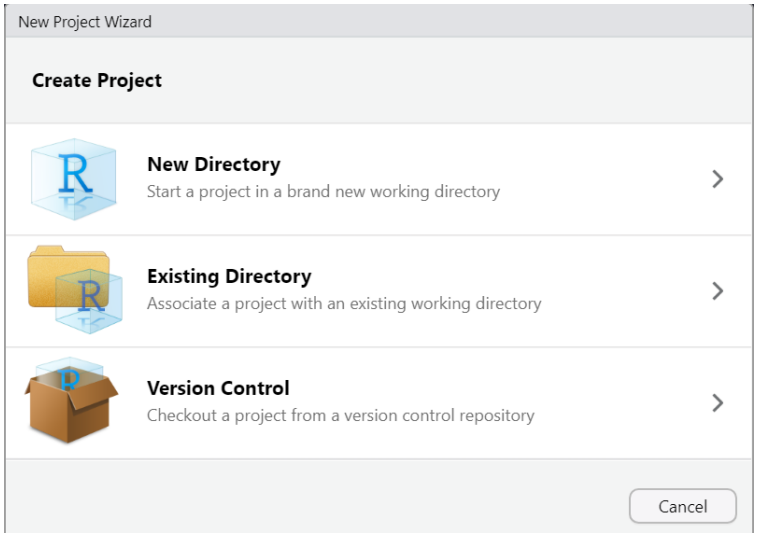
1. Introducción al modelaje estadístico

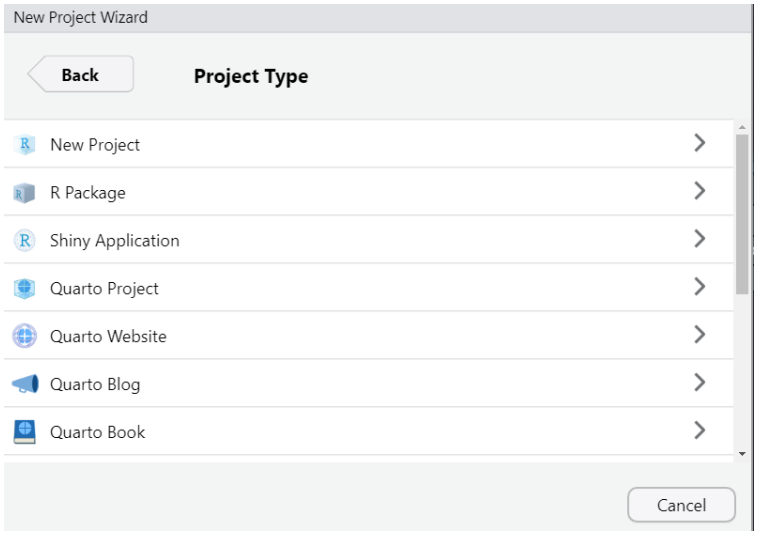
Armando Ocampo

Trabajando con proyectos

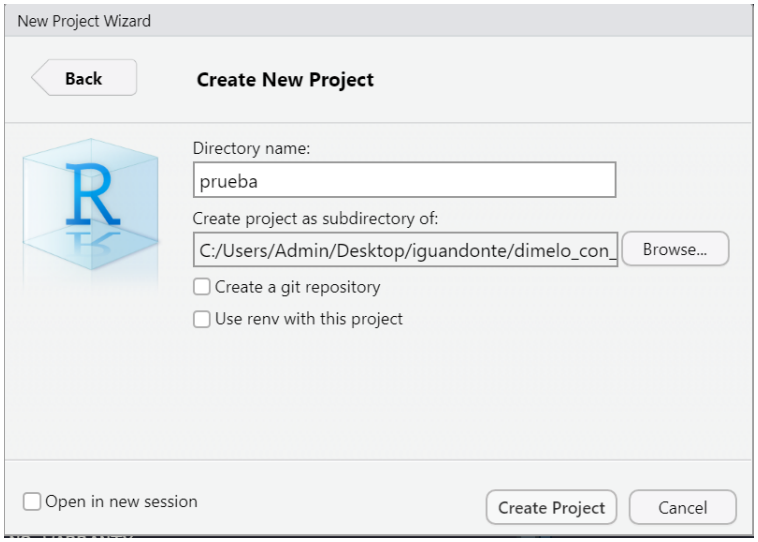
Antes de comenzar, vamos a generar un proyecto. Estos nos permiten organizar nuestro trabajo; tener imágenes, scripts, y datasets en un mismo sitio. Para esto, nos dirigiremos al apartado file, o archivo y seleccionaremos 'nuevo proyecto'



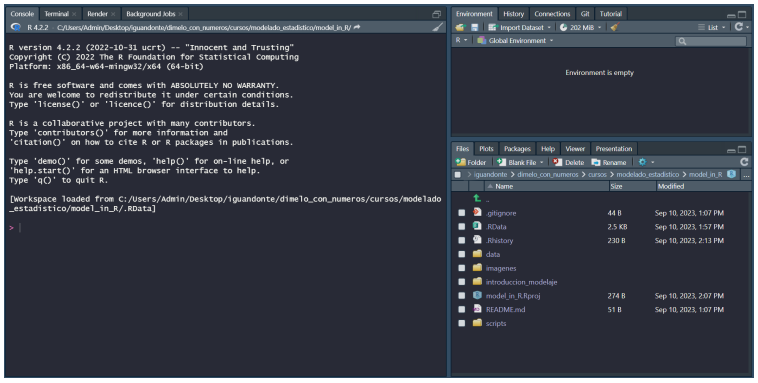
Se generará un apartado para nuestro nuevo proyecto. Por el momento nos enfocaremos solo en 'New directory'. En el apartado de 'tipo de proyecto' seleccionaremos 'nuevo proyecto'.



Ahora, indicaremos la ubicación donde se guardará el proyecto



A continuación, se reiniciará la sesión de R, destacando que nuestro apartado de folder ahora estará colocado en sitio donde se generó el proyecto.



De esta manera, todo lo que creemos puede ser guardado en una misma sesión. Asimismo, podemos cargar archivos sin la necesidad de buscar su ubicación completa en el sistema. En el siguiente ejemplo se generará una carpeta llamada data, posteriormente crearemos un data set llamado iris_df con la información de un conjunto de datos de prueba llamdo iris y lo guardaremos en esta carpeta. Por último, llamaremos de nuevo este conjunto de datos.

```
# para este apartado trabajaremos con la libreria readr de tidyverse
# en caso de no estar instalada usaremos el siguiente comando
# install.packages('readr')
# ahora, llamaremos la libreria
 library(readr)

# creando carpeta data
dir.create('data')

# generando un dataset con la informacion de irir
iris_df <- iris

# guardando el dataset
write.csv(iris_df, 'data/iris.csv', row.names = FALSE)

# de esta manera se carga el conjunto de datos en la carpeta.
```

```r
# Para cargar lod datos utilizaremos la funcion read_csv() de readr
iris_df2 <- read_csv('data/iris.csv')
```

Antes de comenzar veremos brevemente los tipos de variables de trabajo más comúnes

```r
# variables con un solo elemento
x <- 2 # tipo numérico
y <- 'A' # tipo caracter
# los vectores los cuales son un tipo de representación en una sola dimensión
z <- c(1,2,3,4,5) # Tipo numérico
a <- c('a', 'b', 'c') # tipo caracter
# los dataframes, representaciones de datos en dos dimenciones
b <- data.frame(nombre = c('juan', 'pedro', 'dulcinea'),
           edad = c(12,15,13),
           sexo = c('hombre', 'hombre', 'mujer'))
# listas, elemntos que pueden contener cualquier tipo de variables
c <- list(x,y,z,a,b)
```

con esta introducción al manejo de proyectos podemos comenzar con la clase

Nota: una de las principales ventajas del lenguaje de programación R es el uso de vectores. La mayoría de los casos se limitarán a trabajar con ellos

Estadística descriptiva e inferencial

La estadística descriptiva suele ser el primer acercamiento analítico para entender el comportamiento de los datos. Visto de otra manera, se considera como un resumen de las variables más comunes de los datos. Asimismo, en esta categoría encontramos dos grupos, las medidas de tendencia central (media, moda, mediana) y las medidas de disperción (varianza, desviación típica). Además, se destacan otras medidas como el valor máximo, mínimo, y rango. Por mencionar algunos.

Por su parte, la estadística inferencial se encarga de generar predicciones desde la información presente en el conjunto de datos. Por ejemplo, definir las ventas del próximo mes a partir de las ventas del año pasado. Detallar la probabilidad de que ocurra un evento, dado la ocurrencia de otro. Entre estos podemos mencionar a los modelos lineales, series de tiempo, cadenas de Markov, solo por definir algunos ejemplos.

Medidas de tendencia central

Comenzaremos con las medidas de tendencia central. Media aritmética, mediana, moda

La media aritmética es el promedio de un conjunto de valores. Para calcularla se suman todos los valores y se divide sobre el total de elementos. En R, se puede calcular mediante la función mean(). A continuación, calcularemos el promedio de los valores de *Sepal.Length*, *Sepal.Width*, *Petal.Length*, y *Petal.Width* del conjunto de datos iris.

```r
print(mean(iris$Sepal.Length))
```

```
## [1] 5.843333
```

```r
print(mean(iris$Sepal.Width))
```

```
## [1] 3.057333
```

```r
print(mean(iris$Petal.Length))
```

```
## [1] 3.758
```

```r
print(mean(iris$Petal.Width))
```

```
## [1] 1.199333
```

Todos los caminos conducen a Roma. Ahora vamos a calcular el promedio acorde a la fórmula. Tal vez no sea la manera más rápida, pero ten en cuenta que cuando programes no existirá una sola respuesta correcta.

```
# primero vamos a sumar todos los valores de sepal length

suma <- sum(iris$Sepal.Length)

# ahora obtendremos el total de valores presentes en esta columna

valores <- length(iris$Sepal.Length)

# Por último, realizaremos la función descrita para la media aritmética

promedio <- suma/valores
print(promedio)
```

```
## [1] 5.843333
```

```
# si comparamos ambas funciones obtendremos el mismo resultado

print(promedio)
```

```
## [1] 5.843333
```

```
print(mean(iris$Sepal.Length))
```

```
## [1] 5.843333
```

Nota: en ambos casos trabajamos solo con vectores, dejando de lado el resto de la información y sus dimensiones

En ocasiones encontraremos conjuntos de datos con información faltante. Si utilizamos la función *mean()* de forma directa encontraremos un error

```
d <- c(1, 3, 4, 2, 2, 1, 4, 5, NA, 6, 5, 4, 4)
mean(d)
```

```
## [1] NA
```

Para ello podemos utilizar el argumento na.rm = TRUE, para eliminar el/los valores faltantes. Podemos acceder a los argumentos de las funciones mediante la función *help()*.

```
d <- c(1, 3, 4, 2, 2, 1, 4, 5, NA, 6, 5, 4, 4)
mean(d, na.rm = TRUE)
```

```
## [1] 3.416667
```

Nota: uno de los problemas de la media es su poca sensibilidad a valores extremos

La mediana es el valor que se encuentra en medio al ordenar un conjunto de datos. Este orden puede ser de manera ascendente o descendente. Cuando los datos tienden a la normalidad su valor es similar a la media aritmética. No obstante, cuando encotramos valores extremos, la mediana no sueleverse comprometida. Para obtener la mediana de un conjunto de datos se utiliza la función *median()*. A continuación realizaremos algunos ejemplos utilizando el set de datos iris.

```
print(median(iris$Sepal.Length))
```

```
## [1] 5.8
```

```
print(median(iris$Sepal.Width))
```

```
## [1] 3
```

```
print(median(iris$Petal.Length))
```

```
## [1] 4.35
```

```
print(median(iris$Petal.Width))
```

```
## [1] 1.3
```

Asimismo, si existen datos faltantes, es posible utilizar el argumento na.rm= TRUE

```
d <- c(1, 3, 4, 2, 2, 1, 4, 5, NA, 6, 5, 4, 4)
median(d, na.rm = TRUE)
```

```
## [1] 4
```

La moda es el valor con mayor repetición dentro de un conjunto de datos. Es decir, es aquel valor con la mayor frecuencia. Al igual que la mediana, si se aplica a un conjunto de datos que sigue la distribución normal, su valor será similar a la media aritmética. Dentro de la paquetería base de estadística en R no existe una función que permita calcular la moda. No obstante, descargaremos la paquetería *modeest* y utilizaremos la función mfv(). Para ello, instalaremos y llamaremos a esta paquetería en el flujo de trabajo.

```
# install.packages('modeest')
library(modeest)

e <- c(2,2,2,4,1,5,6,7,5,5,3,5,2,5,4,5,5,8,9)
print(mfv(e))
```

```
## [1] 5
```

Al igual que el resto de las funciones, al encontrar datos faltantes podemos omitirlos con el argumento na_rm = TRUE

```
f <- c(2,2,2,4,1,5,6,7,NA,5,3,5,2,NA,4,NA,NA,8,9)
print(mfv(f, na_rm = TRUE))
```

```
## [1] 2
```

Medidas de dispersión

Las medidas de dispersión ayudan a entender la distribución de los datos y como estos se comportan en relación al promedio. De esta manera, definimos que tan homogéneos o heterogéneos son los datos. Dentro de estas medidas encontramos a la varianza y la desviación estándar.

La varianza indica que tan dispersos se encuentran los datos en comparación al promedio. Cuanto mayor sea la varianza, existe una dispersión de los datos elevada. Para obtener la varianza de un conjunto de datos se utiliza la función *var()*.

```
print(var(iris$Sepal.Length))
```

```
## [1] 0.6856935
```

```
print(var(iris$Sepal.Width))
```

```
## [1] 0.1899794
```

No obstante, una de las desventajas de la varianza es que el resultado presente se encuentra en unidades cuadradas. Esto, al ser comparado con los datos de inicio genera confusión, por lo cual es necesario transformar este resultado mediante la raiz cuadrada. Es así, como se obtiene el grado de dispersión de los datos. La

desviación estándar o típica se genera con la función *sd()*, o al aplicar la función *sqrt()* (raíz cuadrada) sobre el resultado de la varianza.

```
print(sqrt(var(iris$Sepal.Length)))
```

```
## [1] 0.8280661
```

```
print(sd(iris$Sepal.Length))
```

```
## [1] 0.8280661
```

```
print(sd(iris$Sepal.Width))
```

```
## [1] 0.4358663
```

Cuantiles(Cuartiles, deciles, percentiles)

Los cuantiles dividen el conjunto de datos en partes iguales. Con el objetivo de determinar rangos en los cuales se encuentra cierto porcentaje de la información. En primer lugar, echaremos un vistazo a los cuartiles. Estos, dividen a la población en cuatro sectores, realizando tres cortes en específico. Siendo estos el 25%, 50% y 75%. Para hacer esto en un conjunto de datos utilizamos la función *quantile()*.

```
print(quantile(iris$Sepal.Length))
```

```
##   0%  25%  50%  75% 100%
##  4.3  5.1  5.8  6.4  7.9
```

```
print(quantile(iris$Sepal.Width))
```

```
##   0%  25%  50%  75% 100%
##  2.0  2.8  3.0  3.3  4.4
```

Los deciles realizan 9 cortes en el conjunto de datos, obteniendo 10 grupos en los cuales se encuentran distribuidos los datos. Al igual que en los cuartiles, se utiliza la función *quantile()*. Sin embargo, se agrega el argumento probs = para definir el número de cortes dentro del dataset. Estos cortes, se incluyen dentro de un vector acompañado de la función concatenar *c()*.

```
print(quantile(iris$Sepal.Length, probs = c(0, 0.1, 0.2, 0.3, 0.4, 0.5,
                                            0.6, 0.7, 0.8, 0.9, 1)))
```

```
##   0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
## 4.30 4.80 5.00 5.27 5.60 5.80 6.10 6.30 6.52 6.90 7.90
```

Por su parte, los percentiles realizan 99 cortes en el grupo de datos. Proporcionando 100 grupos en los cuales se distribuye la información. Al igual que los deciles utilizamos la función *quantile()*, acompañado del argumento probs = . En el siguiente ejemplo, se muestra el límite en el cual se encuentran el 77% de los datos para la longitud de sépalo.

```
print(quantile(iris$Sepal.Length, probs = 0.77))
```

```
##   77%
## 6.473
```

De la misma manera, es posible establecer rangos para determinar los valores que abarca cierto porcentaje de la porblación. El siguiente ejemplo muestra el rango en el cual se encuentran el 33% y 66% de los datos.

```
print(quantile(iris$Sepal.Length, probs = c(0.33, 0.66)))
```

```
##   33%   66%
## 5.400 6.234
```

Nota: si solo queremos conocer un cuantil en específico se suele omitir la función *c()*. Por otra parte, al utilizar un vector, obligatorio utilizar la función concatenar

No obstante, podemos echar un vistazo rápido sobre el comportamiento de los datos mediante la función *glimpse()* de la paquetería *dplyr* y la función *summary()* de la paquetería *base*

Para utilizar la función *glimpse()* debemos tener instalada la paquetería *dply* y posteriormente, llamarla al flujo de trabajo

```
# install.packages('dplyr')
library(dplyr)
glimpse(iris)
```

```
## Rows: 150
## Columns: 5
## $ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.~
## $ Sepal.Width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.~
## $ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.~
## $ Petal.Width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.~
## $ Species      <fct> setosa, setosa, setosa, setosa, setosa, setosa, setosa, s~
```

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

Podemos utilizar la función *summary()* de manera global o seleccionar una variable en específico

```
summary(iris$Sepal.Length)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.300   5.100   5.800   5.843   6.400   7.900
```

Otras medidas

La función *min()* regresa el valor mínimo de un vector determinado. De esta manera, es posible determinar el valor más pequeño descrito en nuestro conjunto de datos.

```
print(min(iris$Sepal.Length))
```

```
## [1] 4.3
```

```
print(min(iris$Sepal.Width))
```

```
## [1] 2
```

Por su parte, la función *max()* muestra el valor máximo de un vector.

```
print(max(iris$Sepal.Length))
```

```
## [1] 7.9
```

```
print(max(iris$Sepal.Width))
```

```
## [1] 4.4
```

Por último, el rango muestra la diferencia presente entre el valor máximo y el valor mínimo de un conjunto de datos. La función *range()* genera un vector con el valor mínimo y el valor máximo. Este resultado puede ser utilizado para encontrar el valor del rango. El siguiente código muestra un ejemplo de cómo realizar este procedimiento.

```
print(range(iris$Sepal.Length)) # vector con valor mínimo y máximo
```

```
## [1] 4.3 7.9
```

```
g <- range(iris$Sepal.Length)
rango <- g[2]-g[1]
print(rango)
```

```
## [1] 3.6
```

Otra manera de encontrar el rango es utilizando las funciones *max()* y *min()*

```
rango2 <- max(iris$Sepal.Length) - min(iris$Sepal.Length)
print(rango2)
```

```
## [1] 3.6
```

TAREA

Genera un proyecto para el curso. En este agregarás las siguientes carpetas: * scripts * images * raw_data * clean_data

Posteriormente, guarda las anotaciones realizadas en este curso dentro de la carpeta scripts