Sesión 2:

El Lenguaje R:
Funciones, Manejo de Bases de Datos, ggplot y Expresiones Regulares

FYGP
¡Ideas que crean valor!

SENA

Objetivos

- Presentar algunas técnicas de manejo de R, como la construcción de funciones, el manejo de dataframes, la construcción de gráficos y el manejo de expresiones regulares.

Palabras claves: Objeto, función, ciclo, dataframe, ggplot, gráfico, expresión regular, prueba lógica.

Introducción

Las posibilidades para trabajar con datos en R son bastante amplias. Pero como en todo lenguaje de programación es necesario manipular en algún momento los datos o construir objetos nuevos para las necesidades específicas que se tengan.

Aunque la comunidad de usuarios de R hace las veces de repositorio de códigos y consola de depuración de errores, no está de más tener una idea de cómo operar las piezas móviles del programa.

Orientación a objetos

R está orientado a objetos. En R, los objetos son estructuras que combinan datos y funciones que operan sobre ello. Por ejemplo, un **dataframe** puede ser guardado en un objeto y aplicando la función **summary()**, proporciona un resumen rápido de los estimadores estadísticos descriptivos básicos.

Mientras que la misma función operada sobre un modelo permitirá conocer los diferentes resultados, del modelo como algunas pruebas estándar realizadas por defecto.

Revisar:
https://www.datanalytics.com/libro_r/orientacion-a-objetos.html
(Bellosta, s. f.)

Funciones en R

Las funciones particulares de código son estandarizadas para que ejecuten una determinada tarea. Por lo tanto, estos códigos pueden ser descritos sobre las partes que lo conforman; por ejemplo, la figura 1 explica las tres partes de todo código que se escribe en R.

El primero es el insumo, este contiene una variable que es la transformación de una base de datos cargado. Luego, la función es el código que permite realizar una actividad de transformación específica sobre el insumo. Finalmente, está el resultado que es mostrado en una de las ventanas de la consola gráfica de RStudio.

Recuerde que, si usted definió una variable y quiere ver el resultado, debe llamar tal variable.

Figura 1. Flujo de proceso resolución CRC 5050 2016. Fuente: Elaboración propia.

Como R los identifica como objetos es necesario darles un **nombre** y cada uno de los **argumentos** o insumos de la función deben estar separados por comas.

El **cuerpo** de la función contiene todos los procedimientos, algoritmos y operaciones necesarias para obtener el **resultado.**

En la figura 2, se muestra la gramática de la una función en R.

```
mifuncion <- function(argumento1, argumento2, ...) {
cuerpo
resultado
}
```

Figura 2. Flujo de una función. Fuente: Elaboración propia.

Ejemplos de funciones

Se creará una función de tal forma que tome los valores x y y y el resultado es x^y, para cualquier x y y proporcionados; y el nombre de la funcion sera **potenciación**

```
potenciacion<-function(x,y){
  salida<-x^y
  print(salida)
}
```

Al escribir y correr este código en **Rstudio**, no sucede nada puesto que se ha almacenado en la memoria del equipo esta nueva función, la cual ya se puede usar las veces que sea necesario.

```
potenciacion(2,3)

## [1] 8
```

Como se observa, la función potenciación toma al 2 y usa como exponente a 3; dando como resultado 8.

Se creará una función que toma como insumo el valor de un producto y el porcentaje de descuento (en decimales) y como resultado muestra el valor a pagar. Esta función la llamaremos descuento y nos mostrará un resumen de los resultados

```
descuento<-function(a,b){
  if (a<0){
    print("El valor del objeto debe ser mayor a cero")
  } else{
    descuento=a*b
    pago=a-descuento
      data.frame(Descripcion=c("Valor",  "Descuento",
"Total a pagar"),Monto=c(a,descuento,pago))

  }
}
```

Como se observa, al correr la función **descuento** no solo calcula el valor a pagar, sino por medio de un objeto **dataframe** se creó un resumen de las operaciones.

```
descuento(45000, 0.25)
```

```
##     Descripcion Monto
## 1         Valor 45000
## 2     Descuento 11250
## 3 Total a pagar 33750
```

Las funciones son comúnmente utilizadas para procesos (u operaciones) que repetimos constantemente, entonces para facilitar su operación y minimizar errores humanos, es mejor programarlas.

Actividad 1: Cree una función de tal forma que el usuario ingrese su altura en metros y su peso; la función debe calcular el índice de masa muscular y en que categoría se encuentra.

Manejo de bases de datos

Los **dataframe** no siempre se encuentran en óptimas condiciones para poder trabajar con ellos; por lo tanto, realizar elementos de limpieza, depuración y en algunos casos transformación son necesarios.

Esto es el primer paso a realizar con un conjunto de datos.

Para el ejercicio usaremos la base **Mundial de variables**, la cual posee la información demográfica de los países del mundo.

https://datos.bancomundial.org/(World Bank Open Data, s. f.)

Utilice el import Dataset de un archivo .csv para hacer la carga de los datos. Vamos a extraer el registro o individuo que está en la posición 6.

```
Mundial_variables[6,]
```

```
## # A tibble: 1 x 16
##   Pais  Poblacion Densidad Urbana Religion Esperanza
Alfabetizacion Incremento
##   <chr>     <dbl>    <dbl>  <dbl> <chr>        <dbl>
<dbl>      <dbl>
## 1 Azer~      7400       86     54 Muslim          75
98        1.4
## # ... with 8 more variables: MortInfantil <dbl>, PIB
<dbl>, Natalidad <dbl>,
## #   Mortalidad <dbl>, Fertilidad <dbl>, Produccion
<dbl>, AlfaHombres <dbl>,
## #   AlfaMujeres <dbl>
```

A continuación, vamos a seleccionar los individuos que se encuentran en las posiciones 1,3,5,9.

```
Mundial_variables[c(1,3,5,9),]
```

```
## # A tibble: 4 x 16
##   Pais  Poblacion Densidad Urbana Religion Esperanza
Alfabetizacion Incremento
##   <chr>     <dbl>    <dbl>  <dbl> <chr>        <dbl>
<dbl>      <dbl>
## 1 Afgh~     20500       25     18 Muslim          44
29        2.8
## 2 Arme~      3700      126     68 Orthodox        75
98        1.4
## 3 Aust~      8000       94     58 Catholic        79
99        0.2
## 4 Barb~       256      605     45 Protstnt        78
99        0.21
## # ... with 8 more variables: MortInfantil <dbl>, PIB
<dbl>, Natalidad <dbl>,
## #   Mortalidad <dbl>, Fertilidad <dbl>, Produccion
<dbl>, AlfaHombres <dbl>,
## #   AlfaMujeres <dbl>
```

Como lo evidencian los ejemplos anteriores, la gramática para de seleccionar filas o individuos es el **nombre_datos[,fila_a_seleccionar]**. La cuales pueden ser distantes como en el ejemplo anterior o filas consecutivas como se muestra en el siguiente ejemplo.

```
muestra1=Mundial_variables[1:20,]
```

Se revisará como seleccionar variables de un dataframe de forma que se pueda trabajar con una sola variable o extraer bases en función de las columnas de la base original.

El primer ejemplo que vamos a tratar son dos formas de seleccionar la variable religión.

```
religion=Mundial_variables$Religion

religion1=Mundial_variables[,5]
```

Se ejemplariza otra forma de seleccionar bases con la función **subset**, es importante tener en cuenta que **select** debe ser un vector con las variables a tomar.

```
muestra3=subset(Mundial_variables, select=c(1:4))
```

Se mostrará ahora la forma de seleccionar bases en función de las características de las variables.

En el ejemplo a continuación, se crea una base llamada Islam la cual contiene los países musulmanes.

```
Islam=subset(Mundial_variables,
Mundial_variables$Religion=="Muslim")
```

Vamos a seleccionar ahora los países con una esperanza de vida mayor a 55

```
Mas55=subset(Mundial_variables,
Mundial_variables$Esperanza>55)
```

Estas selecciones se pueden hacer con más de una variable sin importar la naturaleza de estas. En el siguiente ejemplo se selecciona una base en función de las variables Religión y Esperanza. Para el ejemplo vamos a tomar solo la religión Muslim y la esperanza de vida mayor o igual a 75 años.

```
muestra4=subset(Mundial_variables,
                Mundial_variables$Religion=="Muslim"&
                  Mundial_variables$Esperanza>=75)
```

Unificación de bases

Un proceso común en el preprocesamiento de datos es integrar o unir bases para construir bases más robustas o conformar bases de diferentes procesos.

Para ejemplarizar esto, unificaremos las bases filtradas anteriormente Islam y muestra4. Ahora bien, las dos bases tienen las mismas columnas y están el mismo orden.

```
colnames(muestra4)
##  [1] "Pais"           "Poblacion"      "Densidad"
"Urbana"
##  [5] "Religion"       "Esperanza"
"Alfabetizacion" "Incremento"
##  [9] "MortInfantil"   "PIB"            "Natalidad"
"Mortalidad"
## [13] "Fertilidad"     "Produccion"
"AlfaHombres"    "AlfaMujeres"

colnames(Islam)
##  [1] "Pais"           "Poblacion"      "Densidad"
"Urbana"
##  [5] "Religion"       "Esperanza"
"Alfabetizacion" "Incremento"
##  [9] "MortInfantil"   "PIB"            "Natalidad"
"Mortalidad"
## [13] "Fertilidad"     "Produccion"
"AlfaHombres"    "AlfaMujeres"
```

Uniremos una debajo de la otra usando la función **rbind = unir agregando filas, cuando sabemos que los nombres de las columnas (variables), son iguales y están en el mismo orden.**

```
muestra5=rbind(muestra4, Islam)
```

Es importante recordar que en el enviroment aparece la información sobre las bases, es decir número de variables e individuos (observaciones) como la figura 3 lo demuestra.

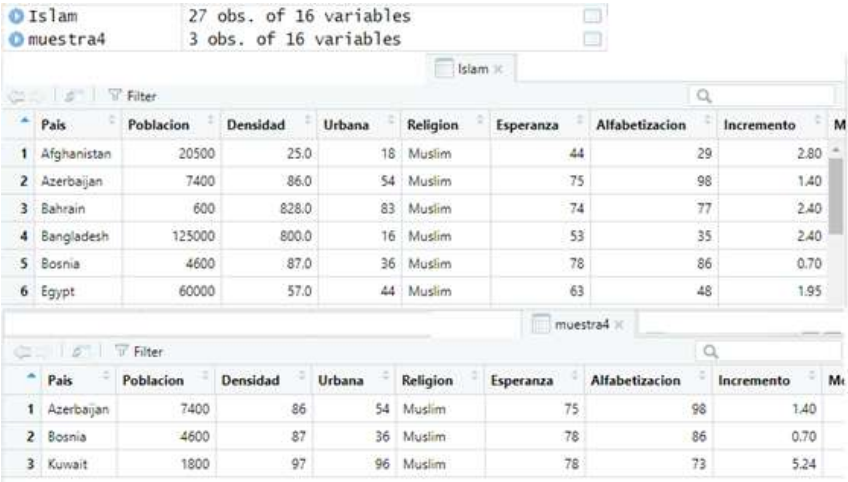

Figura 3. Vista de las fuentes a unir. Fuente: Elaboración propia.

Pero este proceso puede generar individuos repetidos, recordemos que la base muestra4 e Islam se generan por medio de un filtro de la base original, cuyo filtro tiene como parámetro la religión islámica.

En la figura 4 se puede observar el proceso de quitar elementos repetidos, para esto usaremos la función **unique.**

```
muestra6=unique(muestra5)
```

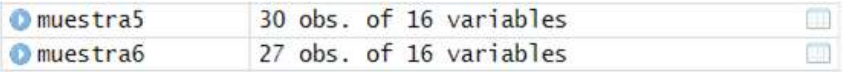

Figura 4. Eliminación de repetidos Fuente: Elaboración propia.

Reemplazar valores

Algunas veces nos encontramos que las bases de datos, que bien sea por malos procesos de alimentación de la información o por errores de digitación, existen valores en las variables erróneas.

Vamos a revisar como cambiar dichas variables sin tener que entrar a modificar la base de datos.

```
Mundial_variables[2,3]

## # A tibble: 1 x 1
##   Densidad
##      <dbl>
## 1       12
```

Como observamos, el valor (el código) toma la variable tres y el segundo registro; por lo tanto vemos que la variable **Dendidas** en el segundo registro tiene el valor de 12.

Vamos remplazar este valor por el valor de 25

```
Mundial_variables[2,3]=25

Mundial_variables[2,3]

## # A tibble: 1 x 1
##   Densidad
##      <dbl>
## 1       25
```

Como se observa el valor fue cambiado de 12 a 25. Veamos otra forma de hacerlo.

Para este caso, vamos a cambiar el quinto registro que tiene la variable **Urbana**, la cual le vamos a dar el valor de 50.

```
Mundial_variables$Urbana[5]

## [1] 58

Mundial_variables$Urbana[5]=50
```

También podemos cambiar el tipo de toda una variable con los comandos que se lista a continuación representado en la figura 5.

Se debe tener en cuenta que si aplico el comando as.factor a una variable de naturaleza numérica como edad, pero y altura, transforma totalmente su naturaleza y no se podrán establecer indicadores estadísticos como promedio, varianza que son exclusivos de variables cuantitativas.

Función	Tipo al que hace coerción
`as.integer()`	Entero
`as.numeric()`	Numerico
`as.character()`	Cadena de texto
`as.factor()`	Factor
`as.logical()`	Lógico
`as.null()`	NULL

Figura 5. Comandos para cambiar tipo Fuente: Elaboración propia.

Actividad 2: Busque en la página de datos abiertos de su ciudad, una base de interés tanto con variables cuantitativas como cualitativas y genere las siguientes sub bases

- Una base llamada cuanti que va contener todas las variables cuantitativas de la base original.
- Una base llamada cuali que va contener todas las variables cualitativas de la base original.
- Seleccione una de las variables cuantitativas y filtre por los valores mayores el promedio de esa variable.
- Seleccione una de las variables cuantitativas y filtre por los valores menores o iguales el tercer cuartil de esa variable.

Uso de librerías

La instalación base de R y Rstudio proporciona una gran serie de herramientas para la analítica de datos como para la estadística, sin embargo, la enorme potencia de R deriva de su capacidad de incorporar en cualquier momento nuevas funciones capaces de realizar nuevas tareas.

Un paquete (package) es una colección de funciones, datos y código R que se almacenan en una carpeta conforme a una estructura bien definida, fácilmente accesible para R.

Existen muchísimos paquetes (más de 12000), cada uno especializado en funciones específicas.

Cuando se ejecuta el comando **library()** en la consola, nos aparece una pestaña con la lista de los paquetes instalados. Esta lista también se pude ver en Rstudio en la pestaña **Packages.**

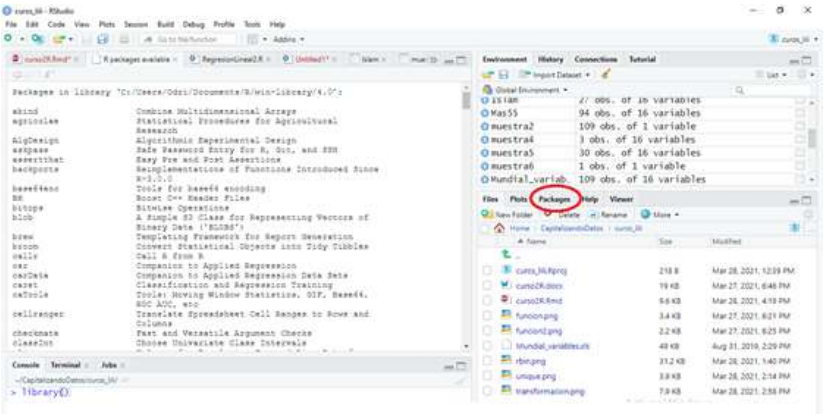

Figura 6. Paquetes instalados. Fuente: Elaboración propia.

Por lo tanto, es importante tener instalados los paquetes que se quieran usar y de igual manera llamarlos a memoria cuando se utilicen.

En Rstudio existen dos formas de instalar paquetes por medio del comando **install.packages("nombre_paquete")** o en la pestaña Packages en el icono Install como lo muestra la figura 7.

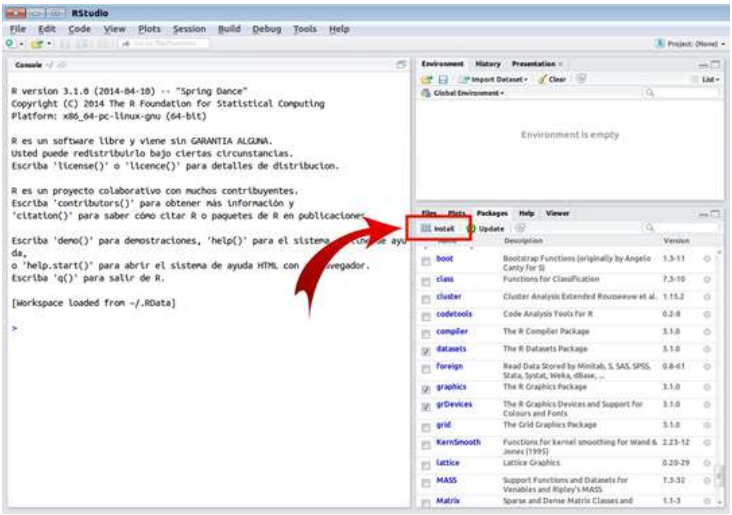

Figura 7. Formas de instalar paquetes. Fuente: Elaboración propia.

En ambos casos, R se conecta a alguno de los repositorios de CRAN (Comprehensive R Archive Network) (The Comprehensive R Archive Network, s. f.) en internet, descarga el archivo que contiene el paquete, lo descomprime y lo instala en nuestro directorio de paquetes por defecto.

Una vez instalado el paquete (lo cual se debe hacer una sola vez), es necesario cargar dicho paquete para poder utilizar dichas funciones; para esto es necesario utilizar al comienzo del scritp el comando l**ibrary("nombre_paquete").**

Si un paquete ha sido ya cargado con anterioridad, volver a ejecutar **library(nombre_paquete)** no tiene ningún efecto.

Actividad 3: Instale el paquete para graficas especializadas ggplot2 y cárguelo a la memoria.

Gráficos avanzados

Las gráficas de datos permiten de forma rápida y visual obtener información de los datos, esto debe ser el pilar fundamental para realizar gráficas puesto que una gráfica mal hecha o que no aporte información no sirve; por el contrario, puede generar supuestos equivocados sobre la información que contiene los datos.

La instalación base de R y Rstudio, permite hace gráficos básicos, pero existen paquetes especializados para realizar gráficos más complejos. Sin importar si se usa las gráficas que están por defecto en R o algún otro paquete es importante recordar que la graficas dependen del tipo de variable que se posee.

Por ejemplo, la función hist permite hacer histogramas de variables cuantitativas como lo muestra la figura 8; donde el parámetro breaks hace referencia a la cantidad de intervalos que se utilizan para construir el histograma.

```
hist(Mundial_variables$Alfabetizacion,breaks =5)
```

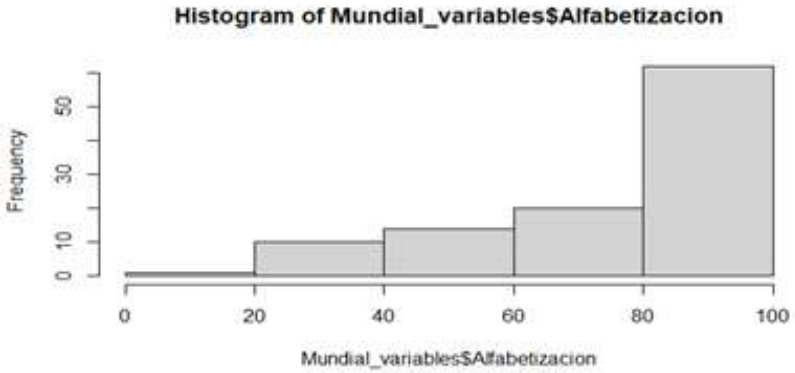

Figura 8. Histograma. Fuente: Elaboración propia.

El comando **boxplot** nos permite hacer una gráfica de caja y bigotes para mostrar la distribución de los datos respecto a la media, observe la figura 9.

```
boxplot(Mundial_variables$Alfabetizacion)
```

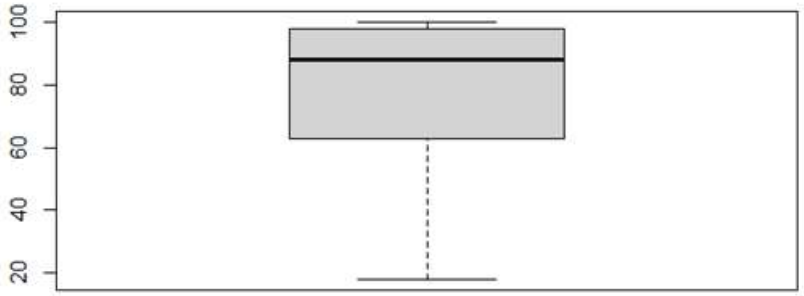

Figura 9. Boxplot o caja y bigotes. Fuente: Elaboración propia.

El comando plot realiza una gráfica de puntos donde el eje x corresponde al índice o ubicación del registro en la variable y el eje y al valor de la variable.

Claramente, la gráfica 10 no aporta mucha información sobre los datos.

```
plot(Mundial_variables$Alfabetizacion)
```

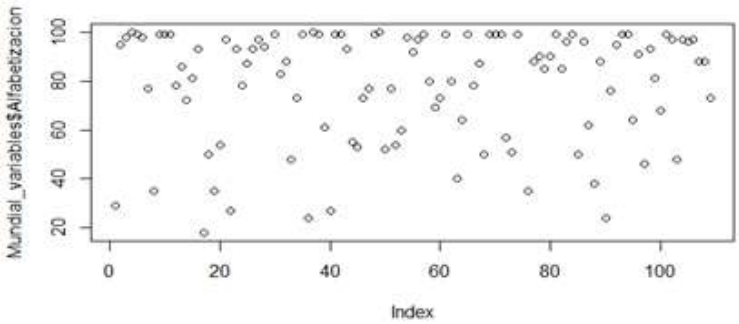

Figura 10. Gráfico de Dispersión. Fuente: Elaboración propia.

Gráficos avanzados

Para realizar gráficos más elaborados vamos a usar el paquete **ggplot**, por lo tanto, cárguelo en memoria.

El paquete **ggplot** realiza gráficos por capas entendiendo estas como las capas graficas que se utilizan en diseño gráfico, cada una de estas contiene información o parámetros de la misma.

Se inicia ejemplarizando el uso de paquete mostrado en la figura 11 con la función más básica a la cual se le añadirán capas.

```
grafica=ggplot(Mundial_variables, aes(x=Religion))
grafica
```

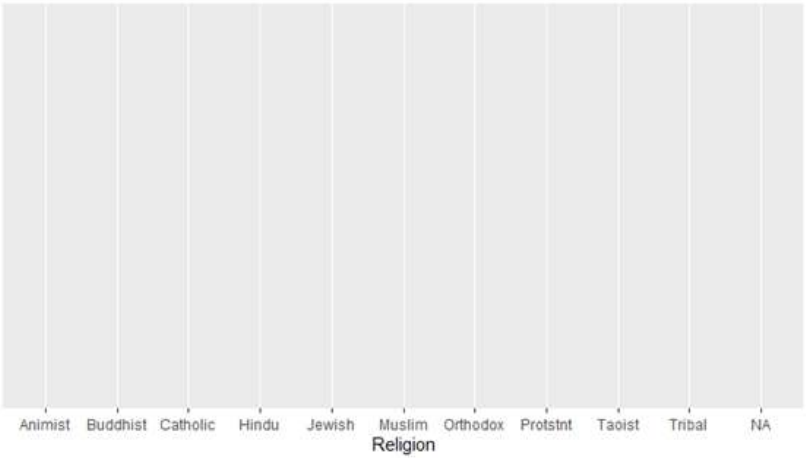

Figura 11. Ggplot. Fuente: Elaboración propia.

Como se puede apreciar en la parte inferior se activa una gráfica básica en la que el eje x contiene los nombres cada una de las categorías existentes en la variable **Religión**.

Esto está bien puesto debido a que solo le estamos informando a R que por medio del paquete **ggplot** se construirá una gráfica y el eje x será la variable **Religión**.

Vamos agregar la siguiente variable, la cual tendrá el tipo de grafica a usar.

```
grafica=grafica+geom_bar()
```

Como se observa al objeto grafica le sumamos la forma de la gráfica y sobrescribe el objeto gráfico.

Como se observa es una gráfica aceptable representado en la gráfica 12, pero si se quisiera mejorarla y personalizarla, es posible usando unas capas más.

Incluir el nombre de los ejes y título.

```
grafica=grafica+xlab("Religion del país")
+ylab("Número de paises")+ggtitle("Gráfico de
barras") grafica
```

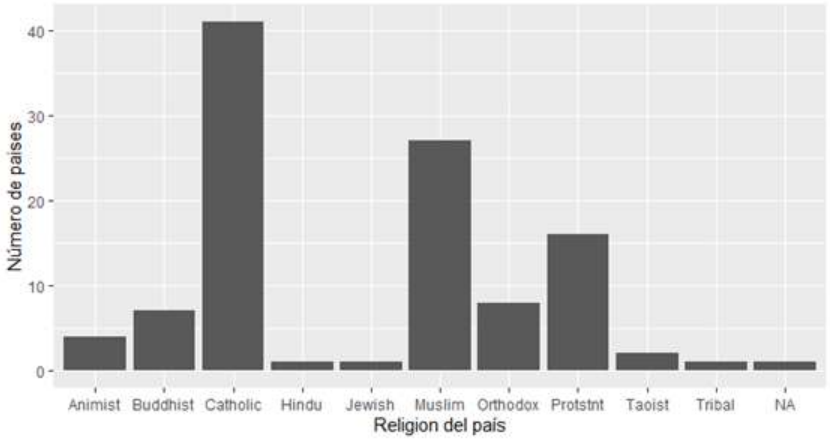

Figura 12. Ggplot para gráficos de barras. Fuente: Elaboración propia.

Incluyendo color mostrado en la gráfica 13.

```
grafica=ggplot(Mundial_variables, aes(x=Religion))+
  geom_bar(width=0.8, colour="red", fill="blue")
grafica
```

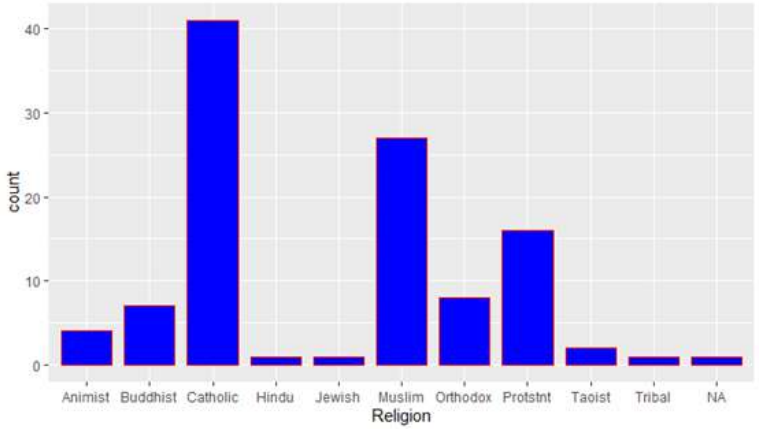

Figura 13. Ggplot para gráficos de barras con color. Fuente: Elaboración propia.

Otra opción agregando los títulos a los ejes y el título de la gráfica, observe la figura 14.

```
grafica+xlab("Religion del país")+ylab("Número de
paises")+ggtitle("Gráfico de barras por paises
Gráfico de barras \n por paises Gráfico de barras por
paises")
```

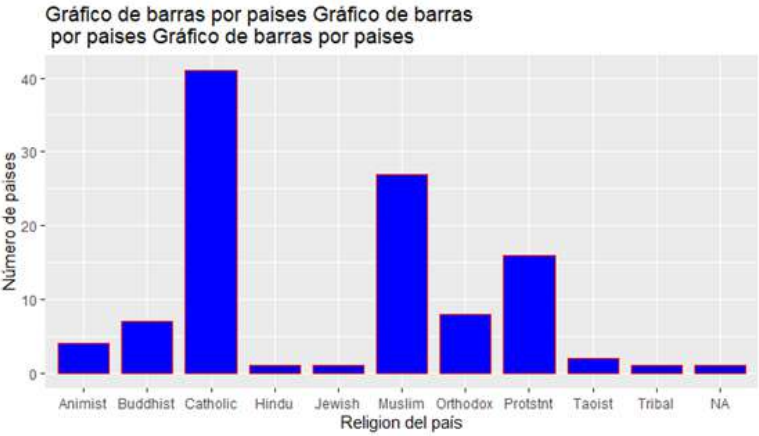

Figura 14. Ggplot para gráficos de barras con color. Fuente: Elaboración propia.

En las gráficas se puede observar que existe la categoría NA, esto puede ser algo que se desee evitar, si ese es el caso vamos a ejemplarizar como quitar los NA, presentado en la figura 15.

```
grafica=ggplot(Mundial_variables[is.na(Mundial_variab
les$Religion)==F,],aes(x=Religion))+geom_bar(width=0.
8, colour="red", fill="blue")

grafica+xlab("Religion del país")+ylab("Número de
paises")+ggtitle("Gráfico de barras")
```

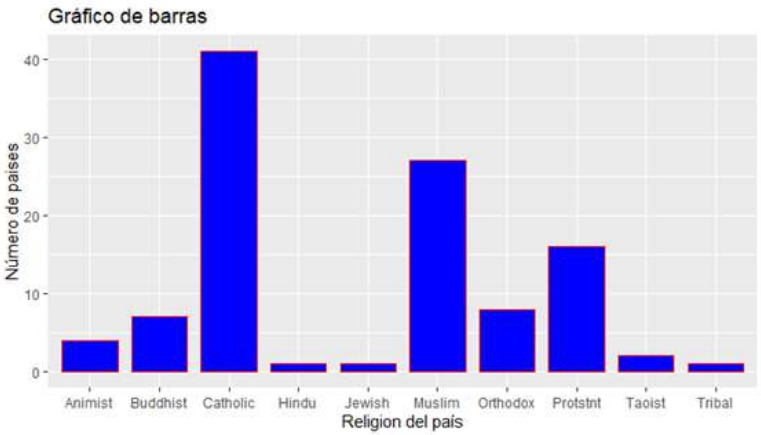

Figura 15. Ggplot para gráficos de barras con color sin NAs. Fuente: Elaboración propia.

Se ejemplariza a continuación como hacer un histograma mostrado en la figura 16.

```
grafica=ggplot(Mundial_variables,
aes(x=Alfabetizacion))
grafica+geom_histogram()

## `stat_bin()` using `bins = 30`. Pick better value
with `binwidth`.
```

Vamos hacer un histograma dividiendo en grupos preestablecidos (cada 10%) mostrado en la figura 17.

```
grafica+geom_histogram(binwidth = 10, fill="red",
colour="black")
```

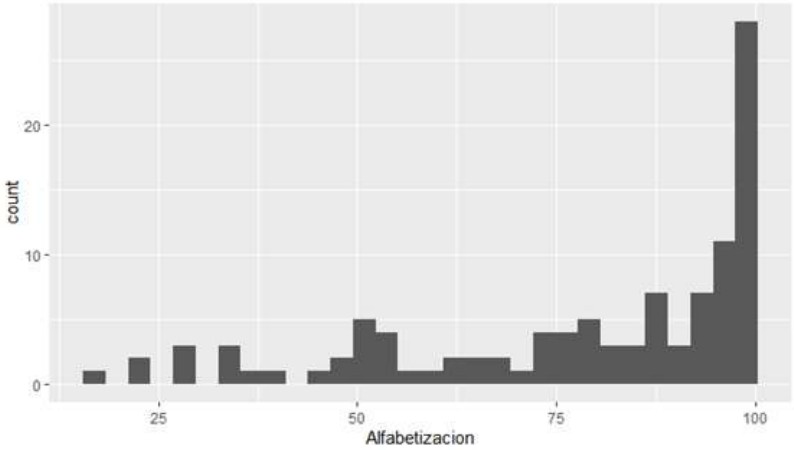

Figura 16. Histograma. Fuente: Elaboración propia.

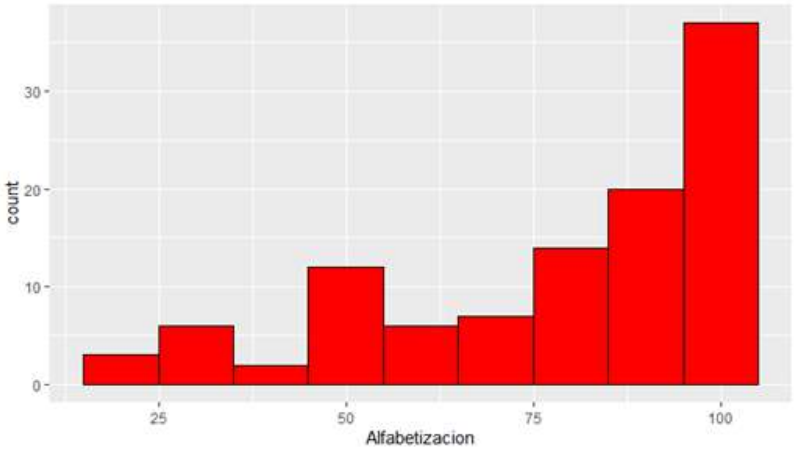

Figura 17. Histograma. Fuente: Elaboración propia.

Para la variable sexo (sex) vamos a realizar un gráfico de barras (ver figura 18) que nos muestre cuantos registros corresponden a hombres y cuantos a mujeres y le daremos color.

```
grafica=ggplot(Hepatitis, aes(x=Sexo))

grafica+geom_bar()
```

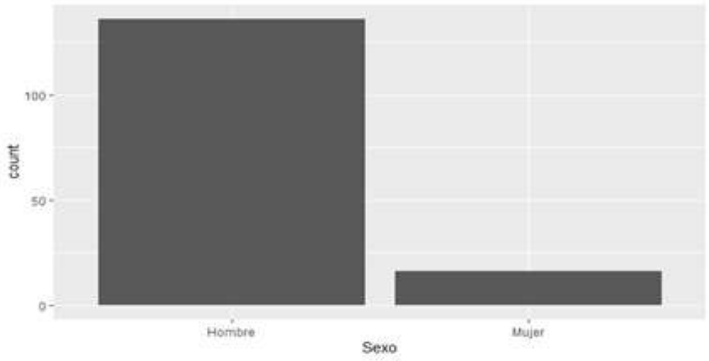

Figura 18. Histograma para Sexo. Fuente: Elaboración propia.

Al igual que en otras ocasiones, podemos agregar color al gráfico como lo demuestra la figura 19.

```
grafica+geom_bar(fill="firebrick",
colour="hotpink2")+ ylab("Conteo")
```

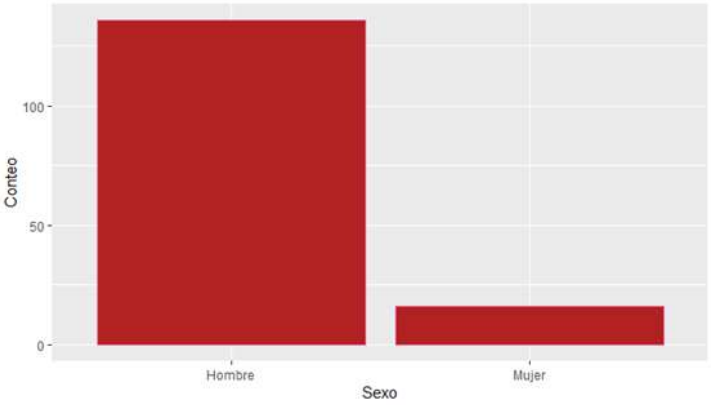

Figura 19. Histograma para Sexo con color. Fuente: Elaboración propia.

Para ver más posibilidades del parámetro colour revise : http://sape.inf.usi.ch/quick-reference/ggplot2/colour(SAPE - Software and Programmer Efficiency Research Group, s. f.)

Como se observa la figura anterior contiene los conteos para categoría, pero ahora ejemplificaremos como hacerlo por porcentaje (ver figura 20).

```
grafica+geom_bar(aes(y=100*
(..count..)/sum(..count..)),
fill="firebrick", colour="hotpink2")
```

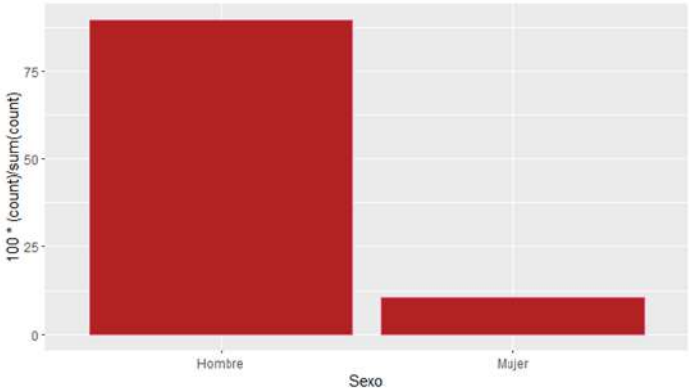

Figura 20. Histograma para Sexo con color (%). Fuente: Elaboración propia

Comparemos dos o más variables cualitativas, para esto realicemos la figura 21 donde usaremos la variable sexo y la variable esteroides

```
grafica=ggplot(Hepatitis,aes(x=Sexo,
fill=Esteroides))

grafica+geom_bar()
```

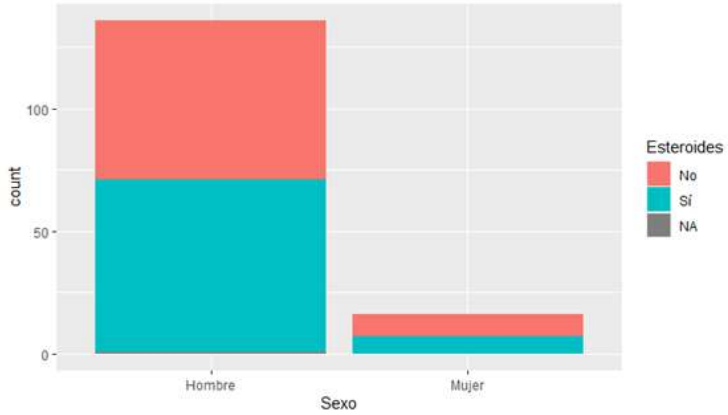

Figura 21. Histograma del Sexo bajo el tipo de esteroides utilizado. Fuente: Elaboración propia

Cambiemos la ubicación para que su lectura sea más sencilla como lo demuestra la figura 22.

```
grafica=ggplot(Hepatitis,aes(x=Sexo,
fill=Esteroides))
grafica+geom_bar()

grafica+geom_bar(position = "dodge")
```

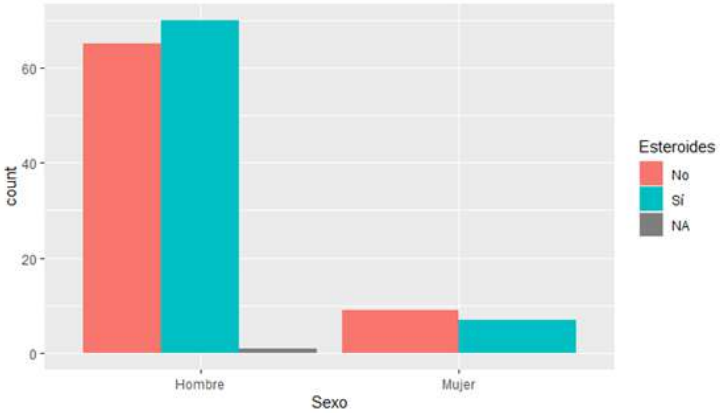

Figura 22. Histograma del Sexo bajo el tipo de esteroides utilizado. Fuente: Elaboración propia

Ejemplarizaremos más graficas usando la base **Hepatitis**, la cual tiene información sobre pacientes con esta enfermedad, por favor cárguela.

Como se observa, aparece de nuevo la categoría **NA** vamos a ejemplarizar como quitarla (ver figura 23).

```
grafica=ggplot(Hepatitis[is.na(Hepatitis$Esteroides)=
=F,],aes(x=Sexo))
grafica+geom_bar()
```

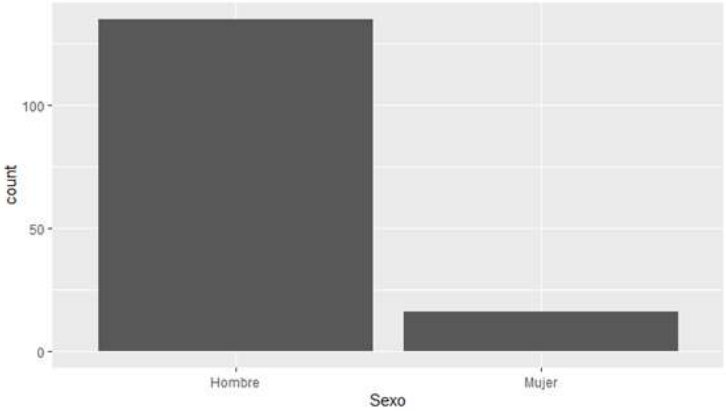

Figura 23. Histograma del Sexo con la base de datos Hepatitis. Fuente: Elaboración propia

Vamos a ejemplarizar el usar visualizar más de tres variables categóricas, tenga en cuenta que, si las variables a usar tienen muchas categorías, se dificultara la lectura de la gráfica, por esto ese usa sobre todo con variables binomiales presentado en la figura 24.

```
grafica=ggplot(Hepatitis,aes(x=Sexo,
fill=Esteroides))

grafica+geom_bar()

grafica+geom_bar(position = "dodge")

grafica+geom_bar(position = "dodge")+ylab(NULL)

grafica+geom_bar(position = "dodge")+facet_grid(Clase
~.)
```

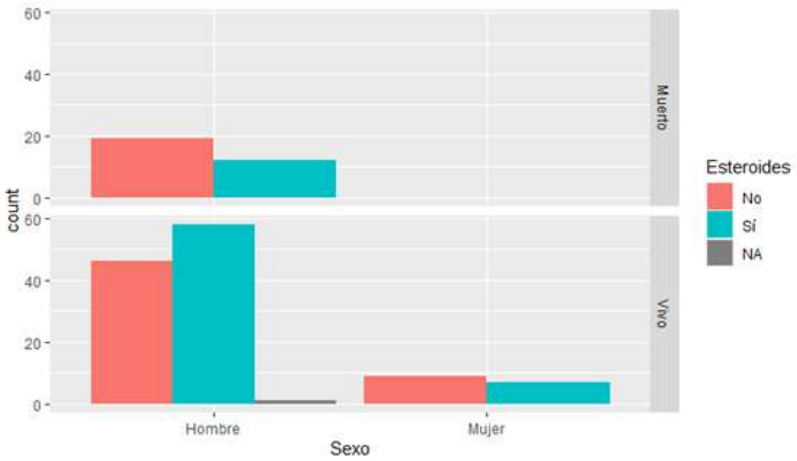

Figura 24. Histograma del Sexo con la base de datos Hepatitis. Fuente: Elaboración propia

Cambiemos la orientación como lo muestra la figura 25.

```
grafica=ggplot(Hepatitis,aes(x=Sexo,
fill=Esteroides))

grafica+geom_bar()

grafica+geom_bar(position = "dodge")

grafica+geom_bar(position = "dodge")+ylab(NULL)
```

```
grafica+geom_bar(position =
"dodge")+facet_grid(.~Clase)
```

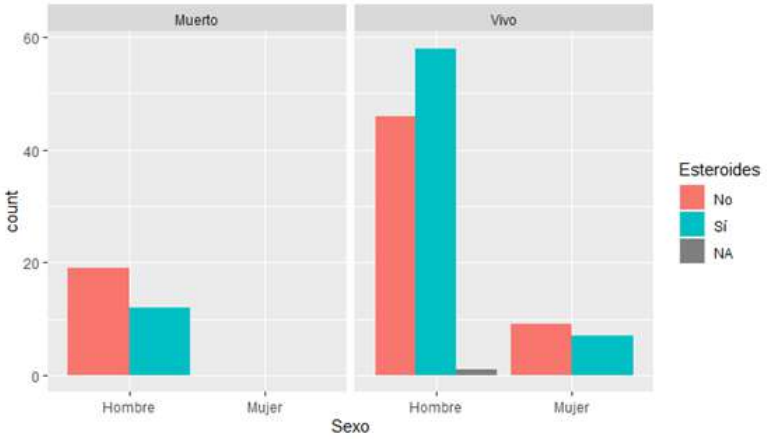

Figura 25. Histograma del Sexo con la base de datos Hepatitis rotada. Fuente: Elaboración propia

Vamos a ejemplarizar la visualización de una variable cualitativa y una cuantitativa, utilizando histogramas (ver figura 26).

```
grafica=ggplot(Hepatitis, aes(x=SGOT, fill=Sexo))
grafica+geom_histogram(binwidth = 20)
```

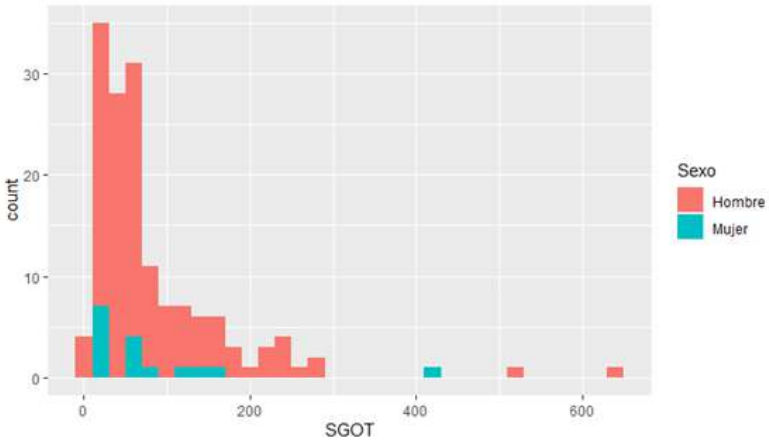

Figura 26. Histograma con una variable cualitativa y cuantitativa. Fuente: Elaboración propia

Otra forma de visualizar esta información sin tanto pixel es por medio de los gráficos de densidad como lo muestra la figura 27.

```
grafica=ggplot(Hepatitis, aes(x=SGOT, fill=Sexo))
grafica+geom_density(alpha=0.4)
```

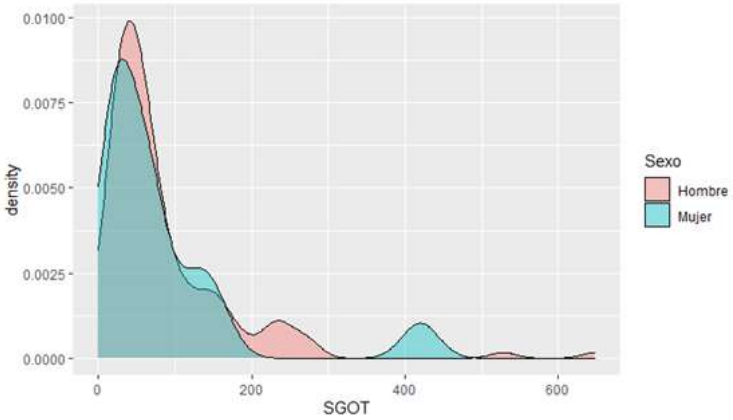

Figura 27. Gráfico de densidad. Fuente: Elaboración propia

Boxplot con ggplot

```
grafica=ggplot(Hepatitis, aes(x=Sexo, y=Albumina))
grafica+geom_boxplot()
```

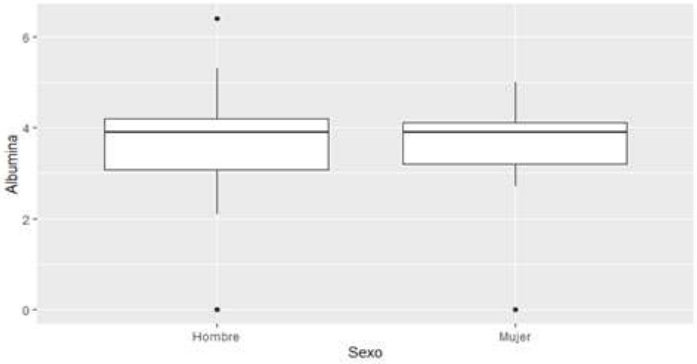

Figura 28. Boxplot. Fuente: Elaboración propia

```
grafica1=ggplot(Hepatitis, aes(y=Albumina,
fill=Sexo))
grafica1+geom_boxplot()+facet_grid(.~Clase)
```

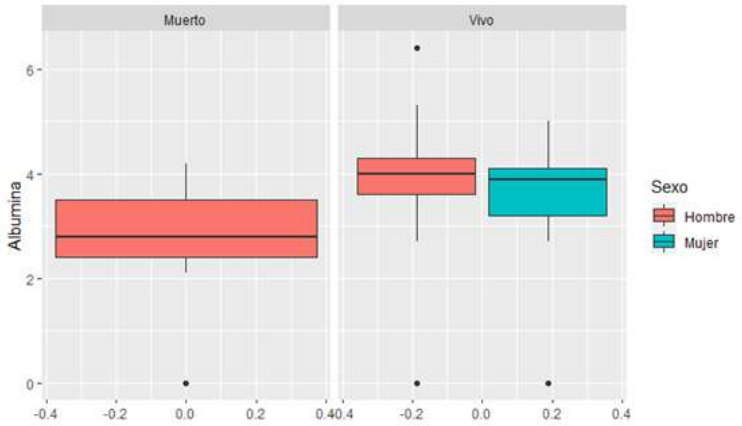

Figura 29. Boxplot por variables filtardas. Fuente: Elaboración propia

Gráfica de puntos

Revisemos como hacer graficas de puntos (ver figura 30); en el primer ejemplo vamos a relacionar dos variables cuantitativas de tal forma que las ubica en el plano como parejas ordenadas.

```
grafica=ggplot(Hepatitis, aes(x=SGOT, y=Albumina))
grafica+geom_point()
```

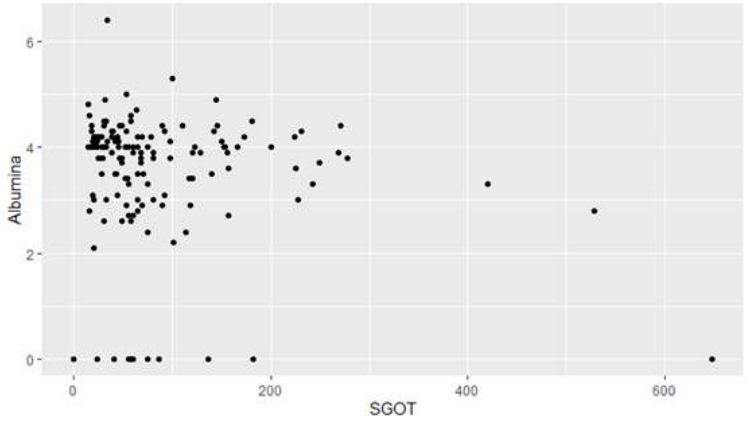

Figura 30. Gráfico de punto o dispersión. Fuente: Elaboración propia

Para los demás ejemplos de grafica de puntos se utiliza como base la gráfica anterior y se le agregan capas; en este caso le pedimos que nos separe por la variable sexo como lo muestra la figura 31.

```
grafica+geom_point()+facet_grid(.~Sexo)
```

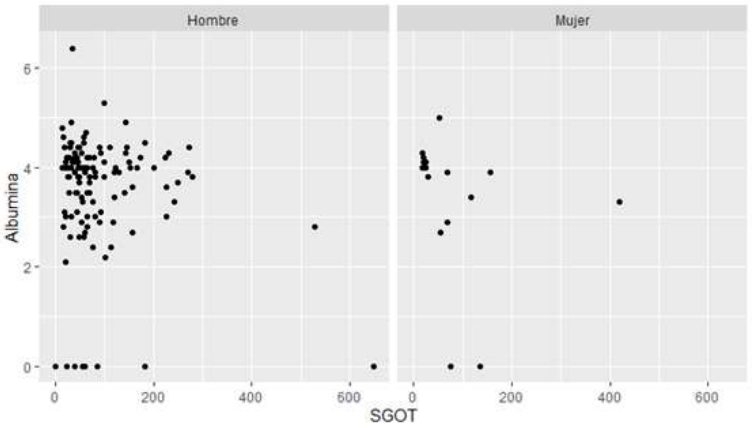

Figura 31. Gráfico de punto o dispersión por sexo. Fuente: Elaboración propia

En este caso se deja la base inicial, pero se le pide que sea asignado un color a cada categoría de la variable sexo (ver figura 32).

```
grafica+geom_point(aes(colour=Sexo))
```

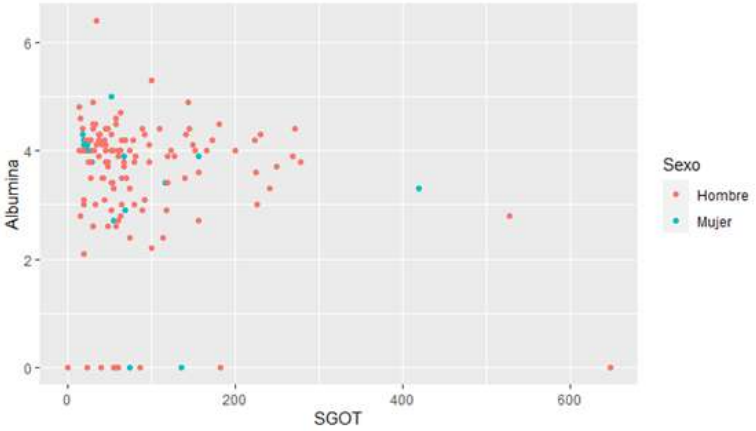

Figura 32. Gráfico de punto o dispersión por sexo con color. Fuente: Elaboración propia

Para los demás ejemplos de grafica de puntos se utiliza como base la gráfica anterior y se le agregan capas; en este caso le pedimos que nos separe por la variable sexo como lo muestra la figura 31.

```
grafica+geom_point(aes(colour=Protina))
```

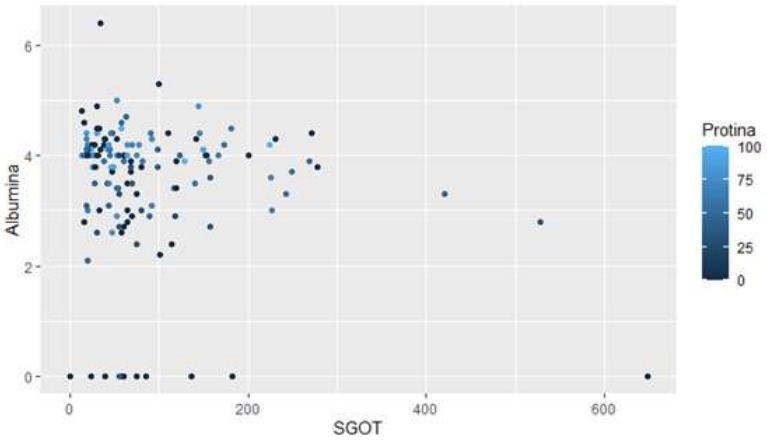

Figura 33. Gráfico de punto o dispersión por sexo con color de degrade. Fuente: Elaboración propia

Combinando algunos ejemplos anteriores (ver figura 34).

```
grafica+geom_point(aes(colour=Protina))+
facet_grid(.~Sexo)
```

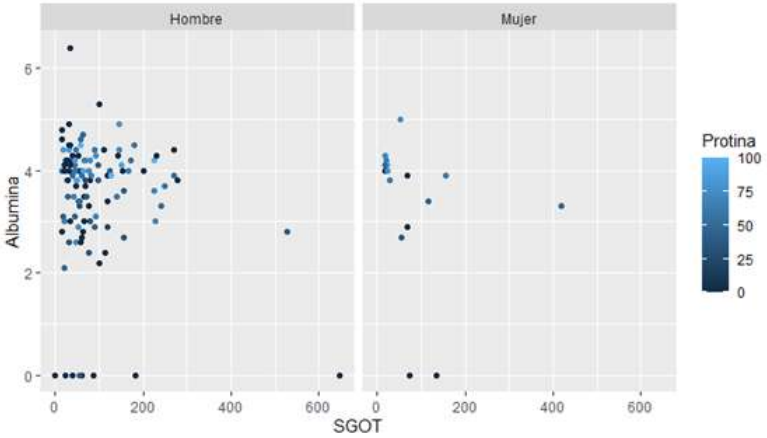

Figura 34. Gráfico de punto con color de degrade por sexo y condición de vida. Fuente: Elaboración propia

Actividad 4: Para la base de datos que ha trabajado durante toda la cartilla haga por lo menos 4 graficas que muestren la información que aportan los datos.

Expresiones regulares

Las expresiones regulares son notación de patrones de texto en contraposición a cadenas de caracteres exactas. Permiten hacer búsquedas básicas e identificar patrones. La figura 35 muestra las expresiones regulares más utilizadas

.	Carácter comodin: cualquier carácter
*	Repetir: cero o más ocurrencias de carácter o clase anteriores
^	Posición en la línea: comienzo de la línea
$	Posición en la línea: fin de la línea
[class]	Clase de carácter: cualquier carácter de la serie
[^class]	Clase inversa: cualquier carácter que no esté en la serie
[x-y]	Intervalo: cualquier carácter que esté dentro del intervalo especificado
\x	Escape: uso literal de un metacarácter
\<xyz	Posición de palabra: principio de la palabra
xyz\>	Posición de palabra: fin de la palabra

Figura 35. Expresiones regulares. Fuente: Elaboración propia.

Actividad 5: Por favor instale los paquetes gsubfn, tidyverse, datos
El uso de la función **grep** permite el uso de expresiones regulares sobre más de un carácter, brinda una forma más flexible.

- .coincide con cualquier carácter individual, como se muestra en el primer ejemplo.
- [^ …]: una lista de caracteres invertida, similar a …, pero coincide con cualquier carácter excepto los que están dentro de los corchetes.
- : suprime el significado especial de los meta caracteres en la expresión regular, es decir,
- |: un operador "o", coincide con los patrones a ambos lados del |.
- (…): agrupación en expresiones regulares. Esto le permite recuperar los bits que coinciden con varias partes de su expresión regular para que pueda alterarlos o usarlos para crear

una nueva cadena. Cada grupo puede entonces ser referido usando \ N, siendo N el No. de (…) usado. Esto se llama referencia inversa.

```
strings <- c("^ab", "ab", "abc", "abd", "abe", "ab 12", "abf")
strings
```

```
## [1] "^ab"   "ab"    "abc"   "abd"   "abe"   "ab 12" "abf"
```

```
grep("ab.", strings, value = TRUE)
```

```
## [1] "abc"   "abd"   "abe"   "ab 12" "abf"
```

```
grep("ab[c-e]", strings, value = TRUE)
```

```
## [1] "abc" "abd" "abe"
```

```
grep("ab[^c]", strings, value = TRUE)
```

```
## [1] "abd"   "abe"   "ab 12" "abf"
```

```
grep("^ab", strings, value = TRUE)
```

```
## [1] "ab"    "abc"   "abd"   "abe"   "ab 12" "abf"
```

```
grep("\\^ab", strings, value = TRUE)
```

```
## [1] "^ab"
```

```
grep("abc|abd", strings, value = TRUE)
```

```
## [1] "abc" "abd"
```

```
gsub("(ab) 12", "\\1 34", strings)
```

```
## [1] "^ab"   "ab"    "abc"   "abd"   "abe"   "ab 34" "abf"
```

Referencias

Bellosta, C. J. G. (s. f.). R para profesionales de los datos: Una introducción. Recuperado 28 de marzo de 2021, de https://www.datanalytics.com/libro_r/orientacion-a-objetos.html

R.(s. f.). The Comprehensive R Archive Network. Recuperado 28 de marzo de 2021, de https://cran.r-project.org/

RStudio. (s. f.). Download the RStudio IDE. Recuperado 27 de marzo de 2021, de https://rstudio.com/products/rstudio/download/

R4DS (s.f.). Welcome | R for Data Science. Recuperado de https://r4ds.had.co.nz/.

SAPE- Software and Programmer Efficiency Research Group. (s. f.). ggplot2 Quick Reference: Colour (and fill). Recuperado 28 de marzo de 2021, de http://sape.inf.usi.ch/quick-reference/ggplot2/colour

World Bank Open Data (s. f.). Data. Recuperado 28 de marzo de 2021, de https://datos.bancomundial.org/

Wickham et al (s.f.). ggplot2:Create Elegant Data Visualisations Using the Grammar of Graphics. Recuperado de https://ggplot2.tidyverse.org/.


FYGP
¡Ideas que crean valor!
SENA