



GFPI-F-135 VALIDA LA DATA DE ACUERDO CON EL PROCESO LIMPIEZA DE DATOS CON R

ACTIVIDADES POR DESARROLLAR:

1. Datos faltantes
2. Columnas irrelevantes (que no responden al problema a solucionar)
3. Filas repetidas
4. Valores extremos o atípicos (OUTLIERS)
5. Errores tipográficos.
6. Imputación de datos perdidos
7. Utilizar SQL desde R
8. Conectar a Base de datos desde R



1. Situaciones para realizar [limpieza en los datos](#):

• **Remover duplicados o datos irrelevantes:**

Tener datos duplicados sucede en la etapa de recolección de datos. Al tener diversas fuentes, se busca juntar los datos, lo que puede resultar en tener duplicados y se deben descartar filas repetidas. Los datos irrelevantes no influyen o no impactan al problema que se está intentando solucionar.

- **Corregir errores estructurales:** Cuando observan nomenclaturas extrañas, errores tipográficos o gramaticales.
- **Corregir outliers:** o valor atípico es aquel que se está por encima o por debajo del rango normal de valores de la variable que se está estudiando.
- **Imputar:** es el proceso estadístico de sustituir valores faltantes o incorrectos con valores nuevos mediante técnicas de estimación.
 - Eliminación de datos
 - Imputación sin cálculos
 - Imputación básica
 - Usando la librería SKLearn
 - Usando distribuciones
 - Usando interpolación

• Manejo de datos faltantes:

Son aquellos vacíos de datos en la información recolectada. Existen opciones para tratar los datos faltantes:



○ **Eliminar todos los registros que contengan datos faltantes en algún campo:**

Siempre y cuando los registros a eliminar sean mínimos. Si son un porcentaje importante de los datos, más del 10%, es mejor considerar otra alternativa.

○ **Reemplazar los datos faltantes por valores basados en otras**


observaciones: Existen varias técnicas para esto (utilizar knn o algoritmo de vecinos cercanos, predecir los datos faltantes, etc.), lo más común es reemplazar por el promedio, o por la mediana (en el caso que el promedio esté sesgado por algunos valores dentro de los datos).

- Prueba de las correlaciones dicotomizadas: donde se construye una variable dicotomizada asignando cero a los variables ausente en las variables que contengan valores ausentes y el valor de 1 a las presentes; dentro las [variables dicotómicas](#) tenemos entre otras:
 - Género: Masculino o Femenino
 - Coin Flip: cara o cruz
 - Tipo de propiedad: residencial o comercial
 - Estado de atleta: profesional o aficionado
 - Resultados del examen: aprobado o reprobado
- Si al realizar la correlación y su resultado es no significativa, podemos decir que son aleatorios y se podrá imputar de forma directa o su



eliminación siempre y cuando no contengan valores significativos en las demás variables que afecten el objeto de estudio. Se recomienda la lectura de datos perdido en este [enlace](#) en especial el ejemplo de la figura 10-6 de la pagina 339.

- Tenemos algunas imputaciones directas que son el proceso de la estimación de valores ausentes en valores validos en la muestra, tales como:
 - Imputación por las características de la distribución (desviación típica y las correlaciones)
- Imputación por valores estimados sobre la base de otra información existente en la muestra, por ejemplo, falta el tipo de documento de identidad podemos relacionarla con su fecha de nacimiento.
- También podemos imputar por:
 - El método de sustitución por la media o mediana
 - El método por un valor constante
 - El método por interpolación lineal
 - El método de imputación por regresión: se utiliza para predecir los valores ausentes en relación con otras variables, este método tiene como desventaja que subestima la varianza de la distribución suponiendo que la variables con datos ausentes tienen correlaciones estanciales con otras variables.
 - El método de imputación múltiple: es la combinación de varios métodos anteriormente mencionados.

 Se recomienda realizar la lectura de [Guía Quartz de limpieza de datos](#).

2. Comandos utilizados para limpieza de datos

Utilizar el dataset [COVID19 Completo de MINSALUD](#) o [formato JSON](#)



El paquete dplyr:

- select: retorna un subconjunto de columnas de un data frame.

```
library(dplyr )  
Genero<-COVID19 %>% select(Sexo,Edad)
```

- filter: extrae un subconjunto de filas de un marco de datos según las condiciones lógicas

```
library(dplyr )  
filter(COVID19, rownames(COVID19) == 'Sexo')
```

- arrange: reordenar filas de un marco de datos

```
library(dplyr )  
DM_PAIS %>% arrange(DM_PAIS$NOMBRE)
```

- rename: renombrar variables en un marco de datos

```
library(dplyr )  
COVID19<-COVID19 %>% rename("SEXO"="Sexo")
```

- mutate: agregar nuevas variables / columnas o transformar variables existentes.

```
library(dplyr )  
COVID19<-COVID19 %>% mutate("IDMUNI"=paste0(  
  substr(COVID19$Código.DIVIPOLA,1,2), '- ', substr(COVID19$Código.DIVI-  
POLA,3,6))  
)
```

- summarise: generar estadísticas de resumen de diferentes variables en los

Datos

```
library(dplyr )  
COVID19 %>%summarise(mediana=median(Edad),varianza=var(Edad))
```



- Crea una columna

```
COVID19$'casos de estudio'<-0
```

- Categoriza con un condicional

```
COVID19$'IDSEXO'<-ifelse(COVID19$Sexo=='F',1,2)
```

- Crea un dataframe con un condicional compuesto

```
MujeresFallecidas<-COVID19[COVID19$Sexo=='F' & COVID19$Estado=='Fallecido',]
```

- Colocar un índice

```
x=length(DM_ESTADO$Estado)
DM_ESTADO$Estado<-c(1:x)
```

- Obtener número de filas de un dataframe

```
x=nrow(ATENCION)
x=length(ATENCION)
```

- Obtener número de columnas de un dataframe

```
z=ncol(ATENCION)
```

- Elimina duplicados

```
DM_CIUADAD <- COVID19[c(3,4)]
```

- Mezclar un dataframe con otro dataframe (JOIN)

```
COVID19<-merge(COVID19,ATENCION,by.x="atención",by.y="NOMBRE",all.x=TRUE)
```

- Renombrar una columna

```
colnames(DM_ESTADO)[1]<- 'IDES-  
TADO'
```

- Borrar una columna

```
COVID19<-COVID19[, -c(1) ]
```

- Borrar una fila (de acuerdo al número)

```
COVID19<-COVID19[-c(1), ]
```

- Convierte una columna de tipo cadena a tipo numérico

```
COVID19$IDSEXO<-as.integer(CO-  
VID19$IDSEXO)
```



- Reemplaza valores vacíos

```
COVID19$PAIS[is.na(COVID19$PAIS)]<- 'COLOMBIA'
```

- Crea tabla de frecuencias

```
table(COVID19$Sexo)
```

- Categorizar una variable

```
COVID19$CatEdad<-cut(as.integer(COVID19$Edad), breaks = c(-1,12,18,62,120),  
  labels = c("Niño","Joven","Adulto","Adulto Mayor"))
```

- Día de la semana

```
date <- data.frame(date = as.Date(c("2020-10-11","2000-10-01",  
  "1999-12-08","2021-05-05")))
weekday <- weekdays(date$date)
print(weekday)
```

- Obtener una parte de una cadena de caracteres.

```
df<-("Centro de Gestión de Mercados, Logística y Tecnologías de la Informa-  
ción")
a<-paste(substring(df,1,1),substring(df,11,11),substring(df,22,22),subs-  
tring(df,32,32),substring(df,44,44),substring(df,63,63))
print(a)
#SALIDA
[1] "C G M L T I"
```

3. Comandos utilizados para limpieza de datos en Python

- Renombrar una columna

```
data.rename(columns={'ID de caso':'ID'},inplace=True)
```

- Borrar una columna

```
data.drop('FACTUAL',axis=1,inplace=True)
```

- Borrar una fila (de acuerdo al índice)

```
data = data.drop(5)
```

- Convierte una columna de tipo cadena a tipo numérico

```
COVID19$IDSEXO<-as.integer(COVID19$IDSEXO)
```



- Crea una columna

```
COVID19['AUX']=0
```

- Categoriza con un condicional

```
import pandas as pd  
import numpy as np
```

```
COVID19['AUX'] = np.where((COVID19['Sexo'] == 'F'), 1, 2)
```

- Crea un dataframe con un condicional compuesto

```
MujeresFallecidas = COVID19[(COVID19['Sexo'] == 'F') & (COVID19['Estado'] == 'Fallecido')]
```

- Colocar un índice

```
x=length(DM_ESTADO$Estado)  
DM_ESTADO$Estado<-c(1:x)
```

- Obtener número de filas de un dataframe

```
x= x=len(MujeresFallecidas)
```

- Obtener número de columnas de un dataframe

```
y=MujeresFallecidas.columns  
print(len(y))
```

- Elimina duplicados

```
COVID19.drop_duplicates(inplace=True)
```

- Mezclar un dataframe con otro dataframe (JOIN)

```
COVID19<-merge(COVID19,ATENCION,left_on="atención",right_on="NOMBRE",  
how="left")
```

- Remplaza valores vacíos

```
MujeresFallecidas['PAIS'].fillna('COLOMBIA', inplace = True)  
print(MujeresFallecidas['PAIS'].value_counts())
```

- Crea tabla de frecuencias

```
print(COVID19.Sexo.value_counts ())
```

- Categorizar una variable

```
rangos=[-1,5,10,18,50,100,110]  
nombrer=['A','B','C','D','E','F']  
COVID19['GEDAD']=pd.cut(COVID19['Edad'],rangos,labels=nombrer)
```

- Día de la semana



```
date <- data.frame(date = as.Date(c("2020-10-11", "2000-10-01",  
                                   "1999-12-08", "2021-05-05")))
weekday <- weekdays(date$date)
print(weekday)
```

- **Obtener una parte de una cadena de caracteres.**

```
df<-("Centro de Gestión de Mercados, Logística y Tecnologías de la Informa-  
ción")
a<-paste(substring(df,1,1),substring(df,11,11),substring(df,22,22),subs-  
tring(df,32,32),substring(df,44,44),substring(df,63,63))
print(a)
#SALIDA
[1] "C G M L T I"
```




4. Datos erróneos e irrelevantes.



La reducción de la dimensionalidad produce una representación más compacta y más fácilmente interpretable del concepto de objetivo, centrando la atención del usuario en las variables más relevantes.

- Utilizar las siguientes librerías

```
library(dplyr)
```

- Configurar la carpeta de trabajo:

```
setwd("c:/Borrar")
```

- Descargar el DATASET, descomprimir, colocar en la variable ruta el lugar donde se descargó el archivo, no olvide de no utilizar “\” sino “/” para construir la ruta.
- Cargar el archivo CSV

```
COVID19<-read.csv("https://siomi.datasena.com/analitica/data/COVID19.csv",  
sep=";", header = TRUE, encoding = "UTF-8")
```

- Medidas de tendencias central

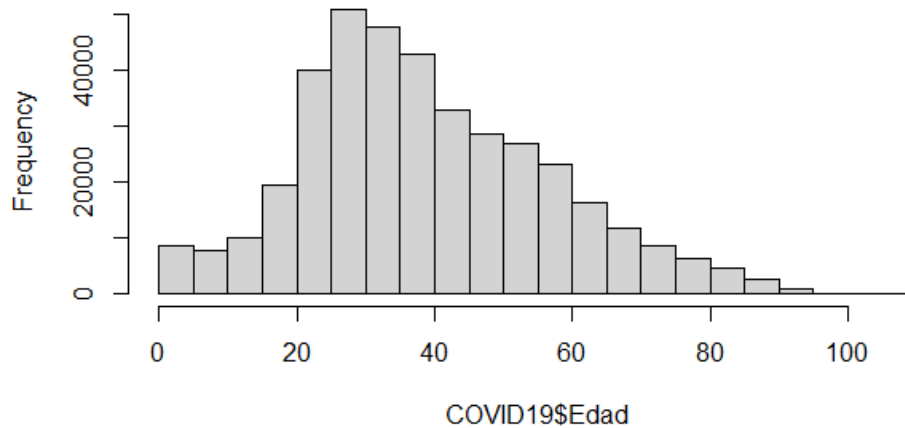
```
summary(COVID19$Edad)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
0.00	27.00	37.00	39.39	51.00	107.00	1

```
hist(COVID19$Edad)
```



Histogram of COVID19\$Edad

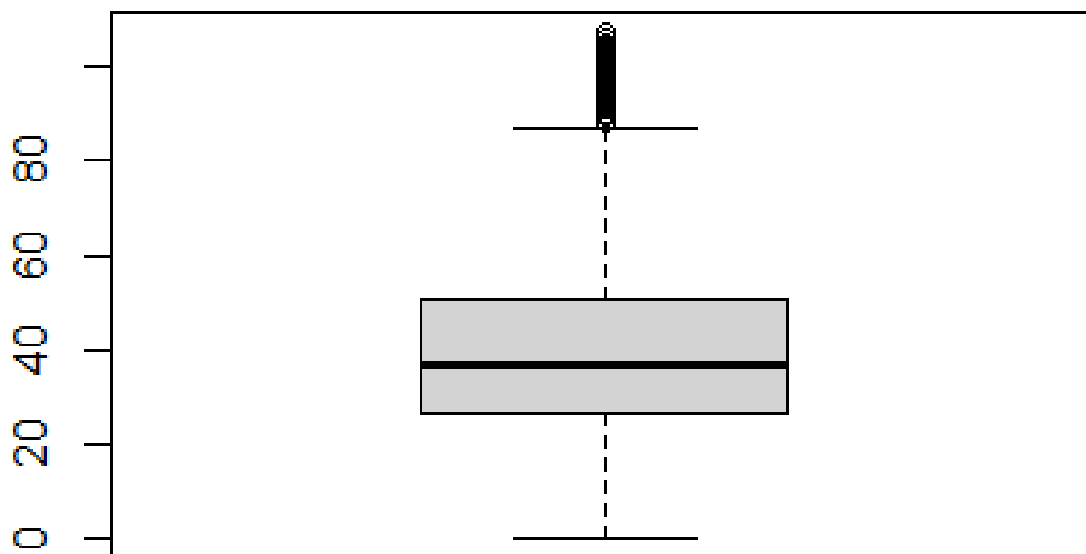


- Quitar filas duplicadas

```
COVID19 <- COVID19 [!duplicated(COVID19), ]
```

- Graficar las cantidades de filas en columnas numéricas, identificar outliers

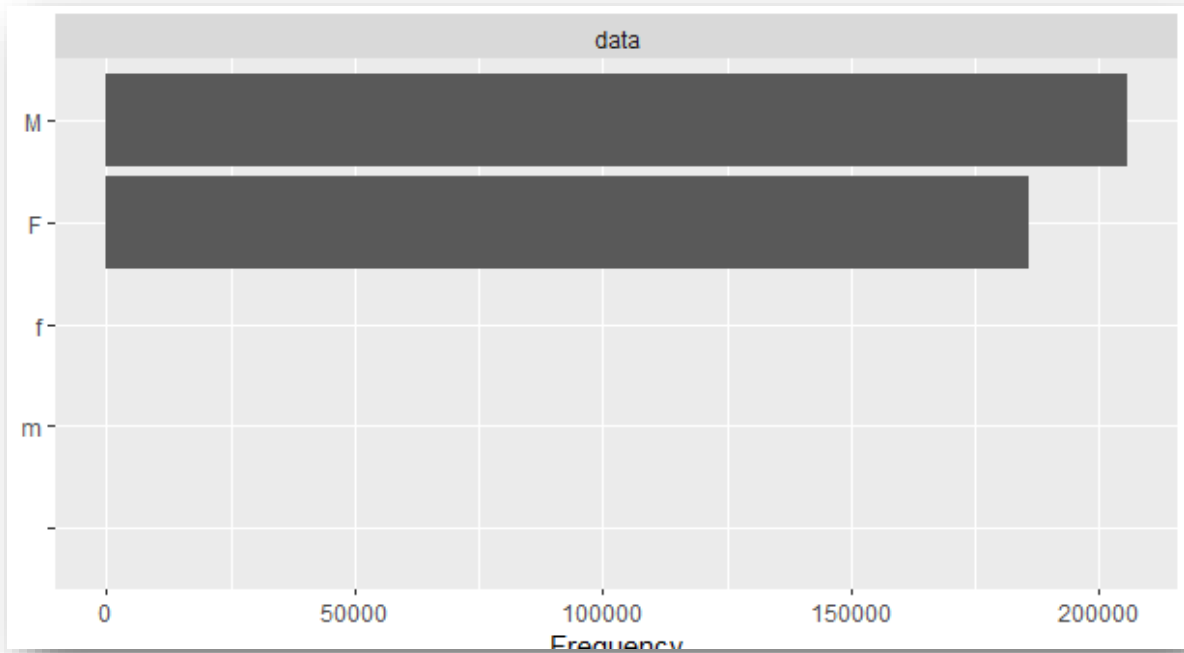
```
boxplot(COVID19$Edad, na.rm=TRUE)
```



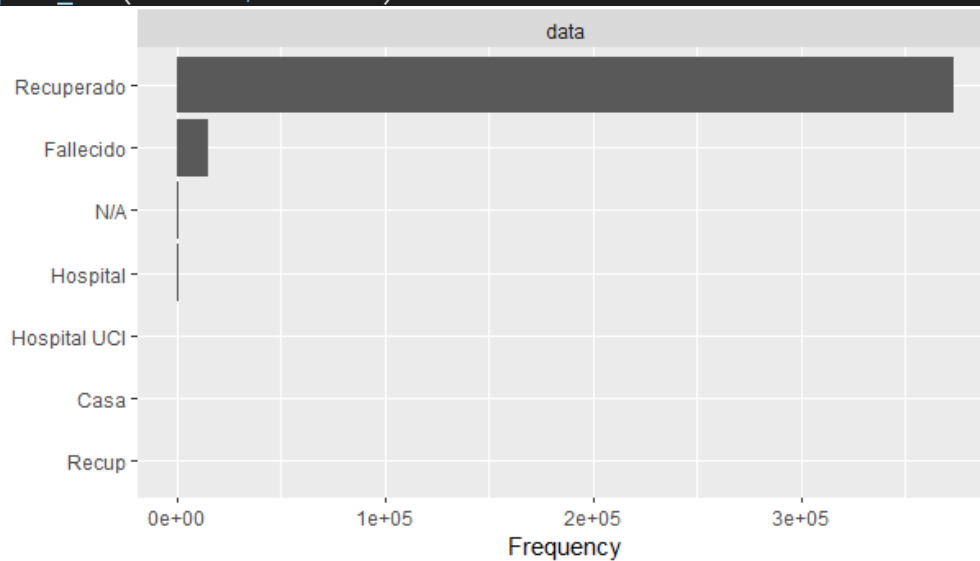


- Identificar errores tipográficos:

```
plot_bar(COVID19$Sexo)
```

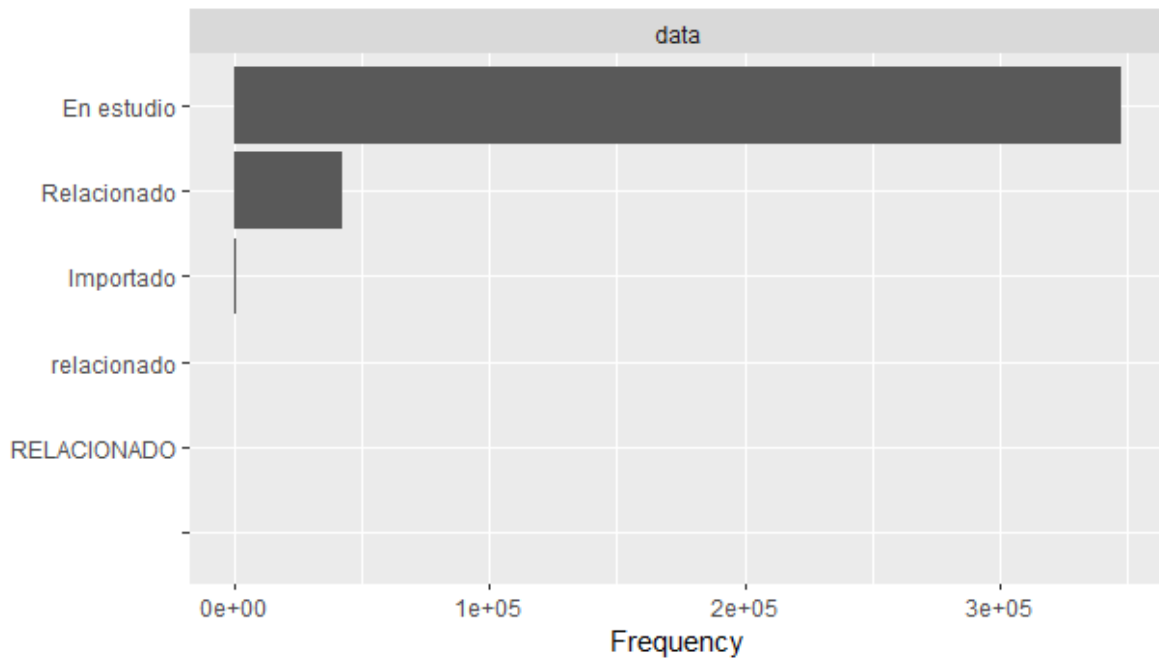


```
plot_bar(COVID19$atención)
```

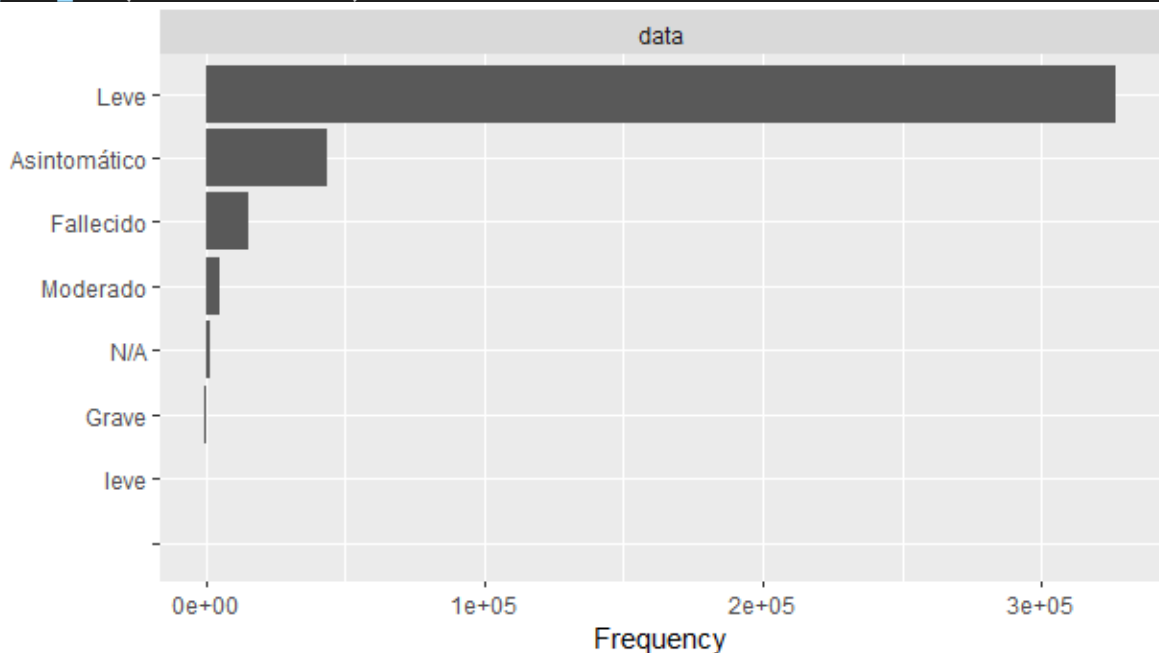




```
plot_bar(COVID19$Tipo)
```

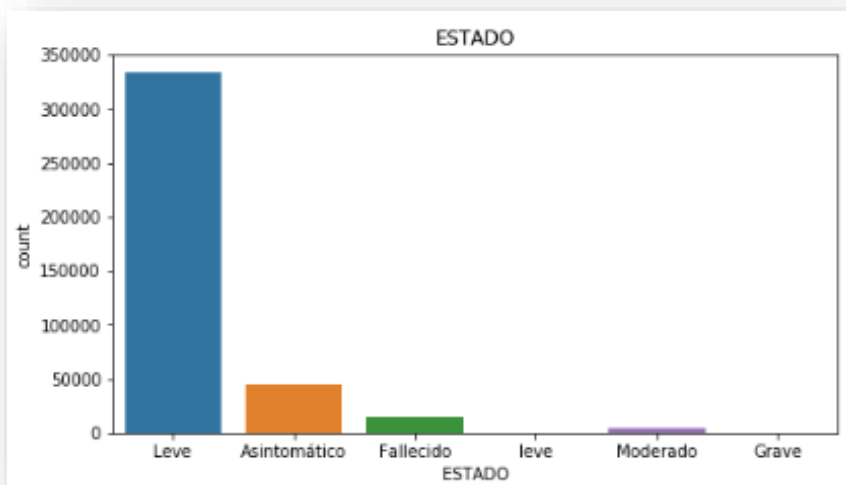


```
plot_bar(COVID19$Estado)
```



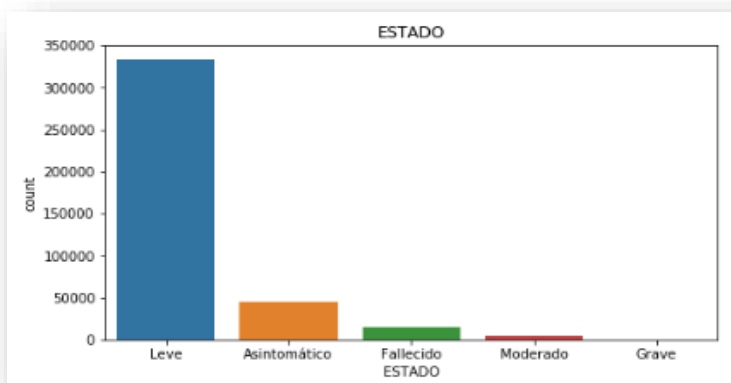


5. Cambios de errores tipográficos.



En el caso de la variable “Estado”, encontramos valores diferentes que fueron escritos mal, por ejemplo “Leve y leve”.

```
COVID19$Estado <- str_replace(COVID19$Estado, "leve", "Leve")
```



El instructor revisa el avance de la práctica y evalúa su desempeño-



6. CONSTRUYENDO DIMENSIONES DEL COVID19



Prepara las dimensiones para el modelo estrella

```
#Convertir a CSV la dimensión "DM_ESTADO"
#extraer una columna a un nuevo dataframe
DM_ESTADO <- COVID19[c(10,10)]
#QUITAR DUPLICADOS
DM_ESTADO<-DM_ESTADO[!duplicated(DM_ESTADO), ]
#COLOCAR INDICE
x=length(DM_ESTADO$Estado)
DM_ESTADO$Estado<-c(1:x)
#Borrar una fila
DM_ESTADO<-DM_ESTADO[-c(8), ]
# RENOMBRA COLUMNAS
colnames(DM_ESTADO)[1]<- 'IDESTADO'
colnames(DM_ESTADO)[2]<- 'NOMBRE'
#exporta a csv el dataframe DM_ESTADO
write.csv(DM_ESTADO,'c:/Borrar/DM_ESTADO.csv', fileEncoding = "UTF-8",row.names = FALSE)
```

Enlazamos en la tabla de hecho la columna "ESTADO" con la dimensión "DM_ESTADO" con el "IDESTADO" correspondiente. Debemos cargar la librería "*stringr*"

```
library(stringr)
```

Procedemos a enlazar el código con el nombre del estado.

	ID.de.caso	Estado
1	1	Leve
2	2	Leve
3	3	Leve
4	4	Leve
5	5	Leve
6	6	Leve
7	7	Leve
8	8	Leve
9	9	Leve
10	10	Leve

```
DM_ESTADO[1,2]
x<-nrow(DM_ESTADO)
for (i in 1:x) {
```



```
z<-DM_ESTADO[i,2]
w<-as.character(DM_ESTADO[i,1])
COVID19$Estado<-str_replace(COVID19$Estado, z, w)
}
print(COVID19[c(1,10)])
```

	ID. de. caso	Estado
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	6	1
7	7	1
8	8	1
9	9	1
10	10	1

Renombramos la columna “ESTADO” como “IDESTADO”

```
colnames(COVID19)[10]<- 'IDESTADO'
```

	ID. de. caso	IDESTADO
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	6	1
7	7	1
8	8	1
9	9	1
10	10	1



El instructor revisa el avance de la práctica y evalúa su desempeño-



Remplazar país vacío con “Colombia” y cambiar título de la columna 11

```
colnames(COVID19)[11]<- 'PAIS'
COVID19$PAIS[COVID19$PAIS==""] <- "COLOMBIA"
```

Cargar la librería “stringr”, para utilizar el método “str_replace”

```
if(!require(stringr)) {
  install.packages("stringr")
}
library(stringr)
```



Corregir errores tipográficos

```
COVID19$PAIS<-str_replace(COVID19$PAIS, "ESTADOS
UNIDOS DE AMÉRICA", "ESTADOS UNIDOS")
COVID19$PAIS<-str_replace(COVID19$PAIS, "ESTADOS
UNIDOS DE AMERICA", "ESTADOS UNIDOS")
COVID19$PAIS<-str_replace(COVID19$PAIS, "MÉXICO",
"MEXICO")
COVID19$PAIS<-str_replace(COVID19$PAIS, "CANADÁ",
"CANADA")
```

```
COVID19$PAIS<-str_replace(COVID19$PAIS, "ARABIA SAUDÍ", "ARABIA SAUDITA")
COVID19$PAIS<-str_replace(COVID19$PAIS, "PERÚ", "PERU")
COVID19$PAIS<-str_replace(COVID19$PAIS, "PANAMÁ", "PANAMA")
```

Crear el dataframe “DM_PAIS”

```
DM_PAIS<-COVID19[,c(11,11)]
```

Quitar filas duplicadas

```
DM_PAIS<-DM_PAIS[!duplicated(DM_PAIS), ]
```

Cambiar títulos de las columnas

```
colnames(DM_PAIS)[1]<- 'IDPAIS'
colnames(DM_PAIS)[2]<- 'NOMBRE'
```

Cuento cuantas filas hay en el dataframe



```
x=nrow(DM_PAIS)
```

Coloco un índice

```
DM_PAIS$IDPAIS<-1:x
```

Guardo el dataframe en un archivo CSV

```
write.csv(DM_PAIS, 'c:/Borrar/DM_PAIS.csv', fileEncoding = "UTF-8", row.names = FALSE)
```

Enlazar tabla de hecho con una dimensión (JOIN)

```
COVID19<-merge(COVID19,DM_PAIS,by.x = "PAIS",by.y = "NOMBRE",all.x = TRUE)
```

Mueve las columnas

```
COVID19 <- COVID19 [, c (2:11)]
```



El instructor revisa el avance de la practica y evalúa su desempeño-

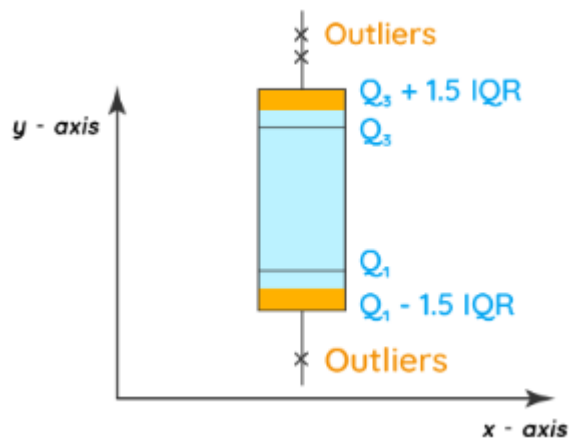
EJERCICIO EN EL AMBIENTE DE FORMACIÓN.



El jefe del proyecto quiere lo siguiente:

1. Después de categorizar la edad en: “niño, joven, adulto y adulto mayor”, debe cambiar la categoría de los niños que tienen el sexo femenino por “niña”.
2. Separe matemáticamente (sin utilizar formulas), el código DIVIPOLA en código del departamento y el código del municipio en las columnas “IDDPTO” y “IDMUNICIPIO” respectivamente.
3. Remplace el código de los países de la dimensión (DM_PAIS), por el código internacional y el código ISO3, por ejemplo: 170 - COL, después sincronice la tabla de hecho con los cambios realizados en la dimensión. Ver [aquí](#) los códigos internacionales.
4. Colocar que día de la semana (en español), se hizo el registro de acuerdo a la fecha del reporte del caso en una columna llamada “DSEMANA”.
5. Colocar el nombre del año chino del contagiado (en español), en una columna llamada “ACHINO”, de acuerdo a su edad.
6. Colocar el nombre de signo zodiacal del contagiado (en español), en una columna llamada “SZODIACO”, de acuerdo a su edad.

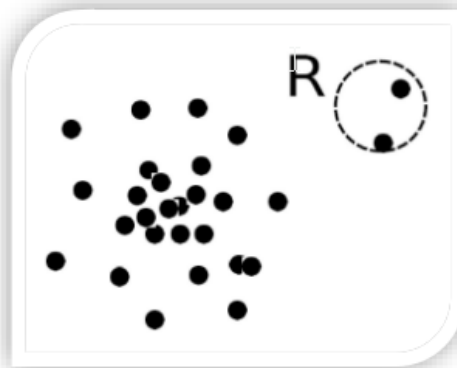
7. ENCONTRANDO VALORES ATÍPICOS (OUTLIERS)



¿Qué es un valor atípico?

Un valor atípico (OUTLIER), es un objeto de datos que se desvía significativamente de los objetos normales, valores que distan mucho del resto de conjunto de datos. como si fuera generado por un mecanismo diferente, hay que tener en cuenta:

- Los valores atípicos son diferentes de los datos de ruido
- El ruido es error o varianza aleatoria en una variable medida.
- El ruido debe ser removido antes de la detección de valores atípicos
- Los valores atípicos son interesantes: viola el mecanismo que genera los datos normales





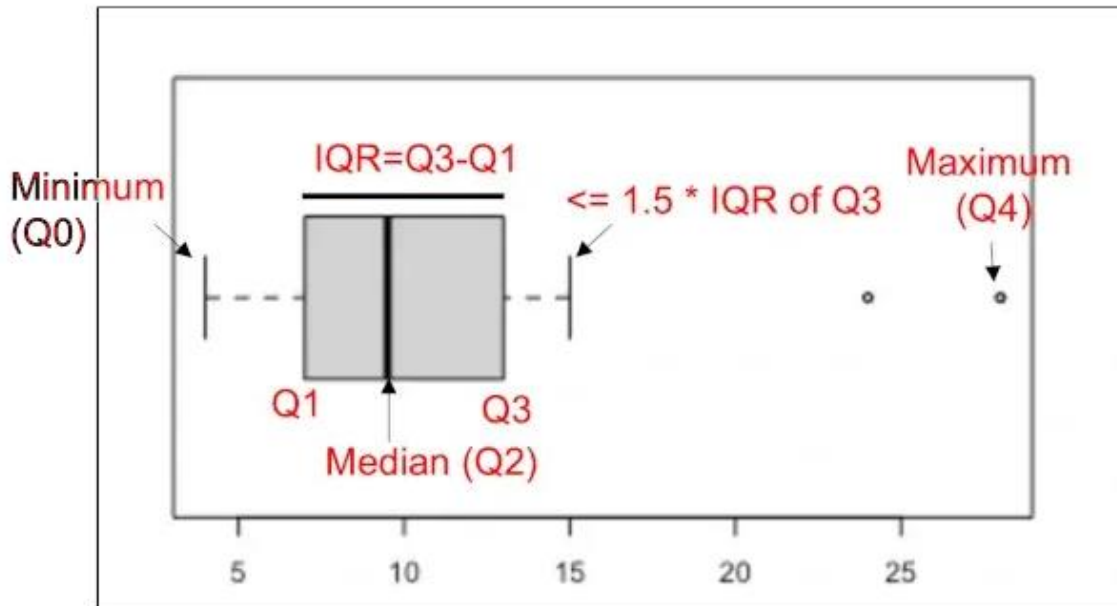
Aplicación

- Detección de fraudes en tarjetas de crédito.
- Detección de fraudes de telecomunicaciones
- Segmentación de clientes
- Análisis médicos

Causas de los OUTLIERS

- Pobre calidad de los datos / contaminación
- Mediciones de baja calidad, equipo que funcione incorrectamente, error manual
- Datos correctos pero excepcionales

```
summary(COVID19$Edad)
res<-quantile(COVID19$Edad, c(0,0.25,0.5,0.75,1))
q1=res[1]
q2=res[2]
q3=res[3]
q4=res[4]
q1
q2
q3
q4
IQR(COVID19$Edad)
IQR(COVID19$Edad,na.rm=TRUE)
iqr=q4-q2
iqr
li=iqr-1.5*iqr
ls=iqr+1.5*iqr
atipicos=COVID19[COVID19$Edad>iqr+1.5*iqr|COVID19$Edad<iqr-1.5*iqr,]
```



Prueba de Grubbs



Permite detectar si el valor más alto o más bajo en un conjunto de datos es un valor atípico. Se detecta un valor atípico a la vez (valor más alto o más bajo), por lo que las hipótesis nula y alternativa son H_0 : el valor más alto no es un valor atípico y H_1 : el valor más alto es un valor atípico, si queremos probar el valor más

alto; H_0 : el valor más bajo no es un valor atípico, H_1 : el valor más bajo es un valor atípico, si queremos probar el valor más bajo.

Como para cualquier prueba estadística, si el valor p es menor que el umbral de significancia elegido (generalmente $\alpha = 0.05$), entonces se rechaza la hipótesis nula y concluiremos que el valor más bajo / más

alto es un valor atípico. Por el contrario, si el valor p es mayor o igual que el nivel de significancia, la hipótesis nula no se rechaza y concluiremos que, con base en los datos, no rechazamos la hipótesis de que el valor más bajo / más alto no es un valor atípico. **Tenga en cuenta que la prueba de Grubbs no es apropiada para un tamaño de muestra de 6 o menos ($n \leq 6$).** Para realizar la prueba de Grubbs en R, usamos la función `grubbs.test` del paquete `outliers`.

```
test <- grubbs.test(mpg$hwy)
test
```

La prueba Q de Dixon¹



La «prueba Q», es una forma de encontrar valores atípicos en conjuntos de datos muy pequeños, normalmente distribuidos. Los conjuntos de datos pequeños generalmente se definen como entre 3 y 7 elementos. Mantener un valor atípico en los datos afecta a cálculos como la media y la desviación estándar, por lo que se deben eliminar los valores atípicos verdaderos.

Se usa para probar si un solo valor es un valor atípico en un tamaño de muestra de entre 3 y 7. Dean y Dixon sugirieron varias otras fórmulas en un papel posterior, pero estos no se usan comúnmente.

Pasos para aplicar la prueba Q:

Paso 1: Ordene sus datos en orden ascendente (de menor a mayor).

Paso 2: Encuentre la estadística Q usando la siguiente fórmula: Donde:

¹ <https://statologos.com/prueba-q-de-dixon-2/>

$$Q_{exp} = \frac{x_2 - x_1}{x_n - x_1}$$

- x_1 es el valor más pequeño (sospechoso),
- x_2 es el segundo valor más pequeño,
- x_n es el valor más grande.

Paso 3: Encuentre el valor crítico Q en la tabla Q

Paso 4: Compare la estadística Q del Paso 2 con el valor crítico Q del Paso 3.

```
test <- dixon.test(COVID19$Edad)
test
Error in dixon.test(COVID19$Edad, type = 10) :
  Sample size must be in range 3-30
```

Hampel filter



Otro método, conocido como filtro de Hampel, consiste en considerar como valores atípicos los valores fuera del intervalo formado por $[mediana - 3 \cdot MAD ; mediana + 3 \cdot MAD]$, donde MAD es la mediana de las desviaciones absolutas de la mediana de los datos. Para este método, primero establecemos los límites de intervalo gracias a las funciones `median` y `mad`:

```
(median(COVID19$Edad) - 3 * mad(COVID19$Edad))
-16.3736
```

Prueba de Rosner



La prueba de Rosner para valores atípicos tiene las ventajas de que:

- se utiliza para detectar conjuntos de valores atípicos (a diferencia de la prueba de Grubbs y Dixon, que muestran valores individuales),
- está diseñado para evitar el problema del enmascaramiento, donde un valor atípico que tiene un valor cercano a otro valor atípico puede pasar desapercibido y,
- es más apropiada cuando el tamaño de la muestra es grande ($n \geq 20$).



Para realizar la prueba de Rosner usamos la función `rosnerTest` del paquete `EnvStats`. Esta función requiere al menos 2 argumentos: los datos y el número de valores atípicos sospechosos k (con $k = 3$ como el número predeterminado de valores atípicos sospechosos). En nuestro caso usamos que el número de valores atípicos sospechosos sea igual a 3, como sugiere el número de *outliers* potenciales descritos en el diagrama de caja al comienzo del artículo.

```
install.packages("EnvStats")
library(EnvStats)
test <- rosnerTest(COVID19$Edad)
test
Results of Outlier Test
-----
```




```
Test Method: Rosner's Test for Outliers

Hypothesized Distribution: Normal

Data: COVID19$Edad

Sample Size: 400489

Test Statistics: R.1 = 3.745404
                 R.2 = 3.690086
                 R.3 = 3.690153

Test Statistic Parameter: k = 3

Alternative Hypothesis: Up to 3 observations are not
                        from the same Distribution.

Type I Error: 5%

Number of Outliers Detected: 0

  i  Mean.i    SD.i Value Obs.Num   R.i+1 lambda.i+1 Outlier
1 0 39.37865 18.05449   107  241086 3.745404   5.286164   FALSE
2 1 39.37848 18.05419   106  139006 3.690086   5.286163   FALSE
3 2 39.37831 18.05391   106  175168 3.690153   5.286163   FALSE
```



El instructor revisa el avance de la practica y evalúa su desempeño-



8. MANIPULACION DE DATOS CON SQL DESDE R

- Instalar el paquete "sqldf"ⁱ

```
install.packages("sqldf")  
library(sqldf)
```

- Cargar el DATASET

```
COVID19<-read.csv("https://siomi.datasena.com/analitica/data/COVID19.csv",  
sep=";", header = TRUE, encoding = "UTF-8")
```

- Realizar la consulta "select" hacia un dataframe "a"

```
a<-sqldf("select * from COVID19")
```

```
> head(a)
```

	ID. de. caso	Fecha	Código. DIVIPOLA	Ciudad. de. ubicación
1	1	2020-03-02T00:00:00.000	11001	Bogotá D.C.
2	2	2020-03-06T00:00:00.000	76111	Guadalajara de Buga
3	3	2020-03-07T00:00:00.000	5001	Medellín
4	4	2020-03-09T00:00:00.000	5001	Medellín
5	5	2020-03-09T00:00:00.000	5001	Medellín
6	6	2020-03-10T00:00:00.000	5360	Itagüí

	Departamento. o. Distrito.	atención	Edad	Sexo	Tipo	Estado
1	Bogotá D.C.	Recuperado	19	F	Importado	Leve
2	valle del Cauca	Recuperado	34	M	Importado	Leve
3	Antioquia	Recuperado	50	F	Importado	Leve
4	Antioquia	Recuperado	55	M	Relacionado	Leve
5	Antioquia	Recuperado	25	M	Relacionado	Leve
6	Antioquia	Recuperado	27	F	Relacionado	Leve

	País. de. procedencia
1	ITALIA
2	ESPAÑA
3	ESPAÑA

- Utilizar la cláusula "distinct"

```
> sqldf("select distinct atención from COVID19")
```

	atención
1	Recuperado
2	Fallecido
3	N/A
4	Casa
5	Hospital UCI
6	Hospital

- Agrupar por el campo "Sexo"

```
> sqldf("select Sexo, count(*) from COVID19 group by Sexo")
```

	Sexo	count(*)
1	F	190477
2	M	210003
3	f	7
4	m	2



- Agrupar

```
> sqldf("select atención,count(*) from COVID19 group by atención")
  atención count(*)
1      Casa      101
2  Fallecido 15275
3   Hospital  1102
4 Hospital UCI   356
5         N/A  1212
6  Recuperado 382443
```

- Consultar con condicional

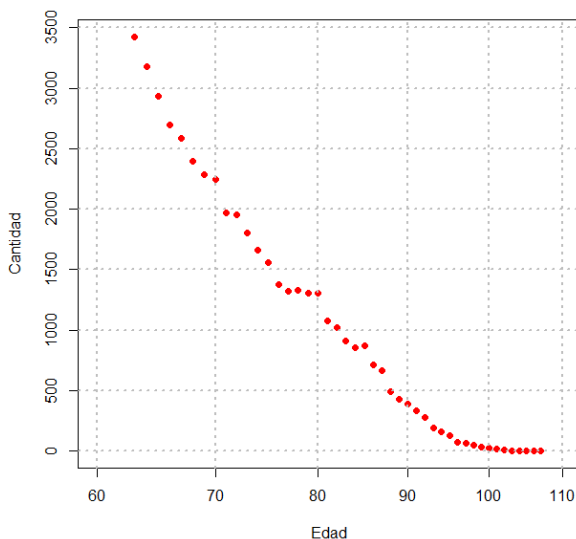
```
> sqldf("select Edad,count(*) from COVID19 where Edad>62 group by Edad")
  Edad count(*)
1    63    3426
2    64    3177
3    65    2933
4    66    2698
5    67    2584
6    68    2398
7    69    2288
```

....

```
38 100      24
39 101      21
40 102       7
41 103       6
42 104       5
43 105       1
44 106       2
45 107       1
```

- Graficando el agrupamiento para adultos mayores infectados

```
a<-sqldf("select Edad,count(*) hay from COVID19 where Edad>62 group by Edad")
plot(a$Edad,a$hay,pch=16,xlab = "Edad",ylab = "Cantidad",col="red",log = "x",xlim=c(60,110))
grid(nx = NULL, ny = NULL,
      lty = 3,      # Tipo de línea
      col = "gray", # Color
      lwd = 2)
```

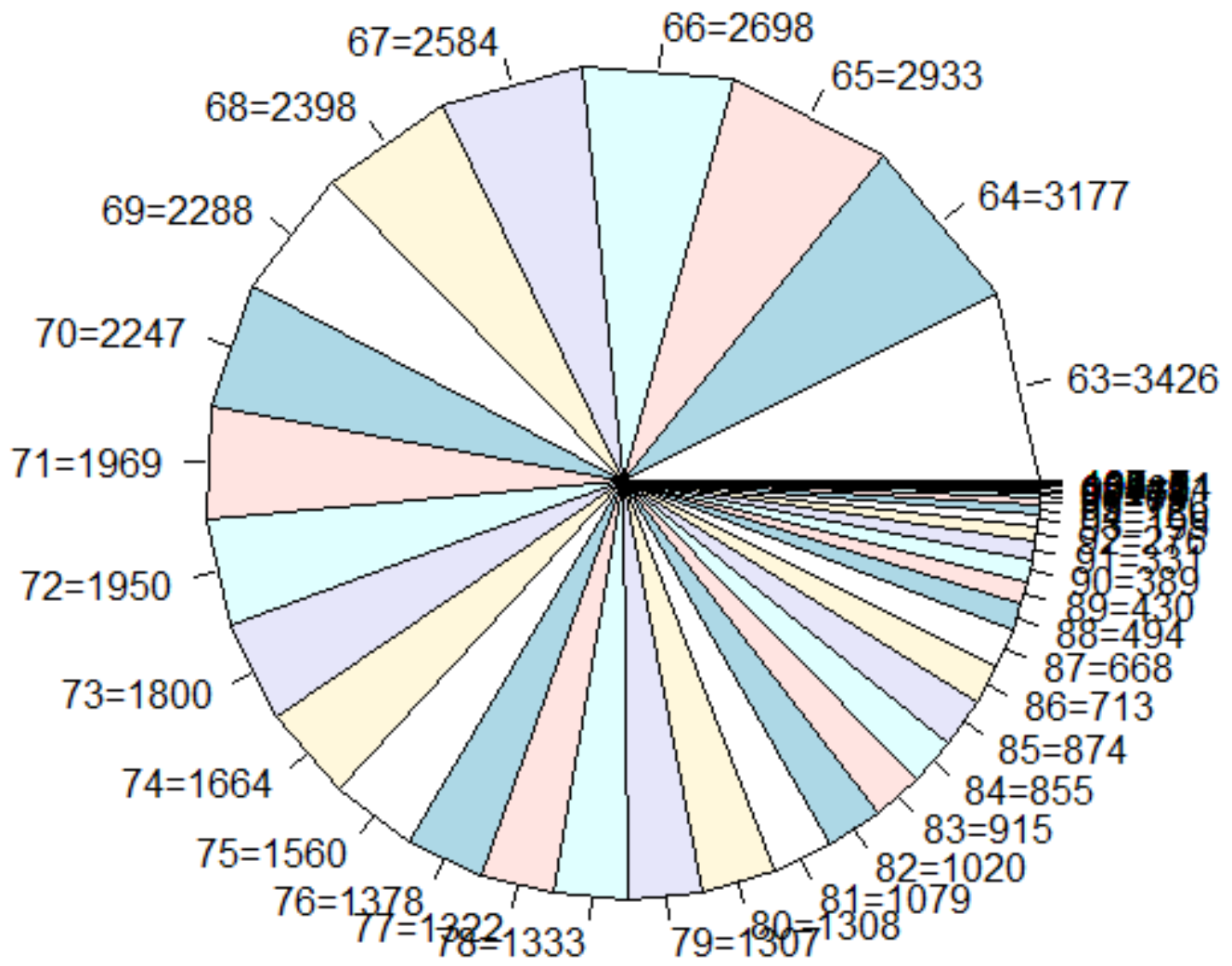




- Torta de Edades Adulto Mayor

```
> a<-sqldf("select Edad,count(*) hay from COVID19 where Edad>62 group by Edad")  
> etiquetas <- paste0(a$Edad, "=", a$hay)  
> pie(a$hay, labels = etiquetas, edges = 10, main="Edad Adulto Mayor")
```

Edad Adulto Mayor





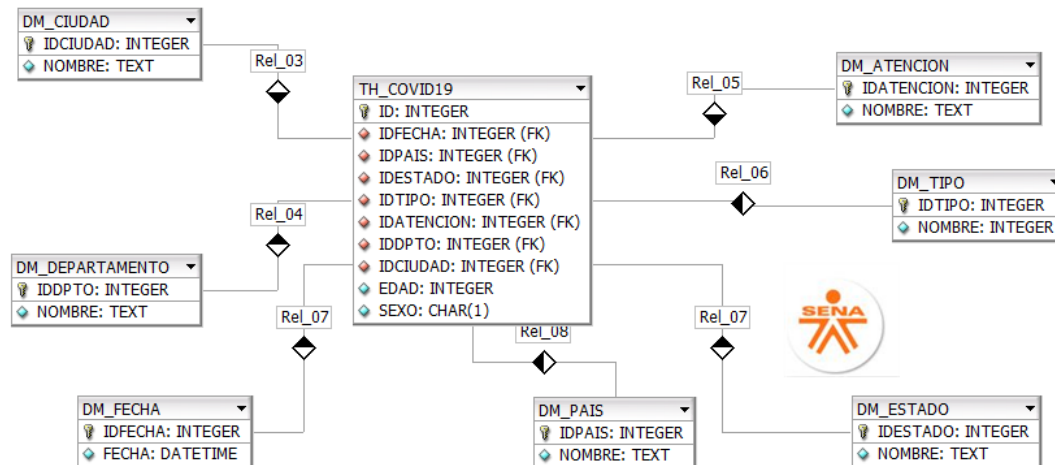
- Utilizar la cláusula “case when”

```
> a<- sqldf("select Edad,  
+ case when Edad <=10 then 'SI' else 'NO' end NIÑO,  
+ case when Edad >10 and Edad<=18 then 'SI' else 'NO' end JOVEN,  
+ case when Edad >18 and Edad<=62 then 'SI' else 'NO' end ADULTO,  
+ case when Edad >62 then 'SI' else 'NO' end 'ADULTO MAYOR'  
+ from COVID19")  
> a
```

	Edad	NIÑO	JOVEN	ADULTO	ADULTO MAYOR
1	19	NO	NO	SI	NO
2	34	NO	NO	SI	NO
3	50	NO	NO	SI	NO
4	55	NO	NO	SI	NO
5	25	NO	NO	SI	NO
6	27	NO	NO	SI	NO
7	85	NO	NO	NO	SI
8	22	NO	NO	SI	NO
9	28	NO	NO	SI	NO
10	36	NO	NO	SI	NO
11	42	NO	NO	SI	NO
12	74	NO	NO	NO	SI
13	68	NO	NO	NO	SI
14	48	NO	NO	SI	NO
15	30	NO	NO	SI	NO
16	61	NO	NO	SI	NO
17	73	NO	NO	NO	SI
18	54	NO	NO	SI	NO
19	54	NO	NO	SI	NO
20	26	NO	NO	SI	NO
21	28	NO	NO	SI	NO
22	36	NO	NO	SI	NO
23	23	NO	NO	SI	NO
24	18	NO	SI	NO	NO
25	49	NO	NO	SI	NO
26	30	NO	NO	SI	NO
27	30	NO	NO	SI	NO
28	22	NO	NO	SI	NO
29	55	NO	NO	SI	NO
30	48	NO	NO	SI	NO



9. MODELO ESTRELLA COVID19



pgAdmin

Management Tools for PostgreSQL

Feature rich | Maximises PostgreSQL | Open Source

👤 Recomendado ver el video [“Crear BD en PostgreSQL”](#)

🗃️ Cargar los datos generados en una base de datos POSTGRESQL llamada “COVID19”, mediante un programa R, de acuerdo al modelo relacional presentado.

Crear la base de datos COVID19 en PostgreSQL:

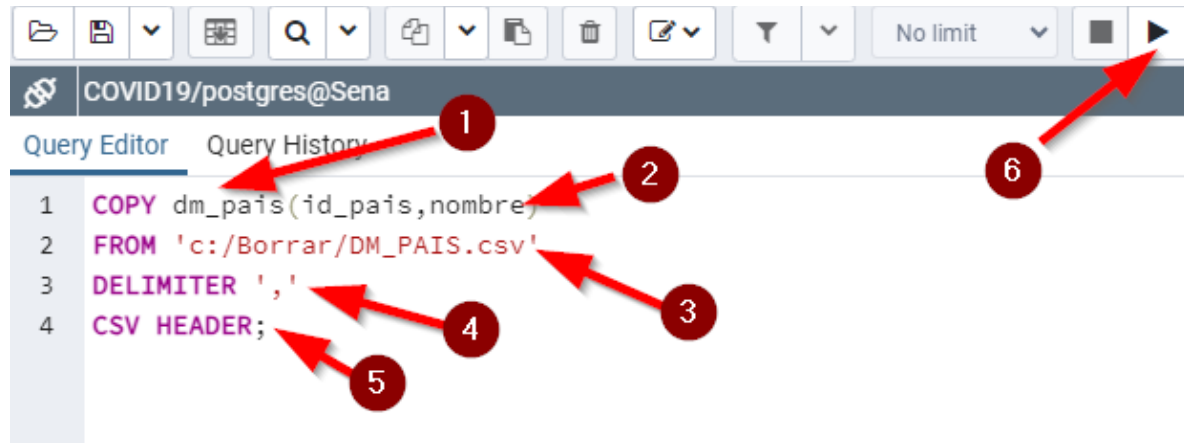


```
CREATE DATABASE "COVID19"  
WITH  
OWNER = postgres  
ENCODING = 'UTF8'  
LC_COLLATE = 'Spanish_Colombia.1252'  
LC_CTYPE = 'Spanish_Colombia.1252'  
TABLESPACE = pg_default  
CONNECTION LIMIT = -1;
```

- 📁 Creamos la tabla DM_PAIS en la base de datos COVID19 en POSTGRESQL
- 📁 Crear la tabla "DM_PAIS"

The screenshot shows the PostgreSQL Query Editor interface. The top bar includes tabs for Dashboard, Properties, SQL, Statistics, Dependencies, and Dependents. The current connection is 'COVID19/postgres@Sena'. The Query Editor tab is active, showing a single query: `CREATE TABLE DM_PAIS(ID_PAIS SERIAL PRIMARY KEY,NOMBRE TEXT)`. This query is highlighted with a red box and a red circle with the number '1'. The Messages tab at the bottom shows the message: 'CREATE TABLE' and 'Query returned successfully in 202 msec.'.

Construir la sentencia COPY así:



10. Conectar R a POSTGRESQL

- Instalar las siguientes librerías:

```
install.packages("RPostgres")
install.packages("DBI")
install.packages("sf")
library(DBI)
library(RPostgres)
library(sf)
```

- Conectando a la base de datos

```
dvr <- RPostgres::Postgres()
db <- 'COVID19' ##Nombre de la BBDD
host_db <- 'localhost'
db_port <- '5432'
db_user <- 'postgres' ##Tu usuario
db_password <- '123' ##Tu contraseña

# 3.0 Conexión
con <- dbConnect(dvr, dbname = db, host=host_db, port=db_port,
                 user=db_user, password=db_password)
```

- Listar las tablas de la base de datos

```
dbListTables(con)
```

- Desplegar las filas de la tabla dmpais

```
dmpais <- st_read(con, layer = "dmpais")
view(dmpais)
```




- Matriz CRUD en POSTGRESQLⁱⁱ

```
install.packages("DBI")
install.packages("sf")
library(DBI)
library(RPostgres)
library(sf)
```

Cargando librerías y conectando con la base de datos POSTGRESQL

```
> library(DBI)
> library(RPostgres)
> library(sf)
>
> dvr <- RPostgres::Postgres()
> db <- 'COVID19' ##Nombre de la BBDD
> host_db <- 'localhost'
> db_port <- '5432'
> db_user <- 'postgres' ##Tu usuario
> db_password <- '123' ##Tu contraseña
>
> # 3.0 Conexión
> con <- dbConnect(dvr, dbname = db, host=host_db, port=db_port,
+                  user=db_user, password=db_password)
```

- Realizar la consulta "select"

```
> res <- dbSendQuery(con, "SELECT * FROM public.dm_pais limit 10")
> dbFetch(res)
  idpais      nombre
1      1      ITALIA
2      2     ESPAÑA
3      3    COLOMBIA
4      4 ESTADOS UNIDOS
5      5     ECUADOR
6      6     FRANCIA
7      7     TURQUÍA
8      8    ALEMANIA
9      9     BRASIL
10     10    CROACIA
> dbClearResult(res)
```



- Insertar un registro nuevo

```
> res <- dbSendQuery(con, "INSERT INTO public.dm_pais VALUES(45,'SOMONDOCO')")  
> dbClearResult(res)
```

Data Output	Explain	Messages	Notifications
idpais [PK] integer	nombre text		
40	40	SUECIA	
41	41	CURAZAO	
42	42	NICARAGUA	
43	43	PORTUGAL	
44	44	REPÚBLICA...	
45	45	SOMONDO...	

- Actualizar un registro

```
> res <- dbSendQuery(con, "UPDATE public.dm_pais SET NOMBRE='SUTAPON' WHERE idpais=45")  
> dbClearResult(res)
```

Data Output	Explain	Messages	Notifications
idpais [PK] integer	nombre text		
40	40	SUECIA	
41	41	CURAZAO	
42	42	NICARAGUA	
43	43	PORTUGAL	
44	44	REPÚBLICA...	
45	45	SUTAPON	



- Borrar un registro

```
> res <- dbSendQuery(con, "DELETE FROM public.dm_pais WHERE idpais=45")  
> dbClearResult(res)
```

Data Output Explain Messages Notifications

	idpais [PK] integer	nombre text
39	39	URUGUAY
40	40	SUECIA
41	41	CURAZAO
42	42	NICARAGUA
43	43	PORTUGAL
44	44	REPÚBLICA...

11. CONECTAR R A MONGODB



- Antes de conectar a MONGODB se debe instalar el servidor y correr en MONGOSH el siguiente [JSON](#)
- Se recomienda ver el video de [creación de BD en MONGODB](#)
- Instalando en R los paquetes de MONGODB

```
> install.packages("mongolite", dependencies = TRUE)
```

- Conectado a MONGODB con una base de datos en ATLAS

```
> mongo(  
+   collection = "casos",  
+   db = "COVID19",  
+   url = "mongodb+srv://fegasu:CursoSena2023@cluster0.jmeme59.mongodb.net/tls=true?tls=true",  
+   verbose = FALSE,  
+   options = ssl_options()  
+ )
```

- Conectando a la colección



Servicio Nacional de Aprendizaje
Formato Taller
Centro de Gestión de Mercados, Logística y Tecnologías de la Información.

```
> dtc_R <- mongo(  
+   collection = "casos",  
+   db = "covid19"  
+ )
```

- Listar todos los documentos

```
> dtc_R$find('{}')
```

ID_de_caso	Fecha	Código_DIVIPOLA	Ciudad_de_ubicación
1	2020-03-02T00:00:00.000	11001	Bogotá D.C.
2	2020-03-06T00:00:00.000	76111	Guadalajara de Buga
3	2020-03-07T00:00:00.000	5001	Medellín
4	2020-03-09T00:00:00.000	5001	Medellín
5	2020-03-09T00:00:00.000	5001	Medellín
6	2020-03-10T00:00:00.000	5360	Itagüí
7	2020-03-08T00:00:00.000	13001	Cartagena de Indias
8	2020-03-09T00:00:00.000	11001	Bogotá D.C.
9	2020-03-08T00:00:00.000	11001	Bogotá D.C.
10	2020-03-12T00:00:00.000	11001	Bogotá D.C.
11	2020-03-11T00:00:00.000	11001	Bogotá D.C.
12	2020-03-10T00:00:00.000	41001	Neiva
13	2020-03-10T00:00:00.000	41001	Neiva
14	2020-03-10T00:00:00.000	76520	Palмира
15	2020-03-13T00:00:00.000	50001	villavicencio
16	2020-03-11T00:00:00.000	11001	Bogotá D.C.

Convirtiendo a DATAFRAME

```
a<-data.frame(dtc_R$find('{}'))  
view(a)
```

ID_de_caso	Fecha	Código_DIVIPOLA	Ciudad_de_ubicación	Departamento_o_Distrito	atención
1	2020-03-02T00:00:00.000	11001	Bogotá D.C.	Bogotá D.C.	R...
2	2020-03-06T00:00:00.000	76111	Guadalajara de Buga	Valle del Cauca	R...
3	2020-03-07T00:00:00.000	5001	Medellín	Antioquia	R...
4	2020-03-09T00:00:00.000	5001	Medellín	Antioquia	R...
5	2020-03-09T00:00:00.000	5001	Medellín	Antioquia	R...
6	2020-03-10T00:00:00.000	5360	Itagüí	Antioquia	R...
7	2020-03-08T00:00:00.000	13001	Cartagena de Indias	Cartagena D.T. y C.	R...
8	2020-03-09T00:00:00.000	11001	Bogotá D.C.	Bogotá D.C.	R...
9	2020-03-08T00:00:00.000	11001	Bogotá D.C.	Bogotá D.C.	R...
10	2020-03-12T00:00:00.000	11001	Bogotá D.C.	Bogotá D.C.	R...
11	2020-03-11T00:00:00.000	11001	Bogotá D.C.	Bogotá D.C.	R...
12	2020-03-10T00:00:00.000	41001	Neiva	Huila	R...
...

Showing 1 to 14 of 250 entries

- Consultar el documento cuyo id es 1



Servicio Nacional de Aprendizaje
Formato Taller
Centro de Gestión de Mercados, Logística y Tecnologías de la Información.

```
> data.frame(dtc_R$find('{"ID_de_caso":1}'))  
ID_de_caso      Fecha CÃ³digo_DIVIPOLA Ciudad_de_ubicaciÃ³n  
1          1 2020-03-02T00:00:00.000      11001      Bogotá D.C.  
Departamento_o_Distrito_ atenciÃ³n Edad Sexo      Tipo Estado  
1          Bogotá D.C. Recuperado   19    F Importado   Leve  
PaÃs_de_procedencia  
1          ITALIA
```

- Consultar los documentos cuyo sexo es igual a "F"

```
> a<-data.frame(dtc_R$find('{"Sexo":"F"}'))  
> view(a)
```

ID_de_caso	Fecha	CÃ³digo_DIVIPOLA	Ciudad_de_ubicaciÃ³n	Departamento_o_Distrito_	atenciÃ³n	Edad	Sexo
1	2020-03-02T00:00:00.000	11001	Bogotá D.C.	Bogotá D.C.	Recuperado	19	F
2	2020-03-07T00:00:00.000	5001	Medellín	Antioquia	Recuperado	50	F
3	2020-03-10T00:00:00.000	5360	Itagüí	Antioquia	Recuperado	27	F
4	2020-03-08T00:00:00.000	13001	Cartagena de Indias	Cartagena D.T. y C.	Recuperado	85	F
5	2020-03-09T00:00:00.000	11001	Bogotá D.C.	Bogotá D.C.	Recuperado	22	F
6	2020-03-08T00:00:00.000	11001	Bogotá D.C.	Bogotá D.C.	Recuperado	28	F
7	2020-03-12T00:00:00.000	11001	Bogotá D.C.	Bogotá D.C.	Recuperado	36	F
8	2020-03-11T00:00:00.000	11001	Bogotá D.C.	Bogotá D.C.	Recuperado	42	F
9	2020-03-10T00:00:00.000	41001	Neiva	Huila	Recuperado	74	F
10	2020-03-10T00:00:00.000	41001	Neiva	Huila	Recuperado	68	F
11	2020-03-13T00:00:00.000	50001	Villavicencio	Meta	Recuperado	30	F
12	2020-03-11T00:00:00.000	11001	Bogotá D.C.	Bogotá D.C.	Recuperado	61	F

12. CONECTAR R A MYSQL



- Instalar el paquete "RMySQL"

```
> install.packages("RMySQL")
```

- Usar la librería "RMySQL"

```
> library(RMySQL)
```

- Conectar al servidor MySQL

```
> con = dbConnect(RMySQL::MySQL(), user="root", password = "", dbname="covid19")
```

- Listar las filas de DM_PAIS en el dataframe "df"

```
> sql <- "select * from dm_pais"
> rs <- dbSendQuery(con, sql)
> df <- fetch(rs, n = -1)
```

	ID_PAIS	NOMBRE
1	1	ITALIA
2	2	ESPAÑA
3	3	COLOMBIA
4	4	ESTADOS UNIDOS
5	5	ECUADOR
6	6	FRANCIA
7	7	TURQUÍA
8	8	ALEMANIA
9	9	BRASIL

Showing 1 to 10 of 44 entries

- Insertar un registro en la tabla "DM_PAIS"

```
> sql <- "insert into dm_pais values(45,'SOMONDOCO')"  
> rs <- dbSendQuery(con, sql)
```

```
MariaDB [covid19]> select * from dm_pais WHERE ID_PAIS=45;  
+-----+-----+  
| ID_PAIS | NOMBRE |  
+-----+-----+  
|      45 | SOMONDOCO |  
+-----+-----+  
1 row in set (0.001 sec)
```

- Actualizar el registro 45

```
> sql <- "UPDATE dm_pais SET NOMBRE='SUTAPON' WHERE ID_PAIS=45"  
> rs <- dbSendQuery(con, sql)
```

```
MariaDB [covid19]> select * from dm_pais WHERE ID_PAIS=45;  
+-----+-----+  
| ID_PAIS | NOMBRE |  
+-----+-----+  
|      45 | SUTAPON |  
+-----+-----+  
1 row in set (0.001 sec)
```

- Borrando el registro 45

```
> sql <- "DELETE FROM dm_pais WHERE ID_PAIS=45"  
> rs <- dbSendQuery(con, sql)
```

```
MariaDB [covid19]> select * from dm_pais WHERE ID_PAIS=45;  
Empty set (0.001 sec)
```

CONECTAR R CON SQLITE



- Descargue el SCRIPT de [creación de la base de datos](#)
- Se recomienda el video de [creación de la base de datos sqlite](#)
- Instalar el paquete "RMySQL"

```
> install.packages("RSQLite")
```



Servicio Nacional de Aprendizaje
Formato Taller
Centro de Gestión de Mercados, Logística y Tecnologías de la Información.

- Usar la librería "RMySQL"
> library(RSQLite)
- Conectar al servidor MySQL
con <- dbConnect(RSQLite::SQLite(), "c:/Borrar/COVID19.db")
- Listar las filas de COVID19 en el dataframe "df"
sql <- "select * from COVID19"
rs <- dbSendQuery(con, sql)
df <- fetch(rs, n = -1)

IDde caso	Fecha	CódigoDIVIPOLA	Ciudaddeubicación	DepartamentooDistrito	atención	Edad	Sexo
1	2020-03-02T00:00:00.000	11001	Bogotá D.C.	Bogotá D.C.	Recuperado	19	F
2	2020-03-06T00:00:00.000	76111	Guadalajara de Buga	Valle del Cauca	Recuperado	34	M
3	2020-03-07T00:00:00.000	5001	Medellín	Antioquia	Recuperado	50	F
4	2020-03-09T00:00:00.000	5001	Medellín	Antioquia	Recuperado	55	M
5	2020-03-09T00:00:00.000	5001	Medellín	Antioquia	Recuperado	25	M
6	2020-03-10T00:00:00.000	5360	Itagüí	Antioquia	Recuperado	27	F
7	2020-03-08T00:00:00.000	13001	Cartagena de Indias	Cartagena D.T. y C.	Recuperado	85	F
8	2020-03-09T00:00:00.000	11001	Bogotá D.C.	Bogotá D.C.	Recuperado	22	F
9	2020-03-08T00:00:00.000	11001	Bogotá D.C.	Bogotá D.C.	Recuperado	28	F

Showing 1 to 10 of 250 entries



13. MUESTRA DE LA POBLACION CON LIMITACIONES FISICA EN SB112012.

```
library(sqldf)
sqldf("SELECT DISTINCT ESTU_LIMITA_SORDOCEGUERA FROM SB112012") #C
sqldf("SELECT DISTINCT ESTU_LIMITA_COGNITIVA FROM SB112012") #G
sqldf("SELECT DISTINCT ESTU_LIMITA_INVIDENTE FROM SB112012") #I
sqldf("SELECT DISTINCT ESTU_LIMITA_MOTRIZ FROM SB112012") #M
sqldf("SELECT DISTINCT ESTU_LIMITA_SORDONINTERPRETE FROM SB112012") #S
sqldf("SELECT DISTINCT ESTU_LIMITA_SORDOINTERPRETE FROM SB112012") #R
Borrador<-SB112012[,c(2:14,16:20,100:112)]
Borrador1<-Borrador[Borrador$ESTU_LIMITA_SORDOCEGUERA=='C',]
Borrador2<-Borrador[Borrador$ESTU_LIMITA_COGNITIVA=="G",]
Borrador3<-Borrador[Borrador$ESTU_LIMITA_INVIDENTE=="I",]
Borrador4<-Borrador[Borrador$ESTU_LIMITA_MOTRIZ=="M",]
Borrador5<-Borrador[Borrador$ESTU_LIMITA_SORDONINTERPRETE=="S",]
Borrador6<-Borrador[Borrador$ESTU_LIMITA_SORDOINTERPRETE=="R",]
Borrador7<-rbind.data.frame(Borrador1,Borrador2,Borrador3,Borrador4,Borrador5,Borrador6)
View(Borrador7)
```

Data						
▶ Borrador	45390 obs. of 31 variables					
▶ Borrador1	2 obs. of 31 variables					
▶ Borrador2	16 obs. of 31 variables					
▶ Borrador3	18 obs. of 31 variables					
▶ Borrador4	89 obs. of 31 variables					
▶ Borrador5	10 obs. of 31 variables					
▶ Borrador6	8 obs. of 31 variables					
▶ Borrador7	143 obs. of 31 variables					
▶ SB112012	45390 obs. of 112 variables					

Filter						
ESTU_TIPO_DOCUMENTO	ESTU_PAIS_RESIDE	ESTU_GENERO	ESTU_NACIMIENTO_DIA	ESTU_NACIMIENTO_MES	ESTU_NAC	
10881 C	CO	M	3	6		▲
40629 C	CO	M	17	7		
4418 C	CO	M	25	5		
5104 C	CO	M	22	3		
5452 C	CO	M	19	4		
7279 T	CO	F	18	7		
14089 C	CO	M	6	2		
18625 C	CO	F	20	8		
21543 C	CO	M	24	8		▼

Showing 1 to 10 of 143 entries



Pero el más preciso es este, porque en el anterior hay casos que tienen más de una limitación.

```
Borrador$tipo<-0;
Borrador<-Borrador %>% mutate(tipo1=if_else(ESTU_LIMITA_SORDOCEGUERA!='',1,0))
Borrador<-Borrador %>% mutate(tipo2=if_else(ESTU_LIMITA_COGNITIVA!='',1,0))
Borrador<-Borrador %>% mutate(tipo3=if_else(ESTU_LIMITA_INVIDENTE!='',1,0))
Borrador<-Borrador %>% mutate(tipo4=if_else(ESTU_LIMITA_MOTRIZ!='',1,0))
Borrador<-Borrador %>% mutate(tipo5=if_else(ESTU_LIMITA_SORDOINTERPRETE!='',1,0))
Borrador<-Borrador %>% mutate(tipo6=if_else(ESTU_LIMITA_SORDONINTERPRETE!='',1,0))
Borrador$tipo<-Borrador$tipo1+Borrador$tipo2+Borrador$tipo3 +
Borrador$tipo4+Borrador$tipo5+Borrador$tipo6

Limitados<-Borrador[Borrador$tipo>0,]
```

Aún más resumido y preciso.

```
Limitados<-subset(SB112012,ESTU_LIMITA_SORDOCEGUERA!=''
|ESTU_LIMITA_COGNITIVA!=''|ESTU_LIMITA_INVIDENTE!=''
|ESTU_LIMITA_MOTRIZ!=''|ESTU_LIMITA_SORDOINTERPRETE!=''
|ESTU_LIMITA_SORDONINTERPRETE!='')
```

14. EVIDENCIA(S) A ENTREGAR:

🔗 Aplicar las técnicas y métodos de limpieza de datos utilizando un DATASET propuesto por el instructor ([SB11-20121-RGSTRO-CLFCCN-V1-0-txt](#)).



Se desea comprobar lo siguiente:

- Quienes se destacaron más en matemáticas, si las mujeres o los hombres con alguna discapacidad, teniendo en cuenta:
 - La ciudad
 - Edad de acuerdo al tipo de documento de identidad
 - Tipo de colegio (Oficial, Privado) y caracterización del colegio (ACADEMICO, TECNICO, etc.)
 - Qué nivel de ingles
 - Nacionalidad

🔗 Entregue los datos solicitados mediante tablas en un [modelo estrella](#) en POSTGRESQL; no olvide adjuntar el código en R y el script SQL, como también el informe de desarrollo que debe contener:

- **Encabezado:** título del informe, nombre del instructor, autor del informe (nombres y apellidos completos), nombre del programa formativo, así como la fecha de realización.
- **Introducción:** describa el tema abordado en [dimensiones y medidas](#).
- **Desarrollo:** corresponde al cuerpo del trabajo, donde se explica con detalle el desarrollo de los aspectos que se mencionan en la introducción. En este apartado deberá incluir:
 - Informe argumentado, el desarrollo del caso de estudio propuesto.
 - Pantallazos que demuestren las acciones.
 - Acta de cambios, eliminaciones o adiciones al DATASET.
 - Código en R utilizado.
- **Conclusiones:** presente las conclusiones a las que llegó luego de haber realizado el taller y el caso propuesto.

Lineamientos generales para la entrega de la evidencia:

- Productos a entregar: un documento que incluya lo solicitado para el desarrollo del caso de estudio propuesto en el taller.
- Formato: ZIP.
- Para hacer el envío de la evidencia remítase al área de la actividad correspondiente y acceda al espacio de evidencias del LMS.

Nota: Esta evidencia se debe realizar en grupo, pero cada integrante sube individualmente describiendo sus compañeros integrantes del grupo.



Servicio Nacional de Aprendizaje
Formato Taller
Centro de Gestión de Mercados, Logística y Tecnologías de la Información.

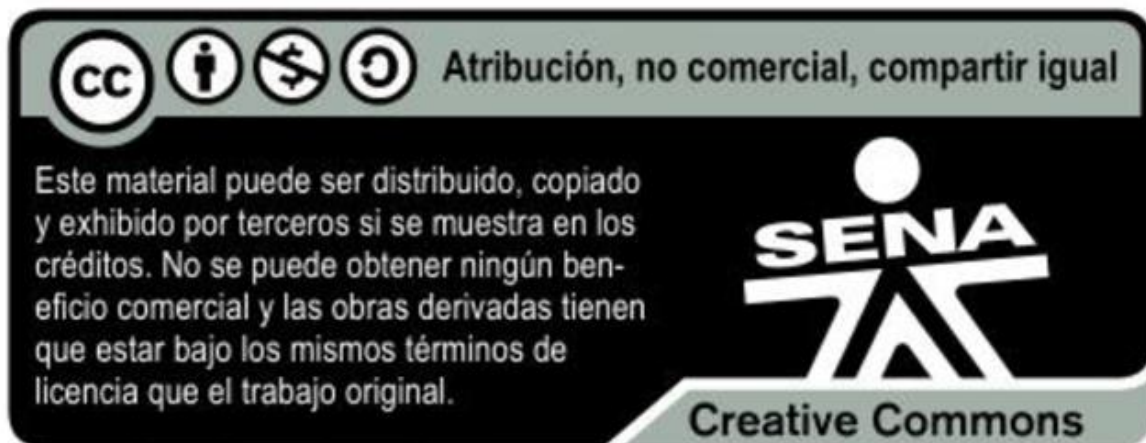
CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	José Fernando Galindo Suarez	Instructor	CGMLTI- Teleinformática	13/05/2023

CONTROL DE CAMBIOS (diligenciar únicamente si realizan ajustes al taller)

	Nombre	Cargo	Dependencia	Fecha	Razón del Cambio
Autor (es)	José Fernando Galindo Suarez	Instructor	CGMLTI Teleinformática	16/08/2023	Correcciones generales

Autor: José Fernando Galindo Suárez jgalindos@sena.edu.co 2023





Contenido

1. Situaciones para realizar limpieza en los datos:	1
2. Comandos utilizados para limpieza de datos	3
3. Comandos utilizados para limpieza de datos en Python.....	6
4. Datos erróneos e irrelevantes.....	9
5. Cambios de errores tipográficos.....	13
6. CONSTRUYENDO DIMENSIONES DEL COVID19	14
EJERCICIO EN EL AMBIENTE DE FORMACIÓN.....	18
7. ENCONTRANDO VALORES ATÍPICOS (OUTLIERS)	19
La prueba Q de Dixon	22
Hampel filter	23
Prueba de Rosner	24
8. MANIPULACION DE DATOS CON SQL DESDE R	26
9. MODELO ESTRELLA COVID19	30
10. Conectar R a POSTGRESQL.....	32
11. CONECTAR R A MONGODB.....	35
12. CONECTAR R A MYSQL	38
13. MUESTRA DE LA POBLACION CON LIMITACIONES FISICA EN SB112012.	41
14. EVIDENCIA(S) A ENTREGAR:.....	43

ⁱ <https://gltaboada.github.io/tgdbook/conexi%C3%B3n-con-bases-de-datos-desde-r.html>

ⁱⁱ <https://mappinggis.com/2020/01/como-integrar-postgresql-postgis-en-r/>