ESCUELA NACIONAL DE ESTUDIOS SUPERIORES UNIDAD UNAM MÉRIDA

LICENCIATURA EN CIENCIAS DE LA TIERRA
ESTADÍSTICA APLICADA
Prof. Edlin Guerra Castro | Prof. María Muciño

Laboratorio 1: Estadística descriptiva y R

El objetivo principal de este laboratorio es introducirlos a R y RStudio, las herramientas computacionales que utilizaremos a lo largo del semestre para aprender a aplicar los conceptos más importantes de *Estadística Aplicada*, pero en especial, para aprender a procesar y analizar datos reales.

El programa R, sus versiones actualizadas y todos los paquetes con funciones, así como otra información relevante se encuentre en los repositorios de R conocidos como **CRAN** (The Comprehensive R Archive Network). Los distintos servidores distribuidos en todo el mundo, conforman el CRAN y son conocidos como los **CRAN mirrors** (de espejo). Para descargar R requieres antes escoger un **CRAN mirror**.

RStudio es un entorno de desarrollo integrado (IDE) para **R**. Incluye una consola, editor de comandos y líneas de programación que admite la ejecución directa de código, así como herramientas para graficar, documentar, registrar el historial de comandos ejecutados, acceder a archivos, y muchas cosas más desde la gestión de un espacio de trabajo. **RStudio** hace que el trabajar con **R** sea más poderoso, y a su vez simple. Para que tengan una idea, esta guía se escribió desde **RStudio**.

Lo que harán ahora es:

1. Descargar e instalar R y Rstudio en su computador, visitando los sitios de R y de RStudio.
2. Una vez instalado y que se cargue la interface de *Rstudio*, generen un archivo de códigos vacío, siguiendo el símbolo de hoja en blanco y cruz verde justo debajo del menú **File**, es la primera opción. Un atajo para crearlo es con las teclas **Ctrl+Shift+N**.
3. Este archivo vacío es un editor de texto inteligente, que reconoce ciertos caracteres y palabras para autocompletar los comandos. Ese espacio será su borrador de comandos y notas. En este archivo irá pegando y ejecutando todos los códigos que a continuación se explican.
4. Acá un pequeño ejemplo, copien y peguen todo le código del recuadro de abajo en su editor de comandos. Luego, coleque el cursor sobre al inicio de la primera linea de comando y presionen el botón **Run** en la esquina superior derecha de su achivo vación; alternativamente, presione **Ctrl+ENTER**. ¿qué ocurrió? haga esto para las primeras seis líneas de comando. Cuando llegue a **p <- ggplot...** selecciones todas las líneas faltantes del código y presione Run o **Ctrl+ENTER**. ¿Qué ocurrió?

```r
install.packages("gcookbook") #instalación del paquete tutorial de gráficos
install.packages("grid") #instalación de paquete gráfico grid
install.packages("ggplot2") #instalación de paquete gráfico ggplot2
library(gcookbook) #cargar paquete gcookbook en la sesión
library(grid) #cargar paquete grid en la sesión
library(ggplot2) #cargar paquete ggplot2 en la sesión

data("climate") #importar datos de clima global del paquete gcookbook

p <- ggplot(subset(climate, Source == "Berkeley"),
            aes(x = Year, y = Anomaly10y)) +
  geom_line() +
  annotate(
    "segment",
    x = 1850,
    xend = 1820,
    y = -.8,
    yend = -.95,
    colour = "blue",
    size = 2,
    arrow = arrow()
  ) +
  annotate(
    "segment",
    x = 1950,
    xend = 1980,
    y = -.25,
    yend = -.25,
    arrow = arrow(
      ends = "both",
      angle = 90,
      length = unit(.2, "cm")
    )
  ) +
  theme_bw() +
  xlab("Año") +
  ylab("Anomalías de la temperatura global (°C)")
p
```

Parte 1: Objetos, funciones y paquetes

R es un lenguaje orientado a objetos, lo que significa que las variables, datos, funciones, resultados, etc., se guardan en la memoria activa del computador en forma de objetos con un nombre específico dado por el usuario en cada sesión. Los objetos se manipulan mediante funciones (que, a su vez, pueden ser tratados como objetos) y operadores. La ventana de la consola es donde se escriben los comandos, después de un indicador o prompt `>` que notifica cuando R está listo para recibir la siguiente instrucción. La tecla `esc` aborta la tentativa de esa línea de comando y da la señal para que aparezca un nuevo prompt. Dos prompts `> >` seguidos invalida esa línea de comando. Si aparece un signo de `+` es que la linea de comando está incompleta y requiere ser completada ante de devolver un resultado. Si aparece un mensaje de *Error* significa que él comando o instrucción no tuvo efecto. Si aparece un *Warning* significa que **R** efectuó la instrucción anterior, pero tuvo algún obstáculo mismo que es descrito inmediatamente. Con las flechas del arriba y abajo del teclado, aparece la linea de comando inmediata anterior y es una manera de no re-escribir dichas líneas cada vez. El signo de número `#` indica un comentario que no será tomado en cuenta hasta que aparezca un nuevo prompt.

Para poder ver los objetos que se encuentran en una sesión activa de **R**, se puede escribir la función de enlistar `ls`, o si estás en **R-Studio**, verifica directamente la pestaña **Environment** en el panel superior derecho.

```
ls()
```

El nombre de un objeto se asigna con el operador `<-` o `=`, y puede estar hecho de letras, números y puntuación. *Nota*: Usar el mismo nombre para dos objetos distintos implica perder la asignación del primer objeto

```
y.y <- 10 * 10
z.12 <- 81 / 9
unam <- "Universidad Nacional Autonoma de Mexico"

xx <- 4
xx
xx <- "xx ya no es el mismo"
xx
```

Para ver el tipo y longitud de un objeto se pueden usar las funciones `mode`, `class`, `length` o `str`. Úselas con los objetos creados:

```r
#Solo con y.y, úsela con los otros dos objetos
mode(y.y)
class(y.y)
length(y.y)
str(y.y)
```

Noten en la consola, del lado izquierdo, que hay números entre corchetes. Esto representa el número de elementos que siguen en esa línea antes de llegar a una línea abajo. Luego de ejecutar la siguiente línea de código ¿puede deducir qué hace `seq` y la relación con los números en corchetes?

```r
muchos <- seq(0, 50)
muchos
```

R es sensible a mayúsculas, pero no a los espacios:

```r
compar
Compar
sum    (3+2)
sum(3+2)
```

En **R** se usan tres tipos de lementos: números (*numeric*), letras (*character*, siempre entre comillas), lógicos (*logical*). Estos elementos son usados para generar objetos. Los objetos pueden clasificarse como:

A) *Vector*: una columna o una fila de elementos, que pueden ser numéricos, de caracteres de texto, de operadores lógicos, etc. Cuando se trata de una variable categórica, el vector puede ser tratado como un factor, y los niveles del factor corresponden a las categorías de dicha variable. Un vector se crea con la funcion `c`, deguido de paréntesis `()` que incluyen todos los elementos del vector separados por coma.

```r
vect1 <- c(2,4,6,3,7,8,9,2)
vect1

vect2 <-c("esp", "ing", "por")
vect2

#Puedes preguntar si un vector tiene elementos de un tipo en particular:
is.numeric(vect1)
is.numeric(vect2)
is.character(vect1)
```

```r
is.character(vect2)

#Qué hace esto:
vect1[5]
vect2[2]
```

Una de las grandes fortalezas de **R** es que permite el acceso a los elementos de un objeto a través de una selección de subconjuntos de éstos. El *sub-setting* es una manera eficiente y flexible de acceder selectivamente a los elementos de un objeto, y se hace mediante el uso de corchetes `[]`.

B) *Matrix*: es un arreglo bidimensional de columnas y renglones, sobre el cual se pueden aplicar operaciones algebraicas. Cada elemento de una matriz puede ser accedido con `[,]`, delante de la coma iría el número de la o las filas, luego de la coma, el número de la o las columnas. La combinación específica de una fila y columna lleva al valor de la celda.

```r
#creando una matriz combinando tres vectores
matr1 <- rbind(c(1,2,3),c(4,5,6),c(7,8,9))
matr1
is.factor(matr1)
is.numeric(matr1)

#creando una matriz con una función

matr2 <- matrix(data = seq(1:9), nrow = 3, ncol = 3, byrow = TRUE)

#Note la diferencia entre matr2 y matr3 si cambiamos el argumento byrow a FALSE
matr3 <- matrix(data = seq(1:9), nrow = 3, ncol = 3, byrow = FALSE)

#Selección de segunda y tercera columna de matr2
matr2[,c(2,3)]

#Selección de primera fila de matr2
matr2[1,]

#Selección el valor de la primera fila y segunda columna de matr2
matr2[1,2]
```

C) *Array*: es un arreglo de dimensiones k>2.

```r
n <- 3
k <- 2
j <- 4
samp <- array(dim = c(n,k,j))
samp

is.factor(samp)
is.numeric(samp)
```

Los vectores, matrices y arreglos solo pueden tener elementos del mismo tipo (e.g. numéricos, lógicos, letras)

D) *Dataframe*: es una tabla compuesta de uno o más vectores de la misma longitud, pero con elementos que pueden ser de diferentes tipos. Es el formato ideal para bases de datos, ya que las variables suelen ser de diferente naturaleza (i.e. continuas, nominales, etc.). Se puede acceder a ellas usando la sintaxis de matrices, pero también con el signo $ para identificar a la columna por su nombre. A medida que avancemos, usaremos paquetes como **dplyr** y **tidyr** diseñados para manipular eficientemente los dataframes.

```r
iris
data(iris)

#haga click sobre <promise> de iris en su ambiente, luego explore visualmente.
# ¿Qué es iris?

dim(iris)    #Pide las dimensiones de la tabla iris
names(iris)  #Pide los nombres de las columnas en iris
iris[,"Species"] # Selecciona la columna por su nombre
iris[,5] # Selecciona la columna por su número de columna
iris$Species     # Selecciona la columna por su asignación en la tabla 'iris'
```

E) *List*: Este objeto puede ser visto como un estante, ya que agrupa ordenadamente objetos de diferente tipo (e.g. vectores, arreglos, tablas, otras listas, etc). Se usa mucho para devolver los resultados de una función que se encuentran en la forma de una colección de objetos:

```r
mi_lista <- vector(mode = "list")

mi_lista[[1]] <- iris
mi_lista[[2]] <- y.y
mi_lista[[3]] <- unam

mi_lista
```

Parte 2: Estadística descriptiva

Las funciones están organizadas en paquetes. El paquete denominado `base` constituye el núcleo de **R** y contiene las funciones básicas del lenguaje. Otro paquete muy importante es `stats` e incluye las funciones estadísticas más importantes y básicas de **R**. Ambos ya vienen preinstalados en **R**. Existen muchos paquetes, a medida que se requiera el uso de alguno específico se irá indicando para que lo descarguen e instalen. Por ahora les adelanto el uso de un set de paquetes agrupados en una familia de paquetes muy usados para ordenar, limpiar, modelar, reproducir, comunicar y graficar datos; este grupo de paquetes se les denomina tidyverse. Para instalarlos pueden escribir en la consola:

```
#Para instalar ggplot2 (realizar gráficos de alta calidad)
install.packages("ggplot2")

#Para depurar y reordenar bases de datos, instala: tidyr
install.packages("tidyr")

#Para administrar bases de datos: usa dplyr
install.packages("dplyr")

#Para importar datos desde Excel: readxl
install.packages("readxl")

#para estimar simetría y curtosis
install.packages("moments")
```

Alternativamente, puedes usar la ventala *Tools/install packages…*, se desplegará una ventana para que escribas el nombre del paquete a instalar. Los paquetes se instalan una sola vez, siempre que estés en el mismo computador. Para usarlos debes incluirlos en tu sesión de trabajo cada vez que se inicia la sesión. Esto se logra con la función `library`:

```
library("tidyverse")
```

Antes de usar paquetes, hagamos un análisis exploratorio de los datos *iris*. Antes de copiar el código, busque en la pestaña *Help* qué es *iris*. Efectuamos un gráfico de dispersión entre las variables "largo". Mida el grado de asociación ¿cómo lo haría?

```
plot(iris$Sepal.Length, iris$Petal.Length)

#¿qué hace cor()?
cor(iris$Sepal.Length, iris$Petal.Length)
```

Usemos el paquete `ggplot2`para mejorar el gráfico:

```
pp <- ggplot(data = iris, aes(x = Sepal.Length, y = Petal.Length, colour = Species)) +
  geom_point()

pp

#Mejoremos con capas
pp +
  theme_bw() +
  xlab("Largo del sépalo (cm)") +
  ylab("largo del pétalo (cm)") +
  scale_y_continuous(breaks = seq(1, 7, 1)) +
  scale_x_continuous(breaks = seq(4, 8, 1)) +
  annotate(
    geom = "text",
    x = 4.3,
    y = 6.5,
    parse = TRUE,
    label = "italic(r) == 0.87"
  )

#¿cuál gráfico le gustó más?
```

YA entendiendo la importancia de los paquetes, hagamos un análisis exploratorio a los datos de los pinguinos de la Antártida del género *Pygoscelis*. Lo primero que debe hacer es instalar el paquete de datos `palmerpenguins` y cárguelo en su sesión. Luego haga lo siguiente:

1. Busque en la pestaña *Help* qué es *palmerpenguin*.
2. Identifique la base de datos `penguins` y cárguela en su ambiente global con `data(penguins)`.
3. Identifique cuántas variables hay, cuál es la naturaleza de cada una de ellas (tipo de variable, escala), así como cuáles pueden ser consideradas causales y cuáles respuesta.
4. Genere un gráfico de dispersión entre las variables `body_mass_g` y `flipper_lemgth_mm`. Mida el grado de asociación ¿cómo lo haría? Una forma de hacer estas cosas es graficando la asociación, y estimando la correlación (método que se desarrollará en otro laboratorio, pero acá es pide con fines demostrativos)

```
plot(penguins$body_mass_g, penguins$flipper_length_mm)

#¿qué hace cor()?
cor(penguins$body_mass_g, penguins$flipper_length_mm)
```

```
#Si no obtuvo resultado, trate de resulverlo con el argumento "use"
```

Usemos el paquete `ggplot2` para mejorar el gráfico:

```
pp <- ggplot(data = penguins, aes(x = flipper_length_mm,
                                  y = body_mass_g,
                                  colour = species))+
      geom_point()

pp

#Mejoremos con capas
pp +  theme_bw()+
      xlab("Largo de aleta (mm)")+
      ylab("Masa corporal (g)")+
      scale_y_continuous(breaks = seq(2600,6400,400))+
      scale_x_continuous(breaks = seq(170,240,5))

#¿cuál gráfico le gustó más?
```

Llegados a este punto, vamos hacer algo de estadística descriptiva. Esta es la parte que deberán entregar como tarea. Calculen promedio, varianza, desviación estándar, valor mínimo y máximo, cuartiles, simetría y curtosis a la variable masa corporal. Usen para ello las funciones recomendadas y responda las siguientes preguntas:

```
# Para facilitar cálculos, vamos a remover los datos sin registro
# (identificados con NA), usando el siguiente código:

penguins2 <- penguins %>%
  na.omit()
```

PREGUNTAS

1. Copia el comando que sigue. ¿Qué se calculó?

```
xx<-penguins2$body_mass_g
sum(xx, na.rm = T)/length(xx)
```

2. Busca y aplica una función que ejecute la linea de comando anterior. PISTA: escribe `?mean` en la cónsola y enter para ampliar tu búsqueda.

3. Calcula la mediana de la masa corporal usando la función correspondiente.
4. Calcula la varianza de la masa corporal usando el comando `var()`
5. ¿Cuál es la diferencia entre estas dos fórmulas? ¿Representan lo mismo?

```
sum((xx-mean(xx))\^2)/length(xx)

sum((xx-mean(xx))\^2)/(length(xx)-1)
```

6. Con base en el valor de la varianza y usando operadores aritméticos, calcula la desviación estándar de la masa corporal. Confirma tu resultado usando la función `sd()`.
7. Explore el rango de la masa corporal identificando mínimos y máximos con la función `min()` y `max()`, respectivamente.
8. Ahora estime los cuartiles de la masa corporal con la función `quantile()`
9. Describa la forma de la distribución de la masa corporal usando la simetría y curtosis con las funciones `skewness()` y `kurtosis()`.
10. Todas estas estimaciones ignoran las posibles diferencias en la masa corporal entre las especies. ¿Qué le dice este gráfico?

```
boxplot(body_mass_g~species, data = penguins2)
```

11. Calcule estos estimadores para cada especie usando el paquete **dplyr** y sus funciones `group_by()` y `summarize()`. Estas líneas de comando lo ayudarán (interprete los resultados):

```
library(dplyr)

penguins %>%
  group_by(species) %>%
  summarise(media = mean(body_mass_g, na.rm =T),
            desviacion = sd(body_mass_g, na.rm =T),
            simetria = skewness(body_mass_g, na.rm =T),
            curtosis = kurtosis(body_mass_g, na.rm =T))
```

12. Usando como guía el libro digital R Graphics Cookbook, genere: (i) una distribución de frecuencias con histograma, (ii) una distribución de frecuencias basada en densidad, (iii) un diagrama de cajas que incluya promedio. En los tres casos la masa corporal debe distinguirse por especie.

Recuerde cargar las respuestas a estas preguntas en el formato de Google Doc generado en el Google Classroom.