Ejercicios de repaso módulo 2

Rstudio

Servicios Integrales de Recursos Biológicos, Acuáticos y Ambientales

Autores: L. Daniel Carrillo-Colín; Oscar G. Zamora-García y J. Fernando Márquez-Farías

Introducción

Esta guía de ejercicios ha sido diseñada para repasar de los conceptos aprendidos durante la sesión 2 del curso de *R esencial*. Encontrarás una variedad de ejercicios que

abarcan desde la exploración inicial de objetos hasta la creación de graficas en R básico. Cada ejercicio está diseñado para recordarte los fundamentos esenciales y permitirte practicar tus habilidades de programación en R.

Instrucciones

A continuación, te mostraremos una serie de resultados obtenidos a través de líneas de código de R. Es necesario que mandes al correo: cursos.sirbaa@gmail.com, el script que produce los mismos resultados aqui mostrados.

Guarda el archivo con tu primer apellido, separado por un guion bajo y tu nombre, seguido de la palabra modulo2. Ejemplo: "**Carrillo_Daniel_modulo1.R"**

1. Información inicial del código

Comienza siempre el script con la información inicial pertinente en forma de comentario

```
#Autor: Luis Daniel Carrillo Colín
#Fecha: 12/02/2024
#Módulo 1: Primeros pasos en Rstudio
```

2. Exploración inicial de objetos

Se refiere a cómo R almacena los datos en la memoria. Proporciona detalles técnicos sobre cómo se guarda internamente el objeto. Puede devolver tipos como "numeric" (números), "character" (texto), "list" (listas), "function" (funciones), entre otros. Es útil para conocer el tipo de datos dentro de un objeto, pero se centra en la implementación interna.

Tipo de objeto

Crea varios objetos:

1) Uno que sea un número entero con tres dígitos.
2) Uno que sea un vector numérico.
3) Crea una función simple.
4) Un vector de tipo factor.
5) Crea una matriz de 2 filas y tres columnas.

```
## [1] 158

## [1]  0.5  1.7  3.5  4.8  8.9  7.5 15.5
```

```
## function(x) {
##   return(x +54^2)
## }
```

```
mi_factor
```

```
[1] blanco rojo   azul   azul   rojo   rojo   blanco
## Levels: azul blanco rojo
```

```
mi_matriz
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12
```

2.1. Ahora, obtén el tipo de cada objeto

```
## [1] "double"
```

```
## [1] "double"
```

```
## [1] "closure"
```

```
## [1] "integer"
```

```
## [1] "integer"
```

3. Modo

Recuerda que el "modo" muestra cómo se almacenan los datos más pequeños dentro del objeto, y nos da información sobre si son números, texto o valores lógicos. No es muy utilizado para explorar objetos en R porque se enfoca en detalles técnicos sobre cómo se almacenan los datos a nivel interno. Por lo general, typeof y class proporcionan información más útil para entender objetos en R.

3.1 Obtén el modo de los objetos creados anteriormente.

```
## [1] "numeric"
```

```
## [1] "numeric"
```

```
## [1] "function"
```

```
## [1] "numeric"
```

```
## [1] "numeric"
```

4. Clase

Recuerda que la clase de un objeto se refiere a la categoría del objeto en el mundo de R. Proporciona una descripción general de lo que es el objeto en el contexto de R. Puede devolver categorías como "data.frame" (marco de datos), "factor" (factor), "matrix" (matriz), Date" (fecha), entre otros. Es útil para entender cómo R interpreta y maneja el objeto en función de su categoría.

4.1 Obtén la clase de los objetos creados anteriormente.

```
## [1] "numeric"

## [1] "numeric"

## [1] "function"

## [1] "factor"

## [1] "matrix" "array"
```

5. Estructura

La estructura nos muestra en detalle cómo está hecho el objeto, incluyendo los tipos de datos que contiene, cuántos elementos tiene, los nombres de las variables y otras características.

Es útil cuando necesitas explorar minuciosamente la estructura de un objeto y comprender su contenido. Te proporciona una vista detallada del objeto, como un mapa para explorarlo.

Es menos comúnmente utilizado y se enfoca en cómo se almacenan individualmente los datos dentro del objeto (por ejemplo, si son números, texto, lógica, etc.). Puede ser útil en situaciones técnicas específicas, pero generalmente typeof y class son más informativos en la mayoría de los casos.

5.1 Obtén la estructura de los objetos creados anteriormente

```
##  num 158

##  num [1:7] 0.5 1.7 3.5 4.8 8.9 7.5 15.5

## function (x)
##  - attr(*, "srcref")= 'srcref' int [1:8] 13 15 15 1 15 1 13 15
##   ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment
: 0x0000026de7016f18>

##  Factor w/ 3 levels "azul","blanco",..: 2 3 1 1 3 3 2

##  int [1:2, 1:3] 1 2 3 4 5 6
```

6. Operaciones con vectores

Hay que recordar que los vectores se crean con la función *c(),* que significa 'concatenar' o 'combinar'. Haremos algunas operaciones con vectores.

6.1 Crearemos un vector llamado vector1

```
vector1 <- c(3.5,4.7,7.5,8.6,1.2,0.7,2.3,6.8,9.4)
```

6.2 Al objeto "vector 1", multiplícalo por 8 y súmale 252.

```
## [1] 280.0 289.6 312.0 320.8 261.6 257.6 270.4 306.4 327.2
```

6.3 Calcula la raíz cuadrada

```
## [1] 1.870829 2.167948 2.738613 2.932576 1.095445 0.836660 1.516575 2.6
07681
## [9] 3.065942
```

6.4 Del objeto "vector1", calcula el promedio y la desviación estándar

```
## [1] 4.966667

## [1] 3.247307
```

6.5 Filtra los números mayores a 5 del objeto "vector1

```
## [1] 7.5 8.6 6.8 9.4
```

7. Exploración del data frame *mtcars*

7.1 Explora las primeras 6 líneas del data frame mtcars

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

7.2 Obtén el resumen estadístico de las columnas numéricas de “mtcars”

```
##       mpg             cyl             disp             hp       
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0  
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
##       drat             wt             qsec             vs        
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
##        am              gear            carb      
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
##  Median :0.0000   Median :4.000   Median :2.000  
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812  
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

7.3 Calcula el promedio de la variable “mtp”.

Nota: recuerda que para acceder a una columna en particular de un data frame, usa la siguiente sintaxis: mtcars$mtp

```
## [1] 20.09062
```

7.4 Muestra las ultimas líneas del data frame mtcars

```
##                  mpg cyl  disp  hp drat    wt qsec vs am gear carb
## Porsche 914-2   26.0   4 120.3  91 4.43 2.140 16.7  0  1    5    2
## Lotus Europa    30.4   4  95.1 113 3.77 1.513 16.9  1  1    5    2
## Ford Pantera L  15.8   8 351.0 264 4.22 3.170 14.5  0  1    5    4
## Ferrari Dino    19.7   6 145.0 175 3.62 2.770 15.5  0  1    5    6
## Maserati Bora   15.0   8 301.0 335 3.54 3.570 14.6  0  1    5    8
## Volvo 142E      21.4   4 121.0 109 4.11 2.780 18.6  1  1    4    2
```

7.5 Del data frame mtcars, filtra los datos para mostrar solo las filas donde *cyl* son iguales a 6

```
##                  mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4       21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
```

```
## Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Valiant        18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Merc 280       19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C      17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
```

7.6 Explora la estructura del data frame

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

7.7 Calcula la mediana de la columna “wt”

```
## [1] 3.325
```

8. Gráficos

En R, la función plot() es usada de manera general para crear gráficos.

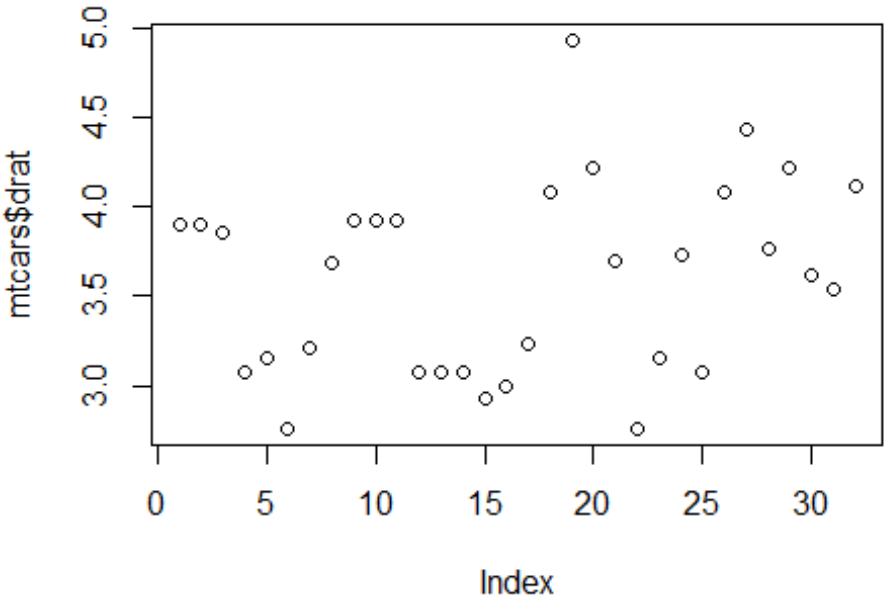
Esta función tiene un comportamiento especial, pues dependiendo del tipo de dato que le demos como argumento, generará diferentes tipos de gráfica. Además, para cada tipo de gráfico, podremos ajustar diferentes parámetros que controlan su aspecto, dentro de esta misma función. Puedes imaginar a plot() como una especie de navaja Suiza multifuncional, con una herramienta para cada ocasión. plot() siempre pide un argumento x, que corresponde al eje X de una gráfica. x requiere un vector y si no especificamos este argumento, obtendremos un error y no se creará una gráfica.

El resto de los argumentos de plot() son opcionales, pero el más importante es *y*. Este argumento también requiere un vector y corresponde al eje Y de nuestra gráfica. Dependiendo del tipo de dato que demos a *x* y *y* será el gráfico que obtendremos.

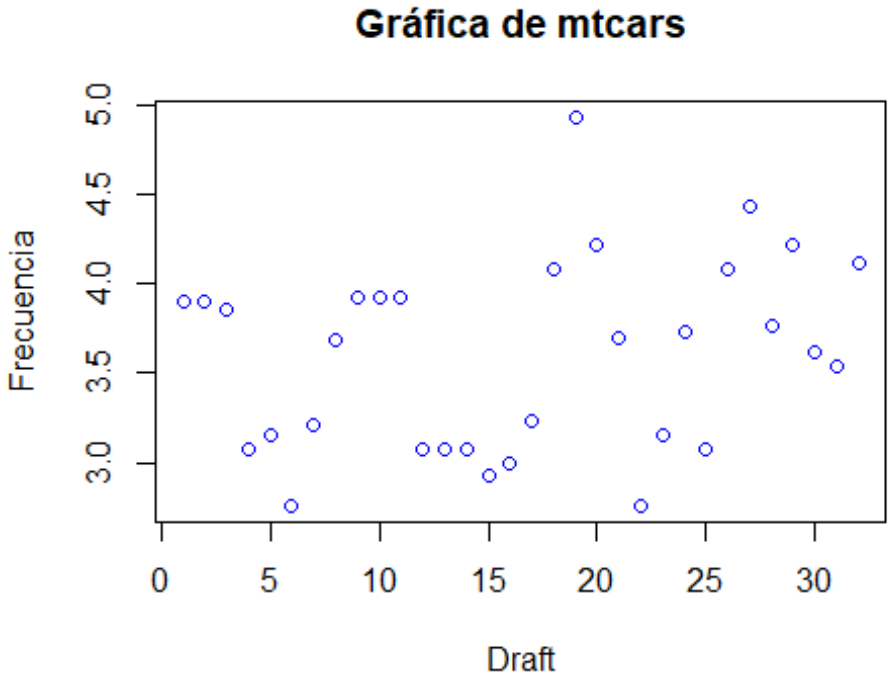
8.1 Gráfica de barras

Este es quizás el tipo de gráfico mejor conocido de todos. Una gráfica de este tipo nos muestra la frecuencia con la que se han observado los datos de una variable categórica. La función plot() puede generar gráficos de barra si damos como argumento *x* un vector de factor o cadena de texto, sin dar un argumento y.

Crea una gráfica de barras de la variable *draft*



8.2 Ajusta los parámetros gráficos con los argumentos **main, xlab, ylab y col**

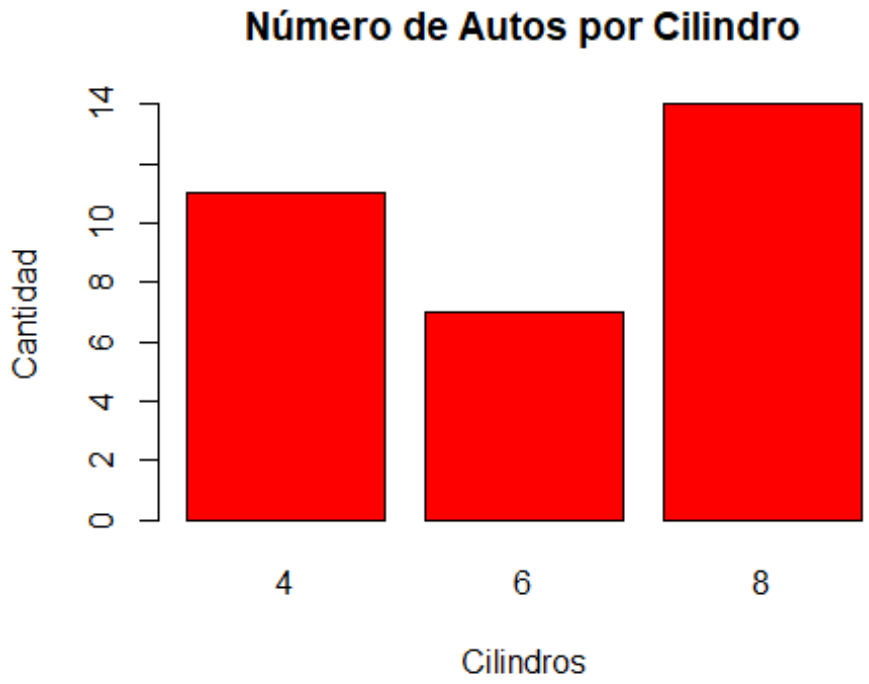


8.3. Gráfica de barras

Además de usar plot(), podemos crear gráficas de barra con la función barplot().

Barplot pide como argumento una matriz, que represente una tabla de contingencia con los datos a graficar. Este tipo de tablas pueden ser generadas con la función table().

Realiza un gráfico de barras que muestre la cantidad de autos para cada número de cilindros ("cyl")

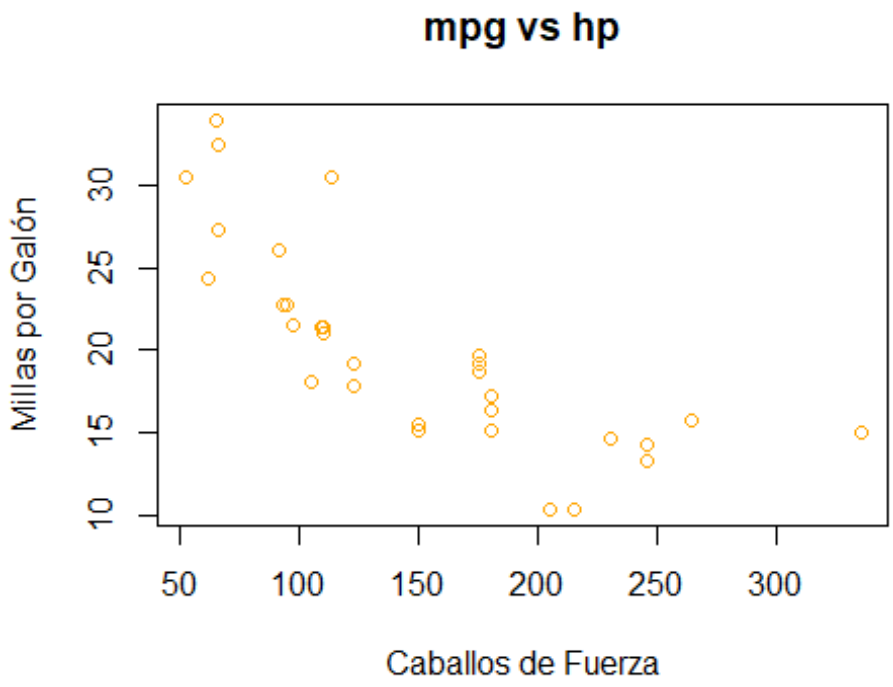


8.4 Gráfico de dispersión

Este tipo de gráfico es usado para mostrar la relación entre dos variables numéricas continuas, usando puntos. Cada punto representa la intersección entre los valores de ambas variables.

Para generar un diagrama de dispersión, damos vectores numéricos como argumentos *x* y *y* a la función plot().

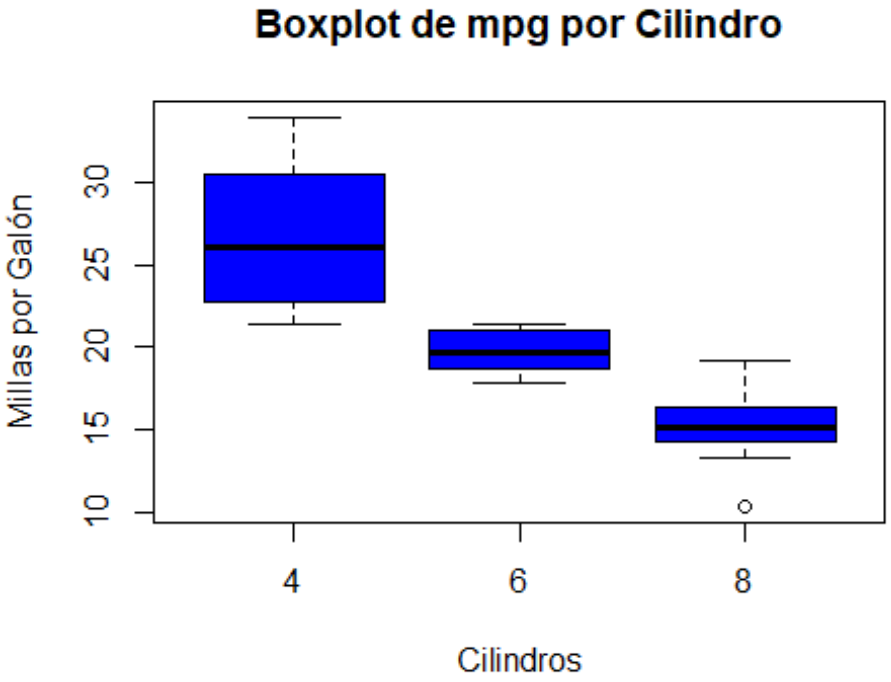
Ahora, realiza un gráfico de dispersión entre millas por galón y caballos de fuerza



8.5 Gráficos de caja y bigotes

Los boxplots, también conocidos como de caja y bigotes son gráficos que muestra la distribución de una variable usando cuartiles, de modo que de manera visual podemos inferir algunas cosas sobre su dispersión, ubicación y simetría.

Realiza un boxplot de millas por galón por número de cilindros



8.6 Histograma

Recuerda que un histograma es una gráfica que nos permite observar la distribución de datos numéricos usando barras. Cada barra representa el número de veces (frecuencia) que se observaron datos #en un rango determinado.

Usa la función “hist()” que muestre la distribución de la aceleración (qsec)