DiplomadosOnline.com

Formación y asesoría a tu alcance

Diplomado en minería de datos

Módulo 1

Primer Laboratorio en *R* y tarea entregable

Comandos básicos

R utiliza funciones para realizar operaciones. Para ejecutar una función escribimos funcname (INPUT1,INPUT2), donde las entradas (o argumentos) INPUT1 e INPUT2 le dicen a ***R*** cómo ejecutar la función. Una función puede tener cualquier número de argumentos. Por ejemplo, para crear un vector de números, utilizamos la función **"c ()"** (para concatenar). Cualquier número dentro del paréntesis se unen. El siguiente comando le dice a ***R*** que una los números 1, 3, 2 y 5, almacenándolos en un vector llamado ***x***. Cuando escribimos ***x***, se nos devuelve el vector.

```
> x <- c (1,3,2,5)
> x
[1] 1 2 3 5
```

Tenga en cuenta que el símbolo **"mayor que (>)"** no es parte del comando; más bien, es mostrado por ***R*** para indicar que está listo para que se inserte otro comando. También podemos guardar cosas usando **"=" en vez de "<-":**

```
>x= c (1,6,2)
> x
[1] 1 2 6
> y = c (1,4,3)
```

Pulsar la flecha direccional hacia arriba varias veces nos muestra los comandos anteriormente escritos, que luego pueden ser editados. Esto es útil ya que a menudo se desea repetir un comando similar. Además, si escribimos **"?Funcname"** permite a ***R*** abrir una nueva ventana de ayuda con información adicional acerca de la función "funcname".

Podemos decirle a ***R*** que sume dos conjuntos de números. Sumará el primer número de ***x*** con el primer número de ***y***, y así sucesivamente. Sin embargo, ***x*** e y deben tener la misma longitud. Podemos comprobar su longitud usando la función **"length ()"**.

```
> length (x)
[1] 3
> length (y)
[1] 3
> x + y
[1] 2 10 5
```

La función **"ls ()"** permite que veamos una lista de todos los objetos, tales como datos y funciones, que hemos guardado hasta ahora. La función **"rm ()"** puede ser utilizado para eliminar cualquiera que no nos interese.

```
> ls ()
[1] "x" "y"
> rm (x, y)
> ls ()
carácter (0)
```

También es posible eliminar todos los objetos a la vez:

```
> rm (list = ls ())
```

La función de la **"matrix ()"** se puede utilizar para crear una matriz de números. Antes de usar la función **"matrix()"**, podemos aprender más sobre ella:

```
>? matrix
```

El archivo de ayuda revela que la función **"matrix()"** tiene una serie de parámetros, pero por ahora nos centramos en los tres primeros: los datos (las entradas de la matriz), el número de filas y el número de columnas. En primer lugar, creamos un simple matriz.

```
> x = matrix (data= c (1,2,3,4), nrow = 2, ncol = 2)
> x
[1] [2]
[1], 1 3
[2], 2 4
```

Podemos también omitir la escritura de "**data =", "nrow = y" "ncol="** en la función **"matrix ()"** es decir, podemos escribir

```
> x = matrix (c (1,2,3,4), 2,2)
```

y esto tendría el mismo efecto. Sin embargo, a veces puede ser útil especificar los nombres de los argumentos pasados, ya que de lo contrario ***R*** asumirá que los argumentos de la función tienen el mismo orden al que aparece en el archivo de ayuda. Como muestra este ejemplo, ***R*** por defecto crea matrices llenándolas sucesivamente por columnas. Alternativamente, la opción byrow = TRUE se puede utilizar para llenar la matriz por filas.

```
> Matriz (c (1,2,3,4), 2,2, byrow = TRUE)
[1] [2]
[1,] 1 2
[2], 3 4
```

Nótese que en el comando anterior no asignamos la matriz a un valor ***x***. En este caso la matriz se imprime en la pantalla, pero no se guarda para cálculos futuros. La función **"sqrt ()"** devuelve la raíz cuadrada de cada elemento de un vector o matriz. El comando **"x ^ 2"** eleva cada elemento de x al cuadrado; cualquier potencia es posible, incluyendo fracciones o potencias negativas.

```
> sqrt (x)
[1] [2]
[1,] 1,00 1,73
[2], 1,41 2,00
> x ^ 2
[1] [2]
[1], 1 9
[2], 4 16
```

La función **"rnorm ()"** genera un vector de variables aleatorias normales, con el primer argumento **"n"** que indica el tamaño de la muestra. Cada vez que llamamos a esta función, se obtendrá una respuesta diferente. Aquí creamos dos conjuntos de números correlacionados, ***x*** e ***y,*** y utilizaremos la función **"cor ()"** para calcular la correlación entre ellos.

```
> x=rnorm (50)
> y=x+rnorm (50, mean=50, sd=.1)
> cor(x,y)
[1] 0.995
```

Por defecto, **"rnorm()"** crea variables aleatorias normales estándar con una media 0 y una desviación estándar de 1. Sin embargo, la media y la desviación estándar puede ser cambiados usando los parámetros **mean y sd**, como se ilustra arriba.

A veces queremos que nuestro código reproduzca exactamente el mismo conjunto de números al azar; podemos utilizar la función **"set.seed ()"** para hacer esto. la función **"set.seed ()"** toma un parámetro entero (arbitrario).

```
> set.seed (1303)
> rnorm (50)
[1] -1.1440 1.3421 2.1854 0.5364 0.0632 0.5022 -0.0004
```

Las funciones **"mean ()" y "var ()"** pueden ser utilizados para calcular la media y la varianza de un vector de números. La aplicación de la función **"sqrt ()"** a la salida de **"var ()"** dará a la desviación estándar. O simplemente podemos utilizar la función **"sd ()".**

```
> set.seed (3)
> y=rnorm (100)
> mean(y)
[1] 0.0110
> var(y)
[1] 0.7329
> sqrt(var(y))
[1] 0.8561
> sd(y)
[1] 0.8561
```

Gráficos

La función **"plot ()"** es la principal forma de representar gráficamente los datos en R. Por ejemplo, **plot (x, y)** produce un gráfico de dispersión de los x frente a y. Hay muchas opciones adicionales que se pueden pasar a la función **"plot()"**. Por ejemplo, pasando el parámetro "xlab" resultará en una etiqueta para el eje ***x***. Para obtener más información acerca de la función **"plot ()"**, escribimos ?**plot.**

```
> x=rnorm (100)
> y=rnorm (100)
> plot(x,y)
> plot(x,y,xlab=" Eje x",ylab=" Eje y",main=" Gráfico X vs Y")
```

A menudo vamos a querer guardar los gráficos de ***R***. El comando que utiliza para hacer esto dependerá del tipo de archivo que queremos crear. Por ejemplo, para crear un pdf, utilizamos la función **"pdf ()"**, y para crear un archivo jpeg, usamos la función **"jpeg ()"**.

```
> Pdf ( "Figura .pdf")
> Plot (x, y, col = "green")
> Dev.off ()
null device
1
```

La función **"dev.off"** le indica a ***R*** que hemos terminado la creación del gráfico. Alternativamente, se puede simplemente copiar la ventana de dibujo y pegarlo en un tipo de archivo apropiado, tal como un documento de Word.

La función **"seq()"** se puede utilizar para crear una secuencia de números. Por ejemplo, **seq (a, b)** crea un vector de enteros entre a y b. Existen muchas otras opciones: por ejemplo, la **seq (0,1, lenght = 10)** crea una secuencia de 10 números que están equidistantes entre 0 y 1. Si escribimos 3:11 es una abreviatura de seq(3,11) para argumentos enteros.

```
> x = seq (1, 10)
> x
[1] 1 2 3 4 5 6 7 8 9 10
> x= 1: 10
> x
[1] 1 2 3 4 5 6 7 8 9 10
> x = seq (-pi, pi, lenght = 50)
```

Ahora vamos a crear algunas gráficas más sofisticadas. La función **"contour()"** produce un gráfico de contorno con el fin de representar los datos en tres dimensiones; una gráfica de contorno es como un mapa topográfico. Toma tres argumentos:

1. Un vector para los valores **x** (la primera dimensión).

2. Un vector para los valores de y (la segunda dimensión).

3. Una matriz cuyos elementos corresponden al valor z (la tercera dimensión) para cada par de coordenadas (x, y).

Al igual que con la función **"plot ()"**, hay muchos otros parámetros que se pueden utilizar para ajustar la salida de la función **"contour()".** Para obtener más información acerca de éstos, veamos el archivo de ayuda escribiendo ?**contour.**

```
> y = x
> f = outer (x, y, function (x, y) cos (y) / (1 + x ^ 2))
> contour (x, y, f) nlevels = 45, add = T)
> fa = (f - t(f)) / 2
> contour (x, y, fa, nlevels= 15)
```

La función **"image()"** funciona de la misma forma que **"contour()"**, excepto que produce un gráfico con código de color cuyos colores dependerá del valor z. Esto es conocido como un mapa de calor, y en ocasiones se utiliza para representar la temperatura en pronósticos de clima. Alternativamente, **"persp ()"** se puede utilizar para producir un gráfico tridimensional. Theta y Phi son los parámetros que controlan los ángulos en los que la trama es visto.

```
> image (x, y, fa)
> persp (x, y, fa)
> persp (x, y, fa, theta = 30)
> persp (x, y, fa, theta = 30, phi = 20)
> persp (x, y, fa, theta = 30, phi = 70)
> persp (x, y, fa, theta = 30, phi = 40)
```

Indexación de datos

A menudo deseamos examinar parte de un conjunto de datos. Supongamos que nuestros datos están almacenado en la matriz **A**.

```
> A = matrix(1:16, 4, 4)
> A
[1] [2] [3] [4]
[1], 1 5 9 13
[2], 2 6 10 14
[3], 3 7 11 15
[4], 4 8 12 16
```

Entonces, escribimos

```
> A [2,3]
[1] 10
```

Seleccionará el elemento correspondiente a la segunda fila y la tercera columna. El primer número después el símbolo de corchete abierto siempre se refiere a la fila, y el segundo número se refiere siempre a la columna. También podemos seleccionar varias filas y columnas a la vez, al proporcionar vectores como índices.

```
> A [c (1,3), C (2,4)]
[1] [2]
[1], 5 13
[2,] 7 15
> A [1: 3, 2: 4]
[1] [2] [3]
[1], 5 9 13
[2], 6 10 14
[3,] 7 11 15
> A [1: 2,]
[1] [2] [3] [4]
[1], 1 5 9 13
[2], 2 6 10 14
> A [, 1: 2]
[1] [2]
[1], 1 5
[2], 2 6
[3], 3 7
[4], 4 8
```

Los dos últimos ejemplos incluyen ya sea ningún índice de las columnas o ningún índice para las filas. Estos indican que ***R*** debe incluir todas las columnas o todas las filas, respectivamente. ***R*** trata a una única fila o columna de una matriz como un vector.

```
> A [1],
[1] 1 5 9 13
```

El uso de un signo negativo "**-**" en el índice le dice a ***R*** que mantenga todas las filas o columnas excepto los indicados con el índice negativo.

```
> A [-c (1,3),]
[1] [2] [3] [4]
[1], 2 6 10 14
[2], 4 8 12 16
> A [-C (1,3), -C (1,3,4)]
[1] 6 8
```

La función **"dim ()"** muestra el número de filas seguido por el número de columnas de una matriz dada.

```
> dim (A)
[1] 4 4
```

Carga de datos

Para la mayoría de los análisis, el primer paso consiste en la importación de un conjunto de datos en ***R***. La función **"read.table()"** es una de las principales formas de hacer esto. El archivo de ayuda contiene detalles sobre cómo utilizar esta función. Podemos utilizar la función **"write.table ()"** para exportar datos.

Antes de intentar cargar un conjunto de datos, hay que asegurarse de que ***R*** conoce el lugar donde están almacenados los datos (directorio adecuado). Por ejemplo en Windows se puede seleccionar el directorio mediante **"Change dir..."** del menú Archivo. Sin embargo, los detalles de cómo hacer esto dependen del sistema operativo (por ejemplo, Windows, Mac, Unix) que se utiliza.

Comenzamos por la carga del conjunto de datos **"Auto".** Estos datos son parte de la biblioteca de **"ISLR"** (deben tenerla instaladada) de R, pero para ilustrar la función **"read.table ()"** lo cargaremos desde un archivo de texto. Los siguientes comandos cargará el archivo **"Auto.data**" en R y lo guarda como un objeto llamado **"Auto"**, en un formato conocido como *data frame*. Una vez que los datos se han cargado, la función **"fix()"** se puede utilizar para visualizar la data en una hoja de cálculo.

Sin embargo, la ventana debe estar cerrada antes de poder ingresar otros comandos en ***R***.

```
> Auto = read.table ("Auto.data")
> fix(Auto)
```

Tenga en cuenta que "Auto.data" es simplemente un archivo de texto, que puede ser abierto en el ordenador mediante un editor de texto estándar. A menudo es buena idea ver el conjunto de datos utilizando un editor de texto o de otros programas como Excel antes de cargarlo en ***R***.

Este conjunto de datos particular no se ha cargado correctamente, ya que ***R*** asume que los nombres de las variables son parte de los datos y están incluidos en la primera fila. El conjunto de datos también incluye un número de valores perdidos, indicado por un signo de interrogación "**?**". Los valores perdidos son comunes en los conjuntos de datos reales. El uso de la opción **header= T (o heder = TRUE)** en la función **"read.table ()"** le dice a ***R*** que la primera línea del archivo contiene los nombres de las variables, y el uso de la opción **na.strings** le dice a ***R*** que cada vez que se encuentre un carácter particular o conjunto de caracteres (como un signo de interrogación), debe ser tratado como un elemento que falta en la matriz de datos.

```
> Auto=read.table ("Auto.data", header =T,na.strings ="?")
> fix(Auto)
```

Excel es un programa de almacenamiento de datos en formato común. Una manera fácil de cargar tales datos en ***R*** es guardarlo como un archivo CSV (valores separados por comas) y luego usar la función "read.csv ()" para cargarlos.

```
> Auto=read.csv ("Auto.csv", header =T,na.strings ="?")
> fix(Auto)
> dim(Auto)
[1] 397 9
> Auto [1:4 ,]
```

La función de **"dim()"** nos dice que los datos tienen 397 observaciones, o filas, y nueve variables, o columnas. Hay varias maneras de trabajar con datos faltantes. En este caso, sólo cinco de las filas contienen observaciones que faltan, elegimos la función **"na.omit()"** para que simplemente elimine estas filas.

```
> Auto=na.omit(Auto)
> dim(Auto)
[1] 392 9
```

Una vez cargada la data correctamente podemos usar la función **"names()"** para ver los nombres de las variables.

```
> names(Auto)
[1] "mpg " "cylinders " " displacement" "horsepower "
[5] "weight " " acceleration" "year" "origin "
[9] "name"
```

Gráficos adicionales y resúmenes numéricos.

Podemos utilizar la función **"plot ()"** para producir gráficos de dispersión de variables cuantitativas. Sin embargo, si escribimos los nombres de las variables producirá un error mensaje, porque ***R*** no sabe buscar en los datos "**Auto"** estas variables.

```
> plot(cylinders , mpg)
Error in plot(cylinders , mpg) : object 'cylinders ' not found
```

Para hacer referencia a una variable, hay que escribir el conjunto de datos y el nombre de la variable unido al símbolo $. Alternativamente, podemos utilizar la función **"attach ()"** para decirle a ***R*** tome las variables de estos datos y los tenga disponibles por su nombre.

```
> plot(Auto$cylinders , Auto$mpg )
> attach (Auto)
> plot(cylinders , mpg)
```

La variable de **cylinders** se almacena como un vector numérico, por lo que ***R*** la ha tratado como cuantitativa. Sin embargo, ya que sólo hay un pequeño número de posibles valores para los **cylinders**, uno puede preferir tratarla como una variable cualitativa.

La función **"as.factor ()"** convierte las variables cuantitativas en cualitativas.

```
> cylinders =as.factor (cylinders )
```

Si la variable que representa el eje ***x*** es categórica, entonces, los diagramas de caja (boxplots) se generan automáticamente con la función "**plot ()"**. Como es costumbre, una serie de opciones se puede especificar con el fin de personalizar los gráficos.

```
> plot(cylinders , mpg)
> plot(cylinders , mpg , col ="red ")
> plot(cylinders , mpg , col ="red", varwidth =T)
> plot(cylinders , mpg , col ="red", varwidth =T,horizontal =T)
> plot(cylinders , mpg , col ="red", varwidth =T, xlab=" cylinders ",
ylab ="MPG ")
```

La función **"hist ()"** se puede utilizar para trazar un histograma. Tenga en cuenta que col = 2 tiene el mismo efecto que **col = "red".**

```
> hist(mpg)
> hist(mpg ,col =2)
> hist(mpg ,col =2, breaks =15)
```

Los función **"pairs()"** crea un gráfico de dispersión matricial es decir, un diagrama de dispersión para cada par de variables de cualquier conjunto de datos dado. También podemos producir la matriz de dispersión con sólo un subconjunto de las variables.

```
> pairs(Auto)
> pairs(  mpg + displacement + horsepower + weight + acceleration , Auto)
```

En combinación con la función **"plot ()", "identify()"** proporciona un método interactivo para identificar el valor de una variable en particular en un gráfico. **"identify()"** tiene tres argumentos: el eje ***x***, el eje ***y*** y la variable cuyos valores nos gustaría ver impreso para cada punto. A continuación, hacemos clic en un punto dado en la gráfica y causará que ***R*** muestre el valor de la variable de interés. Haciendo clic derecho en la gráfica salimos de la función **"identify()".** Los números impresos bajo la función **"identify()"** corresponden a las filas de los puntos seleccionados.

```
> plot(horsepower ,mpg)
> identify (horsepower ,mpg ,name)
```

La función de **"summary ()"** produce un resumen numérico de cada variable en un conjunto de datos en particular.

```
> summary (Auto)
mpg                cylinders          displacement
Min. : 9.00        Min . :3.000       Min. : 68.0
1st Qu .:17.00     1st Qu .:4.000     1st Qu .:105.0
Median :22.75      Median :4.000      Median :151.0
Mean :23.45        Mean :5.472        Mean :194.4
3rd Qu .:29.00     3rd Qu .:8.000     3rd Qu .:275.8
Max. :46.60        Max . :8.000       Max. :455.0
horsepower         weight             acceleration
Min. : 46.0        Min . :1613        Min . : 8.00
1st Qu.: 75.0      1st Qu .:2225      1st Qu .:13.78
Median : 93.5      Median :2804       Median :15.50
Mean :104.5        Mean :2978         Mean :15.54
3rd Qu .:126.0     3rd Qu .:3615      3rd Qu .:17.02
Max. :230.0        Max . :5140        Max . :24.80
year               origin             name
Min. :70.00        Min . :1.000       amc matador : 5
1st Qu .:73.00     1st Qu .:1.000     ford pinto : 5
Median :76.00      Median :1.000      toyota corolla : 5
Mean :75.98        Mean :1.577        amc gremlin : 4
3rd Qu .:79.00     3rd Qu .:2.000     amc hornet : 4
Max. :82.00        Max . :3.000       chevrolet chevette : 4
                                      (Other) :365
```

Para las variables cualitativas, tales como nombre, ***R*** listará el número de observaciones que caen en cada categoría. También podemos producir un resumen de una sola variable.

```
> summary (mpg)
Min.      1st Qu.     Median    Mean      3rd Qu.      Max .
9.00      17.00       22.75     23.45     29.00
          46.60
```

Una vez que hemos terminado de usar ***R***, escribimos **q ()** con el fin de apagarlo, o salir. Al salir de ***R***, tenemos la opción de guardar el espacio de trabajo actual por lo que todos los objetos (tales como conjuntos de datos) que hemos creado en esta sesión de ***R*** estará disponible la próxima vez. Antes de salir de ***R***, es posible que desee guardar un registro de todos los comandos que escribió en el último período de sesiones; esto puede llevarse a cabo usando la función **"savehistory ()"**. La próxima vez que entramos en ***R***, podemos cargar el histórico usando la función **"loadhistory ()".**

Tarea entregable laboratorio de R

1. Este ejercicio se refiere al conjunto de datos College, que se puede encontrar en el archivo College.csv. Contiene una serie de variables para 777 diferentes universidades y escuelas superiores de los EE.UU.. Las variables son:

    - Private: Indicador público / privado
    - App: Número de solicitudes recibidas
    - Accept: Número de solicitantes aceptados
    - Enroll: Número de nuevos alumnos matriculados
    - Top10 perc: Estudiantes nuevos correspondiente al diez por ciento superior del "high school"
    - Top25 perc: Estudiantes nuevos correspondiente al veinticinco por ciento superior del "high school"
    - F.Undergrad: Número de estudiantes de tiempo completo

- P.Undergrad: Número de estudiantes a tiempo parcial
- Outstate: Fuera de la matrícula estatal
- Room.Board: los costos de pensión
- Books: los costos estimados de libros
- Personal: Se estima que el gasto personal
- Phd: Porcentaje de profesores con doctorados
- Terminal: Porcentaje de profesores con grado de terminales
- S.F.Ratio: Proporción de estudiantes / profesores
- perc.alumni: Porcentaje de alumnos que donan
- Expend: Los gastos de instrucción por estudiante
- Grad.Rate: Porcentaje de graduación

Antes de leer los datos en ***R***, se puede ver en Excel o en un editor de texto.

(A) Utilizar la función **"read.csv ()"** para leer los datos en R. Llamar a los datos cargados **College**. Asegúrese de que tiene los datos en el directorio de R.

(B) Mira los datos utilizando la "función fix ()". Nótese que la primera columna es sólo el nombre de cada universidad. No deberíamos hacer que ***R*** trate esto como datos. Sin embargo, puede ser útil para tener estos nombres más adelante. Pruebe los siguientes comandos:

```
> rownames (College) = College [ ,1]
> fix (College)
```

Deberíamos ver con el row.names que ahora hay una columna con el nombre de cada universidad guardada. Esto significa que ***R*** ha dado a cada fila un nombre a la universidad correspondiente. ***R*** no tratará de realizar cálculos con los nombres de las filas. Sin embargo, todavía tenemos que eliminar la primera columna de los datos. Intentemos

```
> College = College [ ,-1]
> fix (College)
```

Ahora podemos ver que la primera columna de datos es private. Notemos que otra columna con el nombre "row.names" ahora aparece antes de la columna private. Sin embargo, esto no es una columna de datos sino más bien el nombre que ***R*** está dando a cada fila.

(C) i. Utilice **"summary ()"** para producir un resumen numérico de las variables del conjunto de datos.

ii. Usa **"pairs" ()** para producir una matriz de dispersión de las primeros diez columnas o variables de los datos. Recordemos que puede hacer referencia a los primeros diez columnas de una matriz A usando A [, 1: 10].

iii. Utilice la función **"plot ()"** para producir gráficos de caja de lado a lado de Outstate vs Private.

iv. Crear una nueva variable cualitativa, llamada "Elite", para tratar la variable Top10perc. Vamos a dividir las universidades en dos grupos en función de si o no la proporción de los estudiantes que vienen del 10% superior de su escuela secundaria supera el 50%.

```
> Elite =rep ("No",nrow(college ))
> Elite [college$Top10perc >50]=" Yes"
> Elite =as.factor (Elite)
> college =data.frame(college ,Elite)
```

Utilice el **"summary ()"** para ver cómo muchas universidades de élite existen. Ahora usa la función "**plot ()**" para producir de lado a lado los diagramas de caja de Outstate vs Elite.

v. Utilizar la función "**hist ()**" para producir algunos histogramas con diferentes números de gráficas para algunas de las variables cuantitativas. Con el comando par **(mfrow = c (2,2))** se hacen cosas útiles: dividirá la ventana de impresión en cuatro regiones por lo que cuatro gráficas se pueden realizar de forma simultánea. La modificación de los parámetros de esta función dividirá la pantalla de otras maneras.

vi. Continuar explorando los datos, y proporcionar un breve resumen de lo que se descubre.

2. Este ejercicio consiste en el conjunto de datos "Auto" estudiado anteriormente. Asegúrese que los valores que faltan se han eliminado de los datos.

(A) ¿Cuál de los predictores son cuantitativos y cualitativos?

(B) ¿Cuál es el rango de cada predictor cuantitativo? Puede contestar esto usando la función **"range ()"**.

(C) ¿Cuál es la media y la desviación estándar de cada predictor cuantitativo.

(D) Ahora quite la decima observación. ¿Cuál es el rango, media y desviación estándar de cada predictor en el subconjunto de los datos que queda?

(E) Utilizando el conjunto completo de datos, investigar los predictores de forma gráfica, use diagramas de dispersión u otras herramientas de su elección. Cree algunas gráficas destacando las relaciones entre los predictores. Comente sus conclusiones.

(F) Supongamos que deseamos predecir el rendimiento de la gasolina (millas por galón) sobre la base de las otras variables. ¿Sus gráficas sugieren que cualquiera de los otras variables podrían ser útiles en la predicción de millas por galón? justifique su respuesta.

3. Este ejercicio consiste en el conjunto de datos de vivienda de Boston.

(A) Para empezar, cargaremos el conjunto de datos Boston. El conjunto de datos Boston es parte de la biblioteca MASS de ***R***.

```
> library (MASS)
```

Ahora el conjunto de datos está contenido en el objeto Boston.

```
> Boston
```

Lee sobre el conjunto de datos:

```
>? Boston
```

¿Cuántas filas hay en este conjunto de datos? ¿Cuántas columnas? ¿Qué representan las filas y las columnas?

(B) Hacer algunos diagramas de dispersión por pares de los predictores (columnas) en este conjunto de datos. Describir sus hallazgos.

(C) ¿Alguno de los predictores esta asociado con la tasa de criminalidad per cápita? Si es así, explicar la relación.

(D) ¿Alguno de los suburbios de Boston parecen tener todos altas tasas de criminalidad? ¿Las tasas de impuestos? ¿La proporción alumno-maestro? comentar el rango de cada predictor.

(E) ¿Cuántas comunidades (suburbs) en este conjunto de datos están vinculados al río Charles?

(F) ¿Cuál es la mediana de alumnos por maestro entre las comunidades de este conjunto de datos?

(G) ¿Qué comunidad de Boston tiene la media más baja de ocupación de casas? ¿Cuáles son los valores de los otros predictores para esa comunidad, y cómo esos valores se comparan en general con los otros predictores? Opina sobre tus hallazgos.

(H) En este conjunto de datos, ¿cuántas de las comunidades tienen más de siete habitaciones por vivienda? Más de ocho habitaciones por vivienda? Comentar las comunidades con un promedio de más de ocho habitaciones por vivienda.

Por favor escribir un informe completo en formato Word que incluya gráficos y comentarios sobre sus hallazgos.