 1, function(x) all(is.na(x) | x == '')))
print(paste('El dataset tiene', filas_blanco, 'filas completamente en blanco.'))
# El dataset tiene 11 filas completamente en blanco.

datos_limpios <- datos[!apply(datos, 1, function(x) all(is.na(x) | x == '')), ]
num_filas <- nrow(datos_limpios)
print(paste('El dataset ahora tiene', num_filas, 'filas después de eliminar las filas en blanco.))
# El dataset ahora tiene 127255 filas después de eliminar las filas en blanco.
```

Figura 4: Eliminando filas vacias

- Ordenamos las filas utilizando alguna columna con la función `arrange()` de la biblioteca `dplyr`, en este caso `Empleado`, con el objetivo de poder visualizar si existen filas duplicadas. Aplicamos un filtro para identificar y eliminar filas repetidas. En el conjunto de datos existen 24923 filas repetidas, luego de este proceso, tendremos un total de 102332 filas.

	IdCliente	NombreCliente	Fecha	Empleado	Referencia	Descripcion
1	C822004453	INGENIEROS CONTRATISTAS DE COLOMBIA SAS	2017-06-10	1	REC-0071	REENC XDY3 12R22,5
2	C822004453	INGENIEROS CONTRATISTAS DE COLOMBIA SAS	2017-06-10	1	REC-0071	REENC XDY3 12R22,5
3	C860040576	COLTANQUES SAS	2017-06-16	1	SE-GF-1-0001	ALINEACION
4	C860040576	COLTANQUES SAS	2017-06-16	1	SE-GF-1-0001	ALINEACION
5	C860040576	COLTANQUES SAS	2017-06-16	1	SE-GF-1-0001	ALINEACION
6	C860040576	COLTANQUES SAS	2017-06-16	1	SE-GF-1-0001	ALINEACION
7	C860040576	COLTANQUES SAS	2017-06-16	1	SE-GF-1-0001	ALINEACION
8	C860040576	COLTANQUES SAS	2017-06-16	1	SE-GF-1-0001	ALINEACION
9	C860040576	COLTANQUES SAS	2017-06-16	1	SE-GF-1-0001	ALINEACION
10	C860040576	COLTANQUES SAS	2017-06-16	1	SE-GF-1-0001	ALINEACION
11	C860040576	COLTANQUES SAS	2017-06-16	1	SE-GF-1-0001	ALINEACION
12	C860040576	COLTANQUES SAS	2017-06-30	1	SE-GF-1-0001	ALINEACION

Figura 5: Visualización de datos repetidos

```

datos_repetidos ← datos_limpios[duplicated(datos_limpios) | duplicated(datos_limpios, fromLast = TRUE), ]
print(paste('Datos repetidos:', nrow(datos_repetidos)))
# Datos repetidos: 24923.

View(datos_repetidos)

datos_limpios ← datos_limpios[!duplicated(datos_limpios) & !duplicated(datos_limpios, fromLast = TRUE), ]
print(paste('El dataset ahora tiene:', nrow(datos_limpios), ', luego de eliminar las filas repetidas.'))
# El dataset ahora tiene: 102332 , luego de eliminar las filas repetidas.

```

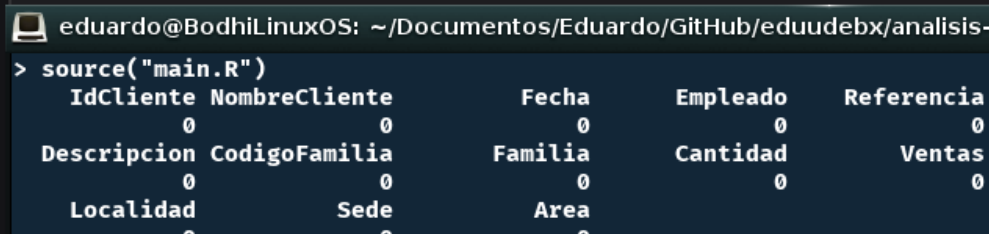
Figura 6: Eliminando datos repetidos

- Buscamos el número de valores faltantes por columna utilizando la función `colSums()`, para aplicar cualquier operación de limpieza en el caso de encontrar datos faltantes. Podemos observar que el número de datos faltantes es **cero** en todas las columnas.

```

print(colSums(is.na(datos_limpios)))

```



IdCliente	NombreCliente	Fecha	Empleado	Referencia
0	0	0	0	0
Descripcion	CodigoFamilia	Familia	Cantidad	Ventas
0	0	0	0	0
Localidad	Sede	Area		
0	0	0		

Figura 7: Buscando datos faltantes por columna

- Validamos que las columnas `Empleado`, `CodigoFamilia`, `Ventas`, `Area` y `Cantidad`, tengan números enteros mayores a cero. Luego de la validación la única que no cumplía con esta restricción es `Cantidad` puesto que tiene únicamente 2 celdas con valores decimales '1.068', '1.006' estos valores fueron redondeados al inmediato superior.

```

datos_limpios$Empleado ← as.numeric(datos_limpios$Empleado)
print(sum(is.na(datos_limpios$Empleado) | datos_limpios$Empleado ≤ 0 | datos_limpios$Empleado ≠ floor(datos_limpios$Empleado)))
# Respuesta: 0

datos_limpios$CodigoFamilia ← as.numeric(datos_limpios$CodigoFamilia)
print(sum(is.na(datos_limpios$CodigoFamilia) | datos_limpios$CodigoFamilia ≤ 0 | datos_limpios$CodigoFamilia ≠ floor(datos_limpios$CodigoFamilia)))
# Respuesta: 0

datos_limpios$Ventas ← as.numeric(datos_limpios$Ventas)
print(sum(is.na(datos_limpios$Ventas) | datos_limpios$Ventas ≤ 0 | datos_limpios$Ventas ≠ floor(datos_limpios$Ventas)))
# Respuesta: 0

datos_limpios$Area ← as.numeric(datos_limpios$Area)
print(sum(is.na(datos_limpios$Area) | datos_limpios$Area ≤ 0 | datos_limpios$Area ≠ floor(datos_limpios$Area)))
# Respuesta: 0

```

Figura 8: Validando columnas con valores enteros

```

datos_limpios$Cantidad ← as.numeric(datos_limpios$Cantidad)
print(sum(is.na(datos_limpios$Cantidad))) # Respuesta: 0
print(sum(datos_limpios$Cantidad ≤ 0)) # Respuesta: 0
print(sum(datos_limpios$Cantidad ≠ floor(datos_limpios$Cantidad))) # Respuesta: 2

valores_no_enteros ← datos_limpios$Cantidad[datos_limpios$Cantidad ≠ floor(datos_limpios$Cantidad)]
print(valores_no_enteros) # 1.068 1.006

datos_limpios$Cantidad ← ifelse(datos_limpios$Cantidad ≠ floor(datos_limpios$Cantidad),
                                ceiling(datos_limpios$Cantidad),
                                datos_limpios$Cantidad)

```

Figura 9: Corrigiendo datos de la columna Cantidad

- En la columna Fecha, validamos que todos los campos tengan el formato mes/dia/año. Al realizar dicha validación, todos los campos cumplen con la condición. Procedemos a cambiar el formato de la fecha a dia/mes/año.

```

datos_limpios$Fecha ← as.character(datos_limpios$Fecha)
print(sum(!is.na(datos_limpios$Fecha) & !is.na(as.Date(datos_limpios$Fecha, format = "%m/%d/%Y"))))
#Respuesta: 0

datos_limpios$Fecha ← format(as.Date(datos_limpios$Fecha, format = "%m/%d/%Y"), "%d/%m/%Y")
View(datos_limpios)

```

Figura 10: Validando columna de Fecha

Fecha	Empleado	Referencia
07/09/2016	1	LL-PLB-0031
07/09/2016	1	LL-TCC-0095
07/09/2016	1	SE-CS-12-0004
07/09/2016	1	SE-CS-2-0002

Figura 11: Formato de fecha dia/mes/año

- En el caso de las columnas NombreCliente, Descripcion, Familia, Localidad y Sede cambiamos su formato de texto a capitalize de tal modo que todas tengan el mismo formato. Esto utilizando la biblioteca stringr.


```

datos_limpios <- datos_limpios %>%
  mutate(
    NombreCliente = str_to_title(NombreCliente),
    Descripcion = str_to_title(Descripcion),
    Familia = str_to_title(Familia),
    Localidad = str_to_title(Localidad),
    Sede = str_to_title(Sede)
  )

View(datos_limpios)

```

Figura 12: Dando un unico formato a las columnas tipo String

	IdCliente	NombreCliente	Fecha	Empleado	Referencia	Descripcion
1	C79647705	Ramirez Oscar	07/09/2016	1	LL-PLB-0031	Xze2 215/75 R17.5
2	C900343104	Colombia Importa Cargo Ltda	07/09/2016	1	LL-TCC-0095	Ltx M/S 245/65r17
3	C900343104	Colombia Importa Cargo Ltda	07/09/2016	1	SE-CS-12-0004	Ajuste De Frenos
4	C900343104	Colombia Importa Cargo Ltda	07/09/2016	1	SE-CS-2-0002	Balanceo Camioneta
5	C900343104	Colombia Importa Cargo Ltda	07/09/2016	1	SE-CS-1-0002	Alineacion Camioneta
6	C900343104	Colombia Importa Cargo Ltda	07/09/2016	1	VFA-0091	Filtro De Aceite (21 A 30)
7	C900343104	Colombia Importa Cargo Ltda	07/09/2016	1	LUA-0020	Mobil 5w30 X 1/4
8	C900364615	Organizacion Suma Sas	12/09/2016	1	SE-GF-7-0002	Servicios Realizados Operacion Suma
9	C900364615	Organizacion Suma Sas	12/09/2016	1	SE-GF-7-0002	Servicios Realizados Operacion Suma

Figura 13: Visualización del resultado final

- Una vez culminado el proceso de limpieza, de un total de 127266 registros del set de datos original, el resultado es uno con 102332. En total se han eliminado un total de 24934 registros.

write.xlsx(datos_limpios, 'datos-limpios.xlsx')

	A	B	C	D	E	F	G	H	I	J	K	L	M
	IdCliente	NombreClien	Fecha	Empleado	Referencia	Descripcion	CodigoFamili	Familia	Cantidad	Ventas	Localidad	Sede	Area
1	C79647705	Ramirez Osca	07/09/2016		1 LL-PLB-0031	Xze2 215/75	101	Llanta		2	992759 Engativa	Calle 80	31
2	C900343104	Colombia Im	07/09/2016		1 LL-TCC-0095	Ltx M/S 245/	101	Llanta		4	1755172 Engativa	Calle 80	31
3	C900343104	Colombia Im	07/09/2016		1 SE-CS-12-000	Ajuste De Fre	106	Servicios		1	21552 Engativa	Calle 80	31
4	C900343104	Colombia Im	07/09/2016		1 SE-CS-2-0002	Balanceo Car	106	Servicios		4	31034 Engativa	Calle 80	31
5	C900343104	Colombia Im	07/09/2016		1 SE-CS-1-0002	Alineacion Ca	106	Servicios		1	32759 Engativa	Calle 80	31
6	C900343104	Colombia Im	07/09/2016		1 VFA-0091	Filtro De Acei	109	Filtros		1	32759 Engativa	Calle 80	31
7	C900343104	Colombia Im	07/09/2016		1 LUA-0020	Mobil 5w30 X	103	Lubricantes		1	38500 Engativa	Calle 80	31
8	C900343104	Colombia Im	07/09/2016		1 SE-GF-7-0002	Servicios Real	106	Servicios		1	539279 Bogota	Grandes Flot	21
9	C900364615	Organizacion	12/09/2016		1 SE-GF-7-0002	Servicios Real	106	Servicios		1	602455 Bogota	Grandes Flot	21
10	C900364615	Organizacion	12/09/2016		1 SE-GF-7-0002	Servicios Real	106	Servicios		1	1737572 Bogota	Grandes Flot	21
11	C900364615	Organizacion	12/09/2016		1 SE-GF-7-0002	Servicios Real	106	Servicios		1	1737572 Bogota	Grandes Flot	21

Figura 14: Exportación de set de datos limpio

3.3. Transformación y visualización

- Previo a la depuración de los datos, se generó un Modelo Relacional con el fin de darle estructura a la información y tener una idea de cómo deberíamos tratarla.

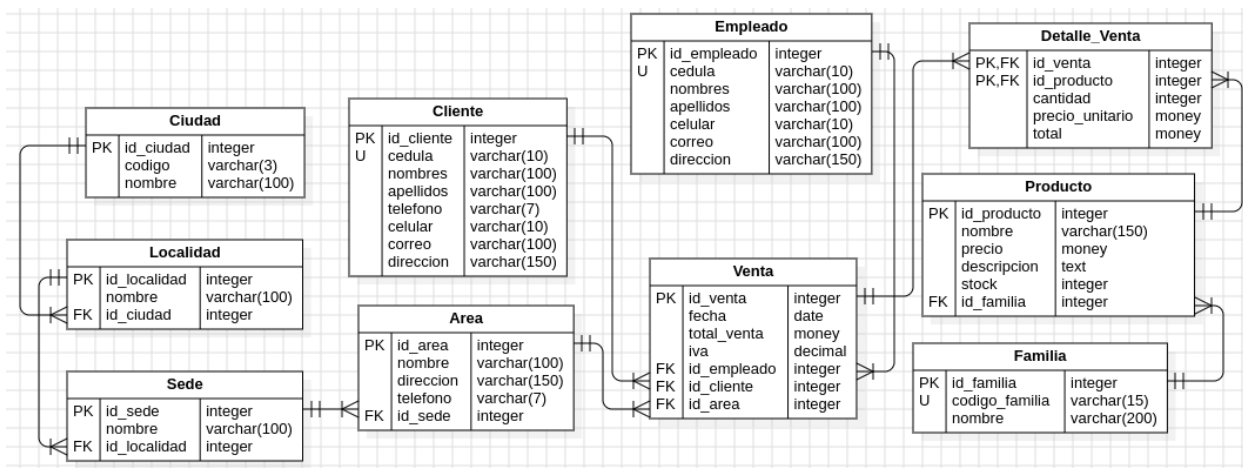


Figura 15: Modelo Físico - Relacional

- La primera propuesta para la visualización de los datos en un gráfico de barras en el cual se analiza la cantidad de ventas realizadas por cada uno de los empleados en los distintos años. Para esto se creó una columna llamada AnioVenta y NombreEmpleado derivadas de Fecha y Empleado respectivamente.

```

datos_limpios$Fecha <- as.Date(datos_limpios$Fecha, format = "%d/%m/%Y")
datos_limpios$AnioVenta <- format(datos_limpios$Fecha, "%Y")
datos_limpios$NombreEmpleado <- paste("Emp", datos_limpios$Empleado)

datos_limpios$AnioVenta <- as.numeric(as.character(datos_limpios$AnioVenta))

par(mfrow = c(2, 3))

anios <- unique(datos_limpios$AnioVenta)

for(anio in anios){
  datos_filtrados <- datos_limpios[datos_limpios$AnioVenta == anio, ]
  nombres_unicos <- sort(unique(datos_filtrados$NombreEmpleado))

  ventas_empleados <- vector('numeric')
  for (nombre in nombres_unicos) {
    filtro <- datos_limpios[datos_limpios$NombreEmpleado == nombre, ]
    ventas_empleados <- c(ventas_empleados, length(filtro$Ventas))
  }

  barplot(height = ventas_empleados, names = nombres_unicos, horiz = 1, las = 1,
          main = paste('Total de ventas por empleado en el año', anio))
  grid(col = '#2B434F')
}
  
```

Figura 16: Filtro de datos para crear el grafico de barras

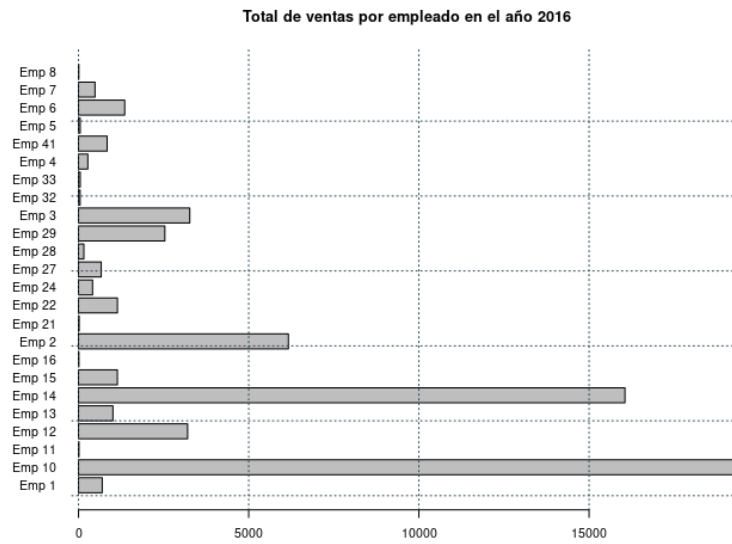


Figura 17: Gráfico de barras 1

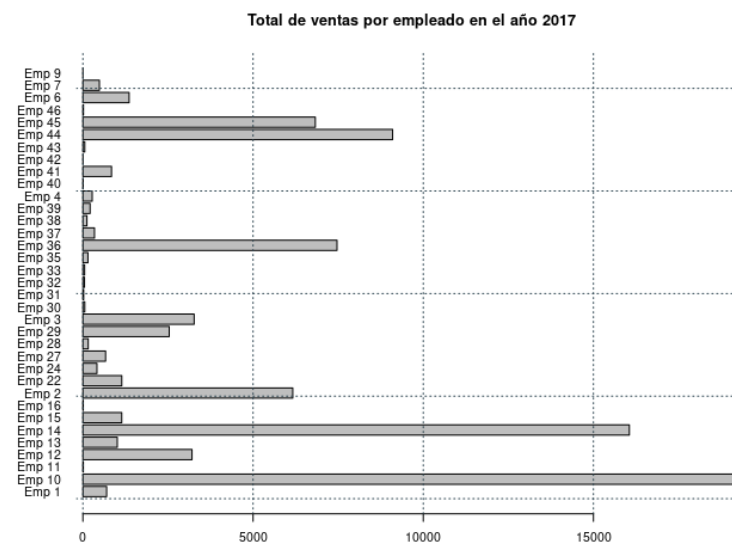


Figura 18: Gráfico de barras 2

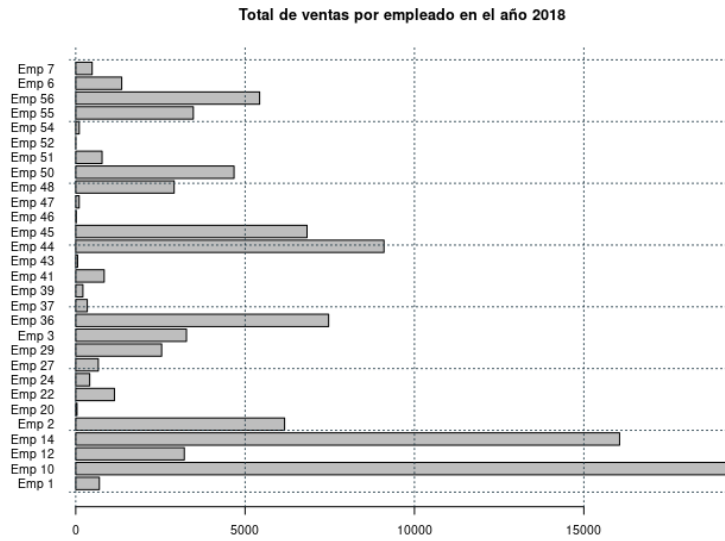


Figura 19: Gráfico de barras 3

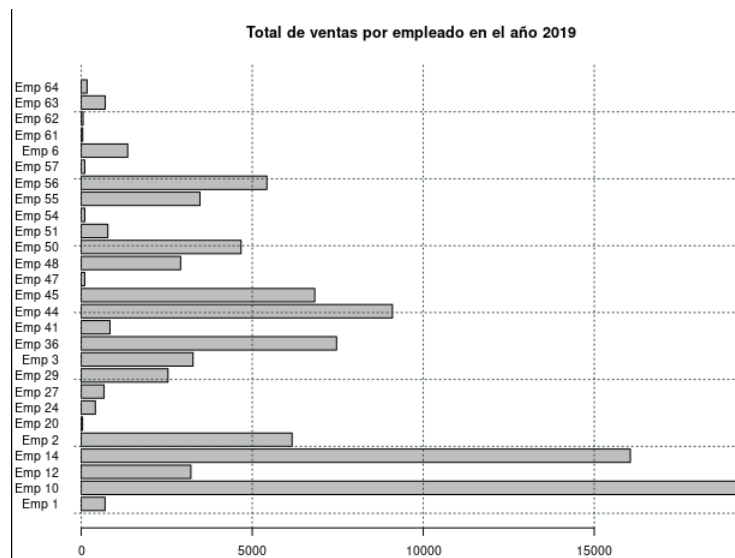


Figura 20: Gráfico de barras 4

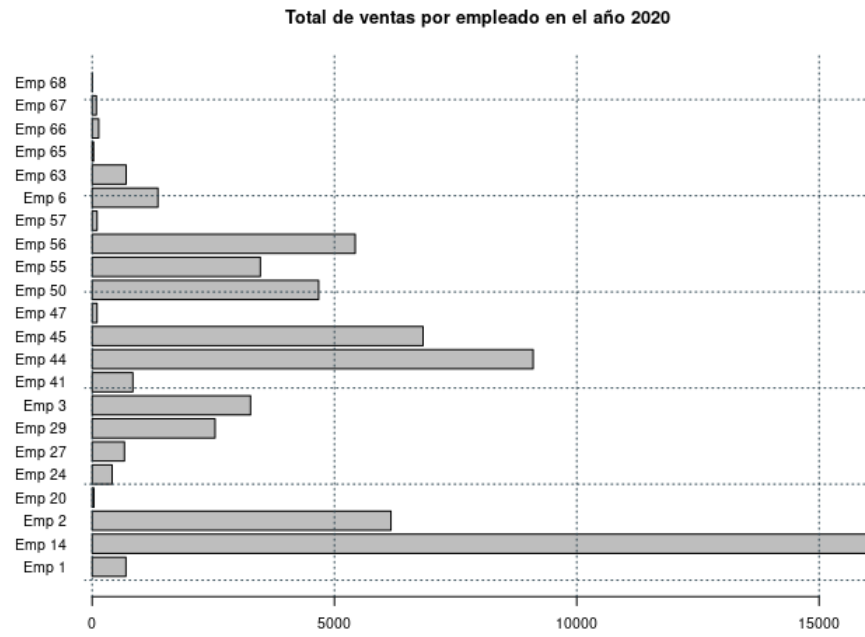


Figura 21: Gráfico de barras 5

■

3.4.

4. Conclusiones

- 1.

5. Bibliografía

-