Notas de clase del curso de introducción a Data Science

Diego Kozlowski y Natsumi Shokida

2019-08-23

Contents

Chapter 1

Temario

- **clase 1**: Introducción al entorno R
- **clase 2**: Tidyverse. (limpieza y organización de datos)
- **clase 3**: Visualización de la información
- **clase 4**: Estadística descriptiva
- **clase 5**:
- **clase 6**:
- **clase 7**:
- **clase 8**:
- **clase 9**:
- **clase 10**:

1.0.0.1 Librerias a instalar

```
install.packages(c("tidyverse","openxlsx",'ggplot2','ggthemes', 'ggrepel','ggalt','kableExtra','C
```

Chapter 2

Introducción a R

2.1 Explicación

2.1.1 ¿Que es R?

- Lenguaje para el procesamiento y análisis estadístico de datos
- Software Libre
- Sintaxis Básica: R base
- Sintaxis incremental[1]: El lenguaje se va ampliando por aportes de Universidades, investigadores/as, usuarios/as y empresas privadas, organizados en librerías (o paquetes)
- Comunidad web muy grande para realizar preguntas y despejar dudas.
- Graficos con calidad de publicación

El *entorno* más cómodo para utilizar el *lenguaje* **R** es el *programa* **R studio**

- Rstudio es una empresa que produce productos asociados al lenguaje R, como el programa sobre el que corremos los comandos, y extensiones del lenguaje (librerías).
- El programa es *gratuito* y se puede bajar de la página oficial

2.1.2 Lógica sintáctica.

2.1.2.1 Definición de objetos

Los **Objetos/Elementos** constituyen la categoría escencial del R. De hecho, todo en R es un objeto, y se almacena con un nombre específico que **no debe poseer espacios**. Un número, un vector, una función, la progresión de letras del abecedario, una base de datos, un gráfico, constituyen para R objetos de

[1] Más allá de los comandos elementales, comandos más sofisticados tienen muchas versiones, y algunas quedan en desuso en el tiempo.

Figure 2.1: https://cran.r-project.org/

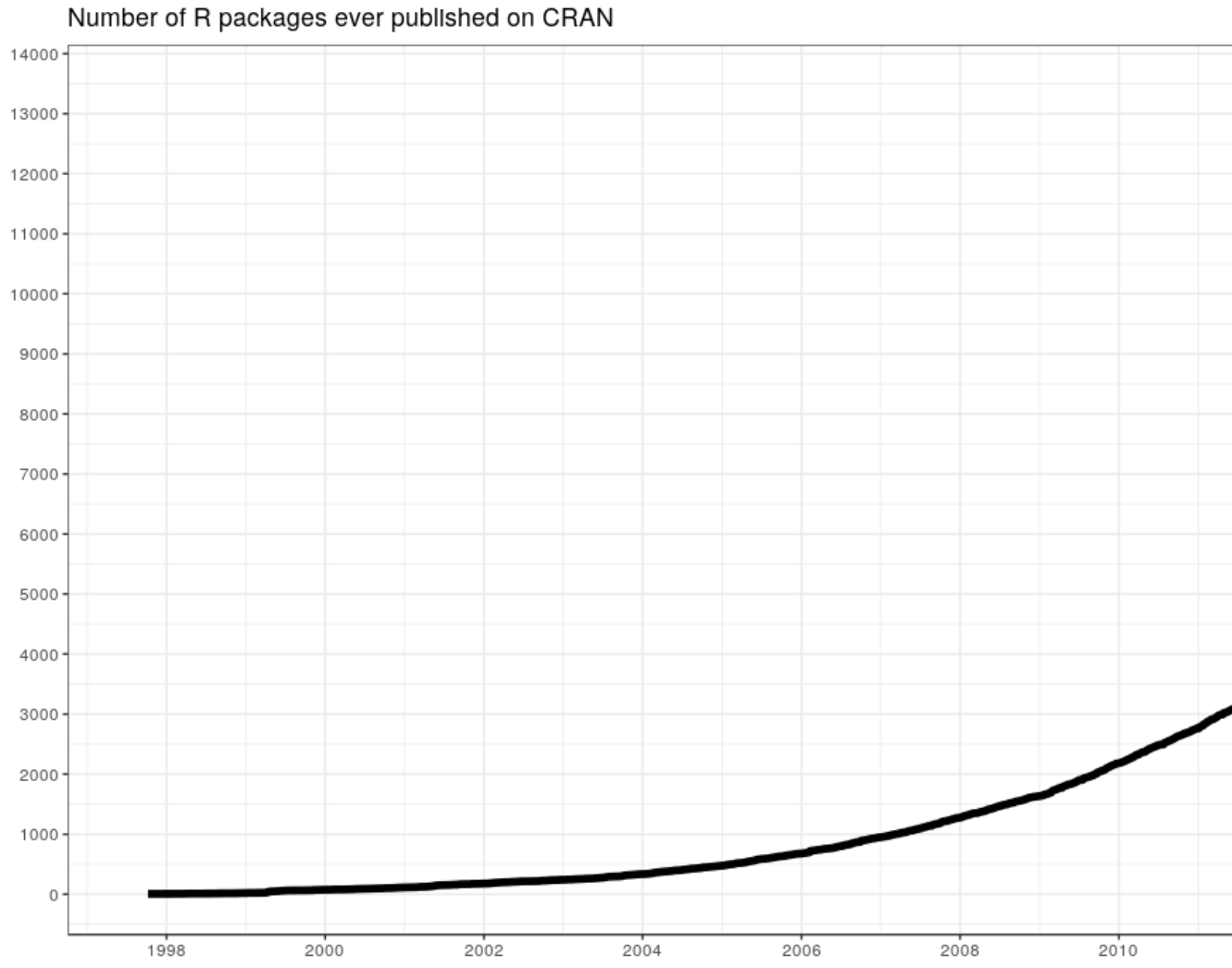


Figure 2.2: fuente: https://gist.github.com/daroczig/3cf06d6db4be2bbe3368

Figure 2.3: https://www.rstudio.com/

Rstudio.bb

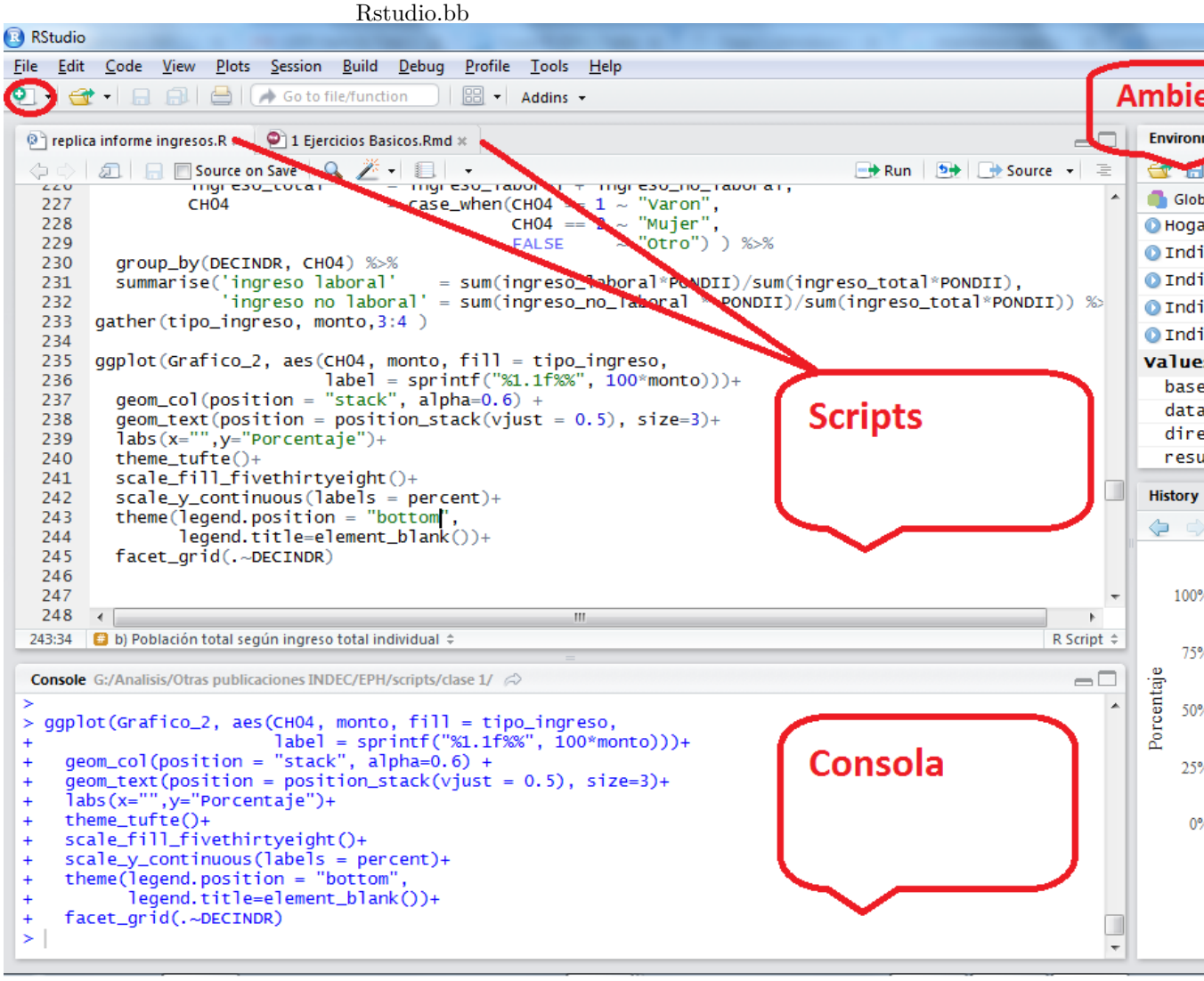


Figure 2.4: Pantalla Rstudio

distinto tipo. Los objetos que vamos creando a medida que trabajamos pueden visualizarse en la panel derecho superior de la pantalla.

El operador `<-` sirve para definir un objeto. **A la izquierda** del `<-` debe ubicarse el nombre que tomará el elemento a crear. **Del lado derecho** debe ir la definición del mismo

```
A <- 1
```

Al definir un elemento, el mismo queda guardado en el ambiente del programa, y podrá ser utilizado posteriormente para observar su contenido o para realizar una operación con el mismo

```
A
```

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

```
A+6
```

```
## [1] 7
```

Al correr una linea con el nombre del objeto, la consola del programa nos muestra su contenido. Entre Corchetes Observamos el número de orden del elemento en cuestión

El operador `=` es **equivalente** a `<-`, pero en la práctica no se utiliza para la definición de objetos.

```
B = 2
B
```

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

`<-` es un operador **Unidireccional**, es decir que:
`A <- B` implica que **A** va tomar como valor el contenido del objeto **B**, y no al revés.

```
A <- B
A   #Ahora A toma el valor de B, y B continua conservando el mismo valor
```

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

```
B
```

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

2.1.3 R base

Con *R base* nos referimos a los comandos básicos que vienen incorporados en el R, sin necesidad de cargar librerías.

2.1.3.1 Operadores lógicos:

- >

- >=
- <
- <=
- ==
- !=

```
#Redefinimos los valores A y B
A <-  10
B  <-  20
#Realizamos comparaciones lógicas

A >  B
```

```
## [1] FALSE
```

```
A >= B
```

```
## [1] FALSE
```

```
A <  B
```

```
## [1] TRUE
```

```
A <= B
```

```
## [1] TRUE
```

```
A == B
```

```
## [1] FALSE
```

```
A != B
```

```
## [1] TRUE
```

```
C <- A != B
C
```

```
## [1] TRUE
```

Como muestra el último ejemplo, el resultado de una operación lógica puede almacenarse como el valor de un objeto.

2.1.3.2 Operadores aritméticos:

```
#suma
A <- 5+6
A
```

```
## [1] 11
```

```
#Resta
B <- 6-8
B
```

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

```
#cociente
C <- 6/2.5
C
```

```
## [1] 2.4
```

```
#multiplicacion
D <- 6*2.5
D
```

```
## [1] 15
```

2.1.3.3 Funciones:

Las funciones son series de procedimientos estandarizados, que toman como imput determinados argumentos a fijar por el usuario, y devuelven un resultado acorde a la aplicación de dichos procedimientos. Su lógica de funcionamiento es:
`funcion(argumento1 = arg1, argumento2 = arg2)`

A lo largo del curso iremos viendo numerosas funciones, según lo requieran los distintos ejercicios. Sin embargo, veamos ahora algunos ejemplos para comprender su funcionamiento:

- paste() : concatena una serie de caracteres, indicando por última instancia como separar a cada uno de ellos

- paste0(): concatena una serie de caracteres sin separar
- sum(): suma de todos los elementos de un vector

- mean() promedio aritmético de todos los elementos de un vector

```
paste("Pega","estas",4,"palabras", sep = " ")
```

```
## [1] "Pega estas 4 palabras"
```

```
#Puedo concatenar caracteres almacenados en objetos
paste(A,B,C,sep = "**")
```

```
## [1] "11**-2**2.4"
```

```
# Paste0 pega los caracteres sin separador
paste0(A,B,C)
```

```
## [1] "11-22.4"
```

```
1:5
```

```
## [1] 1 2 3 4 5
```

```r
sum(1:5)
```

```
## [1] 15
```

```r
mean(1:5,na.rm = TRUE)
```

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

2.1.3.4 Caracteres especiales

- R es sensible a mayúsculas y minúsculas, tanto para los nombres de las variables, como para las funciones y parámetros.
- Los **espacios en blanco** y los **carriage return** (*enter*) no son considerados por el lenguaje. Los podemos aprovechar para emprolijar el código y que la lectura sea más simple[2].
- El **numeral #** se utiliza para hacer comentarios. Todo lo que se escribe después del # no es interpretado por R. Se debe utilizar un # por cada línea de código que se desea anular
- Los **corchetes** `[]` se utilizan para acceder a un objeto:
  - en un vector[n° orden]
  - en una tabla[fila, columna]
  - en una lista[n° elemento]
- el signo **$** también es un método de acceso. Particularmente, en los dataframes, nos permitira acceder a una determinada columna de una tabla
- Los **paréntesis**`()` se utilizan en las funciones para definir los parámetros.
- Las **comas** **,** se utilizan para separar los parametros al interior de una función.

2.1.4 Objetos:

Existen un gran cantidad de objetos distintos en R, en lo que resepcta al curso trabajaremos principalmente con 3 de ellos:

- Valores
- Vectores
- Data Frames
- Listas

2.1.4.1 Valores

Los valores y vectores pueden ser a su vez de distintas *clases*:

Numeric

[2] veremos que existen ciertas excepciones con algunos paquetes más adelante.

```
A <-  1
class(A)
```

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

Character

```
A <-  paste('Soy', 'una', 'concatenación', 'de', 'caracteres', sep = " ")
A
```

```
## [1] "Soy una concatenación de caracteres"
```

```
class(A)
```

```
## [1] "character"
```

Factor

```
A <- factor("Soy un factor, con niveles fijos")
class(A)
```

```
## [1] "factor"
```

La diferencia entre un *character* y un *factor* es que el último tiene solo algunos valores permitidos (levels), con un orden interno predefinido (el cual ,por ejemplo, se respetará a la hora de realizar un gráfico)

2.1.4.2 Vectores

Para crear un **vector** utilizamos el comando `c()`, de combinar.

```
C <- c(1, 3, 4)
C
```

```
## [1] 1 3 4
```

sumarle 2 a cada elemento del **vector** anterior

```
C <- C + 2
C
```

```
## [1] 3 5 6
```

sumarle 1 al primer elemento, 2 al segundo, y 3 al tercer elemento del **vector** anterior

```
D <- C + 1:3 #esto es equivalente a hacer 3+1, 5+2, 6+9
D
```

```
## [1] 4 7 9
```

`1:3` significa que queremos todos los números enteros desde 1 hasta 3.

crear un **vector** que contenga las palabras: “Carlos”,“Federico”,“Pedro”

```
E <- c("Carlos","Federico","Pedro")
E
```

```
## [1] "Carlos"   "Federico" "Pedro"
```

para acceder a algún elemento del vector, podemos buscarlo por su número de orden, entre [ ]

```
 E[2]
```

```
## [1] "Federico"
```

Si nos interesa almacenar dicho valor, al buscarlo lo asignamos a un nuevo objeto, dandole el nombre que deseemos

```
elemento2 <-  E[2]
```

```
elemento2
```

```
## [1] "Federico"
```

para **borrar** un objeto del ambiente de trabajo, utilizamos el comando *rm()*

```
rm(elemento2)
elemento2
```

```
## Error in eval(expr, envir, enclos): object 'elemento2' not found
```

También podemos cambiar el texto del segundo elemento de E, por el texto "Pablo"

```
E[2] <- "Pablo"
E
```

```
## [1] "Carlos" "Pablo"  "Pedro"
```

2.1.5 Data Frames

Un Data Frame es una tabla de datos, donde cada columna representa una variable, y cada fila una observación.

Este objeto suele ser central en el proceso de trabajo, y suele ser la forma en que se cargan datos externos para trabajar en el ambiente de R, y en que se exportan los resultados de nuestros trabajo.

También Se puede crear como la combinación de N vectores de igual tamaño. Por ejemplo, tomamos algunos valores del Indice de salarios

```
INDICE  <- c(100,   100,   100,
             101.8, 101.2, 100.73,
             102.9, 102.4, 103.2)

FECHA  <-  c("Oct-16", "Oct-16", "Oct-16",
```

```
             "Nov-16", "Nov-16", "Nov-16",
             "Dic-16", "Dic-16", "Dic-16")


GRUPO  <-  c("Privado_Registrado","Público","Privado_No_Registrado",
             "Privado_Registrado","Público","Privado_No_Registrado",
             "Privado_Registrado","Público","Privado_No_Registrado")


Datos <- data.frame(INDICE, FECHA, GRUPO)
Datos
```

```
##   INDICE  FECHA                 GRUPO
## 1 100.00 Oct-16    Privado_Registrado
## 2 100.00 Oct-16               Público
## 3 100.00 Oct-16 Privado_No_Registrado
## 4 101.80 Nov-16    Privado_Registrado
## 5 101.20 Nov-16               Público
## 6 100.73 Nov-16 Privado_No_Registrado
## 7 102.90 Dic-16    Privado_Registrado
## 8 102.40 Dic-16               Público
## 9 103.20 Dic-16 Privado_No_Registrado
```

Tal como en un **vector** se ubica a los elementos mediante `[ ]`, en un **dataframe** se obtienen sus elementos de la forma **`[fila, columna]`**.

Otra opción es especificar la columna, mediante el operador **`$`**, y luego seleccionar dentro de esa columna el registro deseado mediante el número de orden.

```
Datos$FECHA
```

```
## [1] Oct-16 Oct-16 Oct-16 Nov-16 Nov-16 Nov-16 Dic-16 Dic-16 Dic-16
## Levels: Dic-16 Nov-16 Oct-16
```

```
Datos[3,2]
```

```
## [1] Oct-16
## Levels: Dic-16 Nov-16 Oct-16
```

```
Datos$FECHA[3]
```

```
## [1] Oct-16
## Levels: Dic-16 Nov-16 Oct-16
```

¿que pasa si hacemos `Datos$FECHA[3,2]` ?

```
Datos$FECHA[3,2]
```

```
## Error in `[.default`(Datos$FECHA, 3, 2): incorrect number of dimensions
```

Nótese que el último comando tiene un número incorrecto de dimensiones, porque estamos refiriendonos 2 veces a la columna FECHA.

Acorde a lo visto anteriormente, el acceso a los **dataframes** mediante `[ ]`, puede utilizarse para realizar filtros sobre la base, especificando una condición para las filas. Por ejemplo, puedo utilizar los `[ ]` para conservar del **dataframe** `Datos` unicamente los registros con fecha de Diciembre 2016:

```
Datos[Datos$FECHA=="Dic-16",]
```

```
##   INDICE  FECHA                 GRUPO
## 7  102.9 Dic-16    Privado_Registrado
## 8  102.4 Dic-16               Público
## 9  103.2 Dic-16 Privado_No_Registrado
```

La lógica del paso anterior sería: Accedo al dataframe `Datos`, pidiendo únicamente conservar las filas (por eso la condición se ubica a la *izquierda* de la `,`) que cumplan el requisito de pertenecer a la categoría **"Dic-16"** de la variable **FECHA**.
Aún más, podría aplicar el filtro y al mismo tiempo identificar una variable de interés para luego realizar un cálculo sobre aquella. Por ejemplo, podría calcular la media de los indices en el mes de Diciembre.

```
###Por separado
Indices_Dic <- Datos$INDICE[Datos$FECHA=="Dic-16"]
Indices_Dic
```

```
## [1] 102.9 102.4 103.2
```

```
mean(Indices_Dic)
```

```
## [1] 102.8333
```

```
### Todo junto
mean(Datos$INDICE[Datos$FECHA=="Dic-16"])
```

```
## [1] 102.8333
```

La lógica de esta sintaxis sería: "Me quedó con la variable **INDICE**, cuando la variable FECHA sea igual a **"Dic-16"**, luego calculo la media de dichos valores"

2.1.6 Listas

Contienen una concatenación de objetos de cualquier tipo. Así como un vector contiene valores, un dataframe contiene vectores, una lista puede contener dataframes, pero también vectores, o valores, y *todo ello a la vez*

```
superlista <- list(A,B,C,D,E,FECHA, DF = Datos, INDICE, GRUPO)
superlista
```

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

```
## [1] Soy un factor, con niveles fijos
## Levels: Soy un factor, con niveles fijos
##
## [[2]]
## [1] -2
##
## [[3]]
## [1] 3 5 6
##
## [[4]]
## [1] 4 7 9
##
## [[5]]
## [1] "Carlos" "Pablo"  "Pedro"
##
## [[6]]
## [1] "Oct-16" "Oct-16" "Oct-16" "Nov-16" "Nov-16" "Nov-16" "Dic-16" "Dic-16"
## [9] "Dic-16"
##
## $DF
##   INDICE  FECHA                 GRUPO
## 1 100.00 Oct-16    Privado_Registrado
## 2 100.00 Oct-16               Público
## 3 100.00 Oct-16 Privado_No_Registrado
## 4 101.80 Nov-16    Privado_Registrado
## 5 101.20 Nov-16               Público
## 6 100.73 Nov-16 Privado_No_Registrado
## 7 102.90 Dic-16    Privado_Registrado
## 8 102.40 Dic-16               Público
## 9 103.20 Dic-16 Privado_No_Registrado
##
## [[8]]
## [1] 100.00 100.00 100.00 101.80 101.20 100.73 102.90 102.40 103.20
##
## [[9]]
## [1] "Privado_Registrado"    "Público"               "Privado_No_Registrado"
## [4] "Privado_Registrado"    "Público"               "Privado_No_Registrado"
## [7] "Privado_Registrado"    "Público"               "Privado_No_Registrado"
```

Para acceder un elemento de una lista, podemos utilizar el operador **$**, que se puede usar a su vez de forma iterativa

```
superlista$DF$FECHA[2]
```

```
## [1] Oct-16
## Levels: Dic-16 Nov-16 Oct-16
```

2.1.7 Ambientes de trabajo

Hay algunas cosas que tenemos que tener en cuenta respecto del orden del ambiente en el que trabajamos:

- Working Directory: El directorio de trabajo, pueden ver el suyo con `getwd()`, es *hacia donde apunta el código*, por ejemplo, si quieren leer un archivo, la ruta del archivo tiene que estar explicitada como el recorrido desde el Working Directory.
- Environment: Esto engloba tanto la información que tenemos cargada en *Data* y *Values*, como las librerías que tenemos cargadas mientras trabajamos.

Es importante que mantengamos bien delimitadas estas cosas entre diferentes trabajos, sino:

1. El directorio queda referido a un lugar específico en nuestra computadora.

- Si se lo compartimos a otro **se rompe**
- Si cambiamos de computadora **se rompe**
- Si lo cambiamos de lugar **se rompe**
- Si primero abrimos otro script **se rompe**

2. Tenemos mezclados resultados de diferentes trabajos:

- Nunca sabemos si esa variable/tabla/lista se creo en ese script y no otro
- Perdemos espacio de la memoria
- No estamos seguros de que el script cargue todas las librerías que necesita

Rstudio tiene una herramienta muy útil de trabajo que son los **proyectos**. Estos permiten mantener un ambiente de trabajo delimitado por cada uno de nuestros trabajos. Es decir:

- El directorio de trabajo se refiere a donde esta ubicado el archivo .Rproj
- El Environment es específico de nuestro proyecto.

Un proyecto no es un sólo script, sino toda una carpeta de trabajo.

Para crearlo, vamos al logo de nuevo projecto (Arriba a la izquierda de la panatalla), y elegimos la carpeta de trabajo.

2.1.8 Tipos de archivos de R

- **Script**: Es un archivo de texto plano, donde podemos poner el código que utilizamos para preservarlo
- **Rnotebook**: También sirve para guardar el código, pero a diferencia de los scripts, se puede compilar, e intercalar código con resultados (este archivo es un rnotebook)
- **Rproject**: Es un archivo que define la metadata del proyecto
- **RDS y Rdata**: Dos formatos de archivos propios de R para guardar datos.

Figure 2.5: logo Rpoject

2.2 Práctica Guiada

2.2.1 Instalación de paquetes complementarios al R Base

Hasta aquí hemos visto múltiples funciones que están contenidas dentro del lenguaje básico de R. Ahora bien, al tratarse de un software libre, los usuarios de R con más experiencia contribuyen sistemáticamente a expandir este lenguaje mediante la creación y actualización de **paquetes** complementarios. Lógicamente, los mismos no están incluidos en la instalación inicial del programa, pero podemos descargarlos e instalarlos al mismo tiempo con el siguiente comando:

```
install.packages("nombre_del_paquete")
```

Resulta recomendable **ejecutar este comando desde la consola** ya que solo necesitaremos correrlo una vez en nuestra computadora. Al ejecutar el mismo, se descargarán de la pagina de CRAN los archivos correspondientes al paquete hacia el directorio en donde hayamos instalado el programa. Típicamente los archivos se encontrarán en `C:\Program Files\R\R-3.5.0\library\`, siempre con la versión del programa correspondiente.
Una vez instalado el paquete, cada vez que abramos una nueva sesión de R y querramos utilizar el mismo debemos **cargarlo al ambiente de trabajo** mediante la siguiente función:

```
library(nombre_del_paquete)
```

Nótese que al cargar/activar el paquete no son necesarias las comillas.

2.2.2 Lectura y escritura de archivos

2.2.2.1 .csv y .txt

Hay **muchas** funciones para leer archivos de tipo *.txt* y *.csv*. La mayoría sólo cambia los parámetros que vienen por default.

Es importante tener en cuenta que una base de datos que proviene de archivos *.txt*, o *.csv* puede presentar diferencias en cuanto a los siguientes parametros:

- encabezado
- delimitador (`,`, tab, `;`)
- separador decimal

```
dataframe <- read.delim(file, header = TRUE, sep = "\t", quote = "\"", dec = ".", fill = TRUE, co
```

Ejemplo. Levantar la base de sueldos de funcionarios

En el parametro `file` tengo que especificar el nombre completo del archivo, incluyendo el directorio donde se encuentra. Lo más sencillo es abrir comillas, apretar `Tab` y se despliega el menú de las cosas que tenemos en el directorio de trabajo. Si queremos movernos hacia arriba, agregamos `../`

```
sueldos_funcionarios <- read.table(file = '../fuentes/sueldo_funcionarios_2019.csv',sep
sueldos_funcionarios[1:10,]
```

```
##             cuil anio mes funcionario_apellido funcionario_nombre
## 1  20-17692128-6 2019   1    RODRIGUEZ LARRETA    HORACIO ANTONIO
## 2  20-17735449-0 2019   1             SANTILLI        DIEGO CESAR
## 3  27-24483014-0 2019   1                ACUÑA      MARIA SOLEDAD
## 4  20-13872301-2 2019   1             ASTARLOA      GABRIEL MARIA
## 5  20-25641207-2 2019   1             AVOGADRO       ENRIQUE LUIS
## 6  27-13221055-7 2019   1            BOU PEREZ          ANA MARIA
## 7  27-13092400-5 2019   1                FREDA     MONICA BEATRIZ
## 8  20-17110752-1 2019   1         MACCHIAVELLI    EDUARDO ALBERTO
## 9  20-22293873-3 2019   1               MIGUEL       FELIPE OSCAR
## 10 20-14699669-9 2019   1               MOCCIA             FRANCO
##                                      repartición asignacion_por_cargo_i
## 1                                Jefe de Gobierno               197745.8
## 2                          Vicejefatura de Gobierno               197745.8
## 3               Ministerio de Educación e Innovación               224516.6
## 4  Procuración General de la Ciudad de Buenos Aires               224516.6
## 5                             Ministerio de Cultura               224516.6
## 6                               Ministerio de Salud               224516.6
## 7  Sindicatura General de la Ciudad de Buenos Aires               224516.6
## 8           Ministerio de Ambiente y Espacio Público               224516.6
## 9                  Jefatura de Gabinete de Ministros               224516.6
## 10     Ministerio de Desarrollo Urbano y Transporte               224516.6
##    aguinaldo_ii total_salario_bruto_i_._ii observaciones
## 1             0                   197745.8
## 2             0                   197745.8
## 3             0                   224516.6
## 4             0                   224516.6
## 5             0                   224516.6
## 6             0                   224516.6
## 7             0                   224516.6
## 8             0                   224516.6
## 9             0                   224516.6
## 10            0                   224516.6
```

Como puede observarse aquí, las bases individuales de la EPH cuentan con más de 58.000 registros y 177 variables. Al trabajar con bases de microdatos, resulta conveniente contar con algunos comandos para tener una mirada rápida de la base, antes de comenzar a realizar los procesamientos que deseemos.

Veamos algunos de ellos:

```
#View(individual_t117)
names(sueldos_funcionarios)
```

```
##  [1] "cuil"                       "anio"
##  [3] "mes"                        "funcionario_apellido"
##  [5] "funcionario_nombre"         "repartición"
##  [7] "asignacion_por_cargo_i"     "aguinaldo_ii"
##  [9] "total_salario_bruto_i_._ii" "observaciones"
```

```
summary(sueldos_funcionarios)
```

```
##             cuil         anio           mes       funcionario_apellido
##  20-13872301-2: 3   Min.   :2019   Min.   :1.00   ACUÑA     : 3
##  20-14699669-9: 3   1st Qu.:2019   1st Qu.:2.00   ASTARLOA  : 3
##  20-16891528-5: 3   Median :2019   Median :3.00   AVELLANEDA: 3
##  20-16891539-0: 3   Mean   :2019   Mean   :3.34   AVOGADRO  : 3
##  20-17110752-1: 3   3rd Qu.:2019   3rd Qu.:5.00   BENEGAS   : 3
##  20-17692128-6: 3   Max.   :2019   Max.   :6.00   BOU PEREZ : 3
##  (Other)      :76                                 (Other)   :76
##         funcionario_nombre
##   ANA MARIA       : 3
##   BRUNO GUIDO     : 3
##   CHRISTIAN       : 3
##   DIEGO CESAR     : 3
##   DIEGO HERNAN    : 3
##   EDUARDO ALBERTO: 3
##  (Other)         :76
##                                                        repartición
##  Consejo de los Derechos de Niñas, Niños y Adoles - Presidencia: 3
##  Ente de Turismo Ley Nº 2627                                    : 3
##  Jefatura de Gabinete de Ministros                              : 3
##  Jefe de Gobierno                                               : 3
##  Ministerio de Ambiente y Espacio Público                       : 3
##  Ministerio de Cultura                                          : 3
##  (Other)                                                        :76
##  asignacion_por_cargo_i  aguinaldo_ii    total_salario_bruto_i_._ii
##  Min.   :197746         Min.   :     0   Min.   :197746
##  1st Qu.:217520         1st Qu.:     0   1st Qu.:217805
##  Median :226866         Median :     0   Median :226866
##  Mean   :224718         Mean   : 14843   Mean   :239560
##  3rd Qu.:231168         3rd Qu.:     0   3rd Qu.:248033
##  Max.   :249662         Max.   :113433   Max.   :340300
##
##         observaciones
##                :93
##  baja 28/2/2019: 1
##
##
##
```

```
##
##
```

```
head(sueldos_funcionarios)[,1:5]
```

```
##            cuil anio mes funcionario_apellido funcionario_nombre
## 1 20-17692128-6 2019   1    RODRIGUEZ LARRETA    HORACIO ANTONIO
## 2 20-17735449-0 2019   1             SANTILLI        DIEGO CESAR
## 3 27-24483014-0 2019   1                ACUÑA      MARIA SOLEDAD
## 4 20-13872301-2 2019   1             ASTARLOA      GABRIEL MARIA
## 5 20-25641207-2 2019   1             AVOGADRO       ENRIQUE LUIS
## 6 27-13221055-7 2019   1            BOU PEREZ           ANA MARIA
```

2.2.2.2 Excel

Para leer y escribir archivos excel debemos utilizar los comandos que vienen con la librería openxlsx

```
# install.packages("openxlsx") # por única vez
library(openxlsx) #activamos la librería

#creamos una tabla cualquiera de prueba
x <- 1:10
y <- 11:20
tabla_de_R <- data.frame(x,y)

# escribimos el archivo
write.xlsx( x = tabla_de_R, file = "../resultados/archivo.xlsx",row.names = FALSE)
#Donde lo guardó? Hay un directorio por default en caso de que no hayamos definido alg

#getwd()

#Si queremos exportar multiples dataframes a un Excel, debemos armar previamente una l
Lista_a_exportar <- list("sueldos funcionarios" = sueldos_funcionarios,
                         "Tabla Numeros" = tabla_de_R)

write.xlsx( x = Lista_a_exportar, file = "../resultados/archivo_2_hojas.xlsx",row.names

#leemos el archivo especificando la ruta (o el directorio por default) y el nombre de
Indices_Salario <- read.xlsx(xlsxFile = "../resultados/archivo_2_hojas.xlsx",sheet = "s
#alternativamente podemos especificar el número de orden de la hoja que deseamos levan
Indices_Salario <- read.xlsx(xlsxFile = "../resultados/archivo_2_hojas.xlsx",sheet = 1)
Indices_Salario[1:10,]
```

```
##            cuil anio mes funcionario_apellido funcionario_nombre
## 1 20-17692128-6 2019   1    RODRIGUEZ LARRETA    HORACIO ANTONIO
## 2 20-17735449-0 2019   1             SANTILLI        DIEGO CESAR
```

```
## 3  27-24483014-0 2019   1               ACUÑA      MARIA SOLEDAD
## 4  20-13872301-2 2019   1             ASTARLOA      GABRIEL MARIA
## 5  20-25641207-2 2019   1             AVOGADRO       ENRIQUE LUIS
## 6  27-13221055-7 2019   1            BOU PEREZ          ANA MARIA
## 7  27-13092400-5 2019   1                FREDA     MONICA BEATRIZ
## 8  20-17110752-1 2019   1          MACCHIAVELLI    EDUARDO ALBERTO
## 9  20-22293873-3 2019   1               MIGUEL       FELIPE OSCAR
## 10 20-14699669-9 2019   1               MOCCIA             FRANCO
##                                        repartición asignacion_por_cargo_i
## 1                                  Jefe de Gobierno               197745.8
## 2                            Vicejefatura de Gobierno               197745.8
## 3                Ministerio de Educación e Innovación               224516.6
## 4  Procuración General de la Ciudad de Buenos Aires               224516.6
## 5                               Ministerio de Cultura               224516.6
## 6                                 Ministerio de Salud               224516.6
## 7  Sindicatura General de la Ciudad de Buenos Aires               224516.6
## 8           Ministerio de Ambiente y Espacio Público               224516.6
## 9                  Jefatura de Gabinete de Ministros               224516.6
## 10     Ministerio de Desarrollo Urbano y Transporte               224516.6
##    aguinaldo_ii total_salario_bruto_i_._ii observaciones
## 1             0                   197745.8
## 2             0                   197745.8
## 3             0                   224516.6
## 4             0                   224516.6
## 5             0                   224516.6
## 6             0                   224516.6
## 7             0                   224516.6
## 8             0                   224516.6
## 9             0                   224516.6
## 10            0                   224516.6
```

Chapter 3

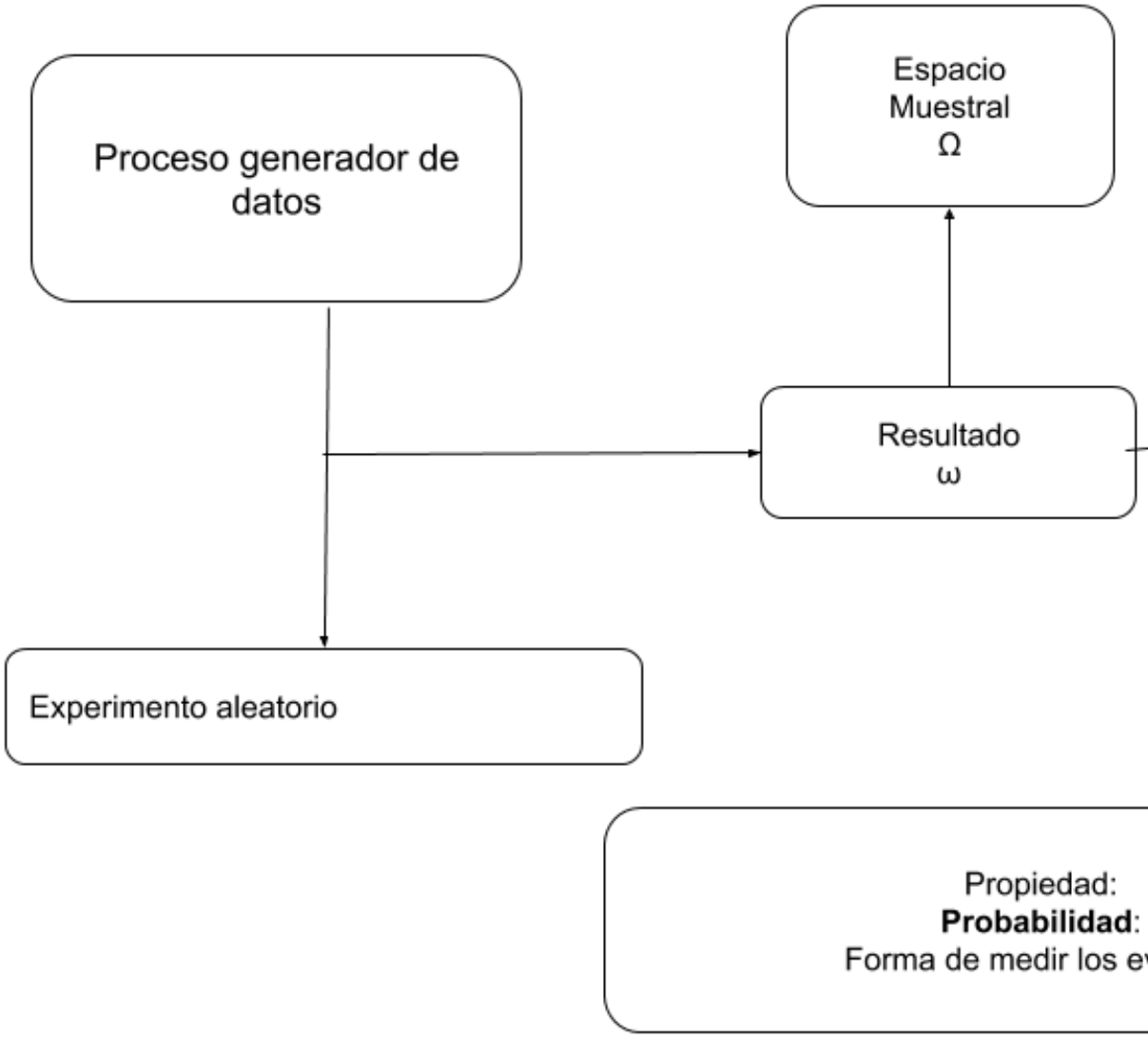
Probabilidad y Estadística

3.1 Explicación

3.1.1 Probabilidad

Previo a estudiar las herramientas de la estadística descriptiva, es necesario hacer un breve resumen de algunos conceptos fundamentales de probabilidad

3.1.1.1 Marco conceptual



- El análisis de las probabilidades parte de un **proceso generador de datos** entendido como cualquier fenómeno que produce algún tipo de información de forma sistemática.
- Cada iteración de este proceso produce información, que podemos interpretar como un **resultado**.
- Existe un conjunto de posibles resultados, que definimos como **espacio muestral**.
- Un **evento** es el conjunto de resultados ocurridos.

- En este marco, la **probabilidad** es un atributo de los eventos. Es la forma de medir los eventos tal que, siguiendo la definición moderna de probabilidad:

A) $P(A) \geq 0 \forall\ A \subseteq \Omega$
B) $P(\Omega) = 1$
C) $P(A \cup B) = P(A) + P(B)\ si\ A \cap B = \emptyset$

ejemplo, tiramos un dado y sale tres

- Espacio muestral: 1,2,3,4,5,6
- Resultado: 3
- Evento: impar (el conjunto 1,3,5)

3.1.1.2 Distribución de probabilidad

- La distribución de probabilidad hace referencia a los posibles valores teóricos de cada uno de los resultados pertenecientes al espacio muestral.
- Existen dos tipos de distribuciones, dependiendo si el espacio muestral es o no numerable.

3.1.1.2.1 Distribuciones discretas

Sigamos con el ejemplo de dado.

Podríamos definir la distribución de probabilidad, si no esta cargado, cómo:

```
## # A tibble: 6 x 2
##   valor probabilidad
##   <int> <chr>
## 1     1 1/6
## 2     2 1/6
## 3     3 1/6
## 4     4 1/6
## 5     5 1/6
## 6     6 1/6
```

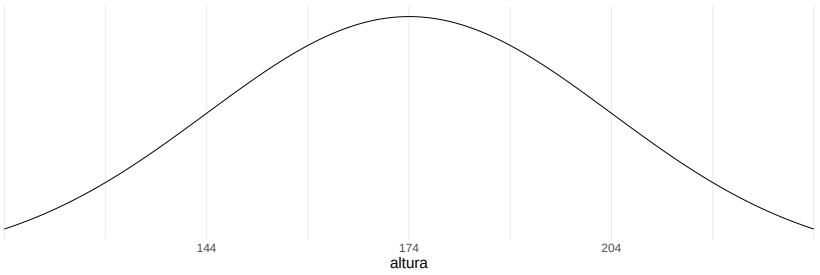
Cómo el conjunto de resultados posibles es acotado, podemos definirlo en una tabla, esta es una distribución *discreta*

3.1.1.2.2 Distribuciones continuas

¿Qué pasa cuando el conjunto de resultados posibles es tan grande que no se puede enumerar la probabilidad de cada caso?

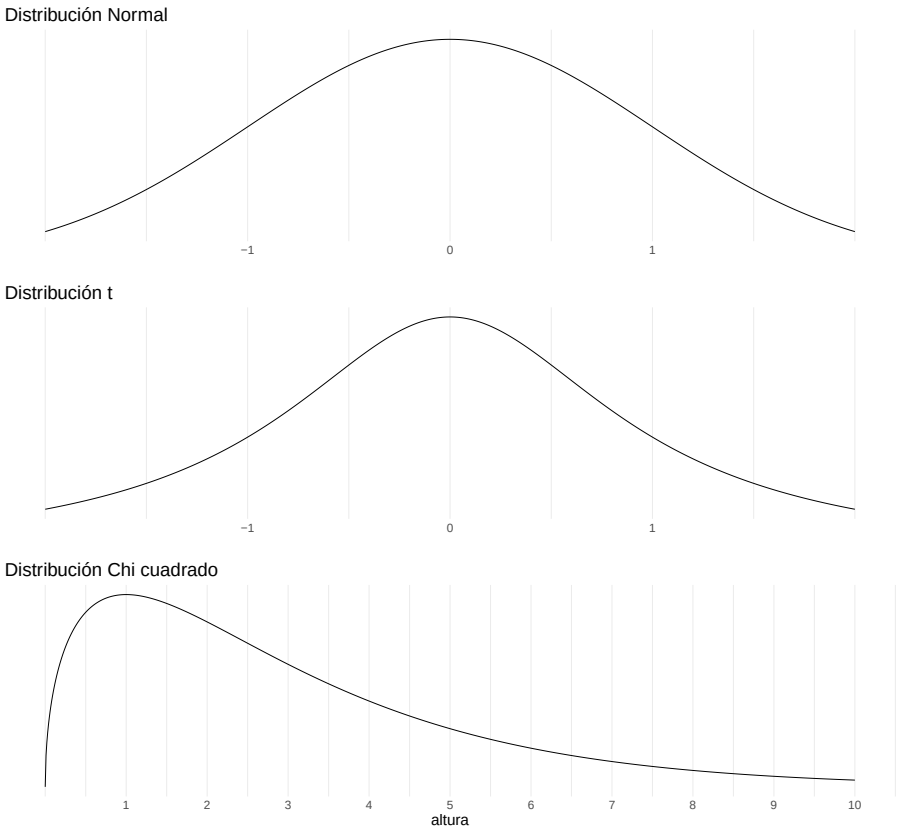
Si, por definición o por practicidad, no se puede enumerar cada caso, lo que tenemos es una **distribución continua**

Por ejemplo, la altura de la población



- En este caso, no podemos definir en una tabla la probabilidad de cada uno de los posibles valores. *de hecho, la probabilidad puntual es 0.*
- Sin embargo, sí podemos definir una *función de probabilidad*, la *densidad.*
- Según qué función utilicemos, cambiara la forma de la curva.

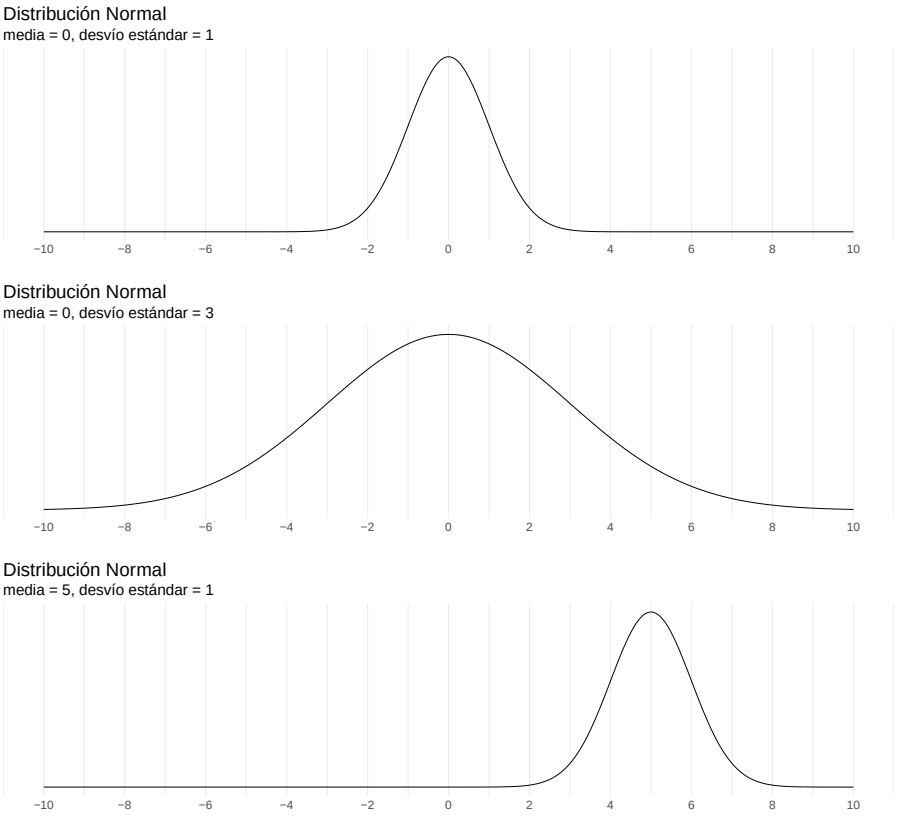
Por ejemplo:



Una distribución de probabilidad se **caracteriza** por sus *parámetros.*

- Por ejemplo, la distribución normal se caracteriza por su *esperanza* y su

varianza (o desvío estándar)

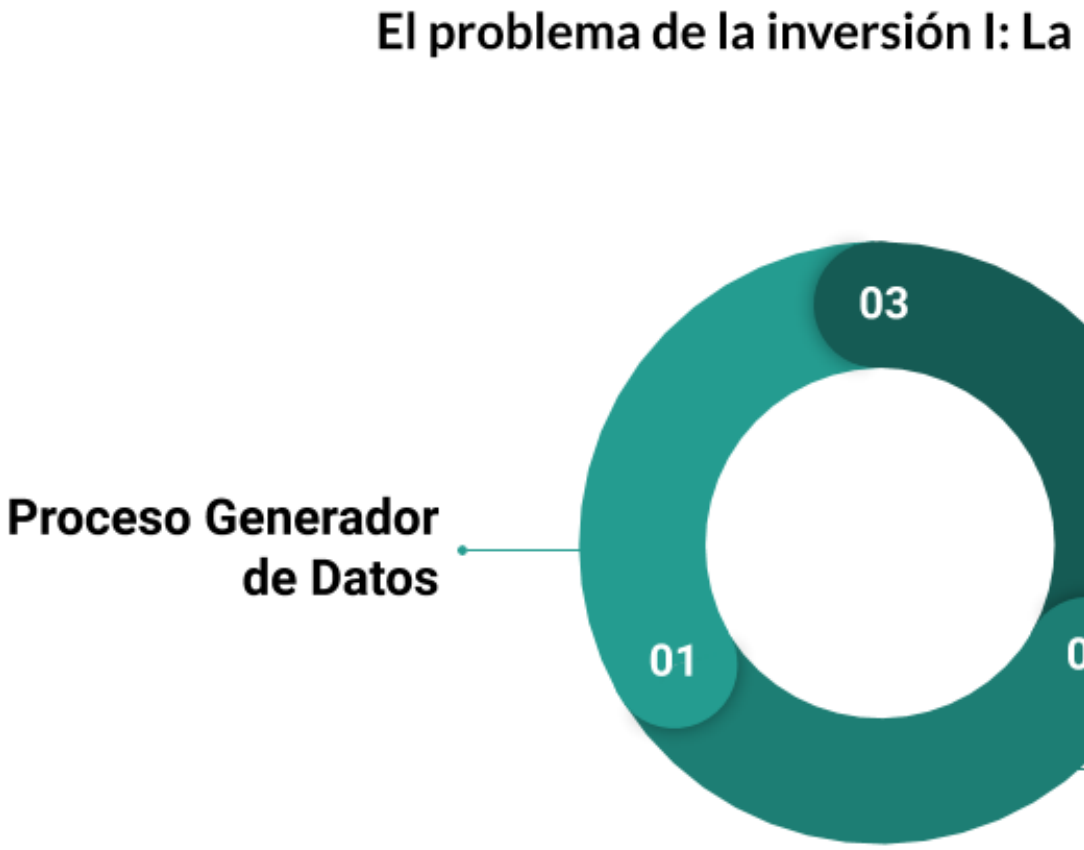


3.1.2 Estadística

3.1.2.1 El problema de la inversión

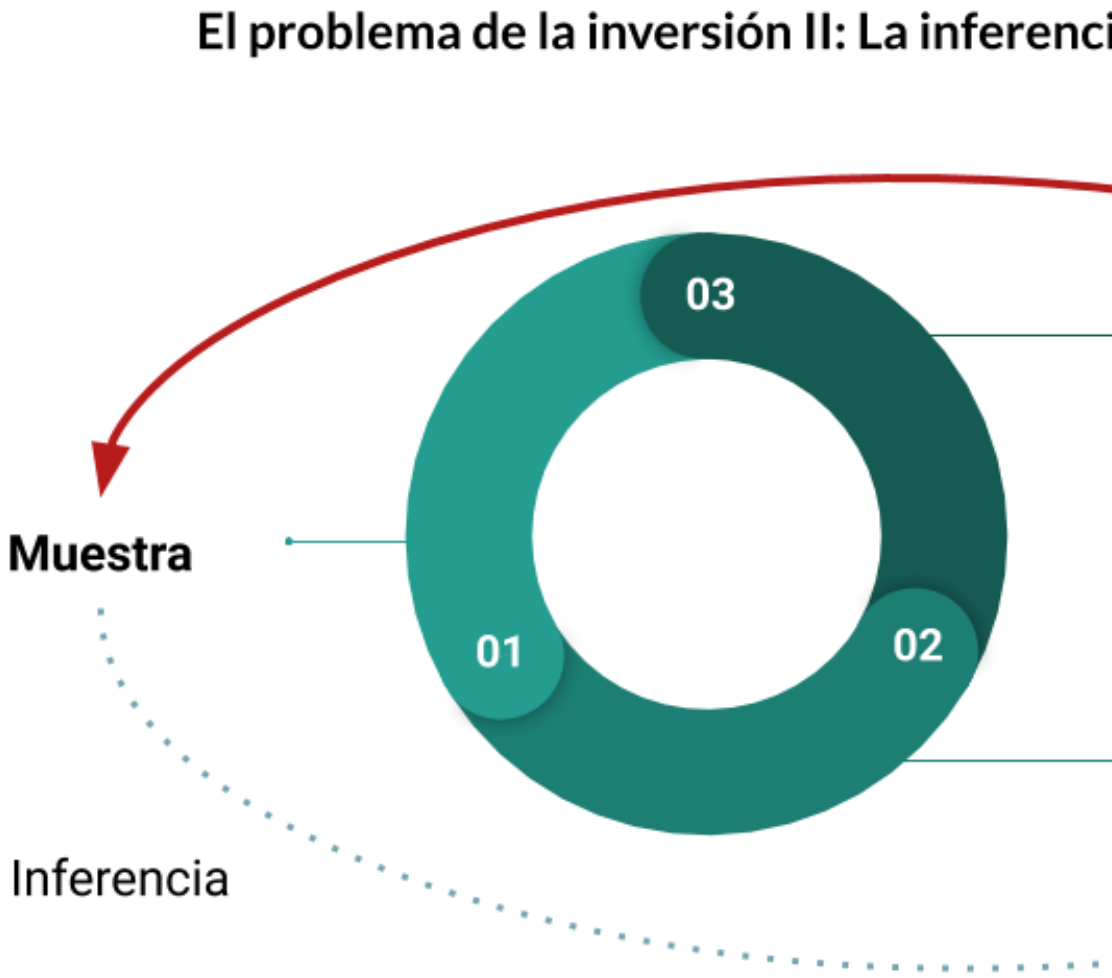
El problema de la probabilidad se podría pensar de la siguiente forma:

1. Vamos a partir de un **proceso generador de datos**
2. para calcular su **distribución de probabilidad**, los **parámetros** que caracterizan a ésta, y a partir de allí,
3. calcular la probabilidad de que, al tomar una **muestra**, tenga ciertos eventos.



El problema de la estadística es exactamente el contrario:

1. Partimos de una **muestra** para
2. inferir cuál es la **distribución de probabilidad**, y los **parámetros** que la caracterizan
3. para finalmente poder sacar conclusiones sobre el **proceso generador de datos**



3.1.2.1.1 Población y muestra

En este punto podemos hacer la distinción entre **población** y **muestra**

- **Población**: El universo en estudio. Puede ser:
  - finita: Los votantes en una elección.
  - infinita: El lanzamiento de una moneda.
- **Muestra**: subconjunto de n observaciones de una población.

Solemos utilizar las mayúsculas (N) para la población y las minúsculas (n) para

las muestras

3.1.2.1.2 Parámetros y Estimadores

- Como dijimos, los **parámetros** describen a la función de probabilidad. Por lo tanto hacen referencia a los atributos de la **población**. Podemos suponer que son *constantes*
- Un **estimador** es un estadístico (esto es, una función de la muestra) usado para estimar un parámetro desconocido de la población.

3.1.2.1.3 Ejemplo. La media

Esperanza o Media Poblacional:

$$\mu = E(x) = \sum_{i=1}^{N} x_i p(x_i)$$

Media muestral:

$$\bar{X} = \sum_{i=1}^{n} \frac{Xi}{n}$$

Como no puedo conocer μ, lo estimo mediante $\bar{X}$

3.1.2.2 Estimación puntual, Intervalos de confianza y Tests de hipótesis

- El estimador $\bar{X}$ nos devuelve un número. Esto es una inferencia de cuál creemos que es la media. Pero no es seguro de que esa sea realmente la media. Esto es lo que denominamos estimación puntual
- También podemos estimar un intervalo, dentro del cual consideramos que se encuentra la media poblacional. La ventaja de esta metodología es que podemos definir la probabilidad de que el parámetro poblacional realmente este dentro de este intervalo. Esto se conoce como **intervalos de confianza**
- Por su parte, también podemos calcular la probabilidad de que el parámetro poblacional sea mayor, menor o igual a un cierto valor. Esto es lo que se conoce como **test de hipótesis**.
- En el fondo, los intervalos de confianza y los tests de hipótesis se construyen de igual manera. Son funciones que se construyen a partir de los datos, que se comparan con distribuciones conocidas, *teóricas*.

3.1.2.2.1 Definición de los tests

- Los tests se construyen con dos hipótesis: La hipótesis nula H_0, y la hipótesis alternativa, H_1. Lo que buscamos es ver si *hay evidencia suficiente para rechazar la hipótesis nula.*

Por ejemplo, si queremos comprobar si la media poblacional, μ de una distribución es mayor a X_i, haremos un test con las siguientes hipótesis:

- $H_0 : \mu = X_i$
- $H_1 : \mu > X_i$

Si la evidencia es lo suficientemente fuerte, podremos rechazar la hipótesis H_0, *pero no afirmar la hipótesis* H_1

3.1.2.2.2 Significatividad en los tests

- Muchas veces decimos que algo es **"estadística mente significativo"**. Detrás de esto se encuentra un test de hipótesis que indica que hay una suficiente *significativdad estadística.*

- La *significatividad estadística*,representada con α, es la probabilidad de rechazar H_0 cuando en realidad es cierta. Por eso, cuanto más bajo el valor de α, más seguros estamos de no equivocarnos. Por lo general testeamos con valores de alpha de 1%, 5% y 10%, dependiendo del área de estudio

- El **p-valor** es _la mínima significatividad para la que rechazo el test. Es decir, cuanto más bajo es el p-valor, más seguros estamos de rechazar H_0

- El resultado de un test esta determinado por

  1. **La fuerza evidencia empírica**: Si nuestra duda es si la media poblacional es mayor a, digamos, 10. Y la media muestral es 11, no es es lo mismo que si es 100, 1000 o 10000.
  2. **El tamaño de la muestra**: En las fórmulas que definen los test siempre juega el tamaño de la muestra: cuanto más grande es, más seguros estamos de que el resultado no es producto del mero azar.
  3. **La veracidad de los supuestos**: Otra cosa importante es que los test asumen ciertas cosas:

  – Normalidad en los datos.
  – Que conocemos algún otro parámetro de la distribución, como la varianza.
  – Que los datos son independientes entre sí,
  – Etc.
  Cada Test tiene sus propios supuestos. Por eso a veces luego de hacer un test, hay que hacer otros tests para validar que los supuestos se cumplen.

- Lo primero, la fuerza de la evidencia, es lo que más nos importa, y no hay mucho por hacer.

- El tamaño de la muestra es un problema, porque si la muestra es muy chica, entonces podemos no llegar a conclusiones significativas aunque sí ocurra aquello que queríamos probar.

- Sin embargo, el verdadero problema en *La era del big data* es que tenemos muestras demasiado grandes, por lo que cualquier test, por más mínima que sea la diferencia, puede dar significativo.

  Por ejemplo, podemos decir que la altura promedio en Argentina es 1,74. Pero si hacemos un test, utilizando como muestra 40 millones de personas, vamos a rechazar que ese es el valor, porque en realidad es 1,74010010. En términos de lo que nos puede interesar, 1,74 sería válido, pero estadísticamente rechazaríamos.

- Finalmente, según la información que tengamos de la población y cual es el problema que queremos resolver, vamos a tener que utilizar distintos tipos de tests. La cantidad de tests posibles es ENORME, y escapa al contenido de este curso, así como sus fórmulas. A modo de ejemplo, les dejamos el siguiente machete:

3.1.3 Algunos estimadores importantes

3.1.3.1 Medidas de centralidad

- **Media**

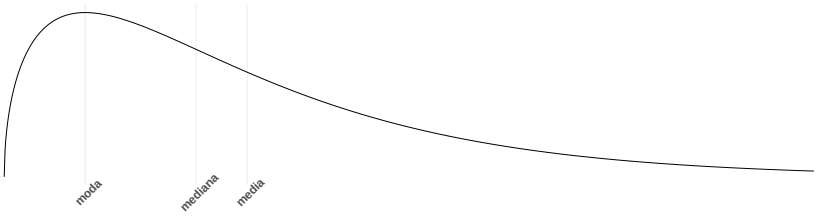
$$\bar{X} = \sum_{i=1}^{n} \frac{Xi}{n}$$

- **Mediana**:

Es el valor que parte la distribución a la mitad

- **Moda**

La moda es el valor más frecuente de la distribución



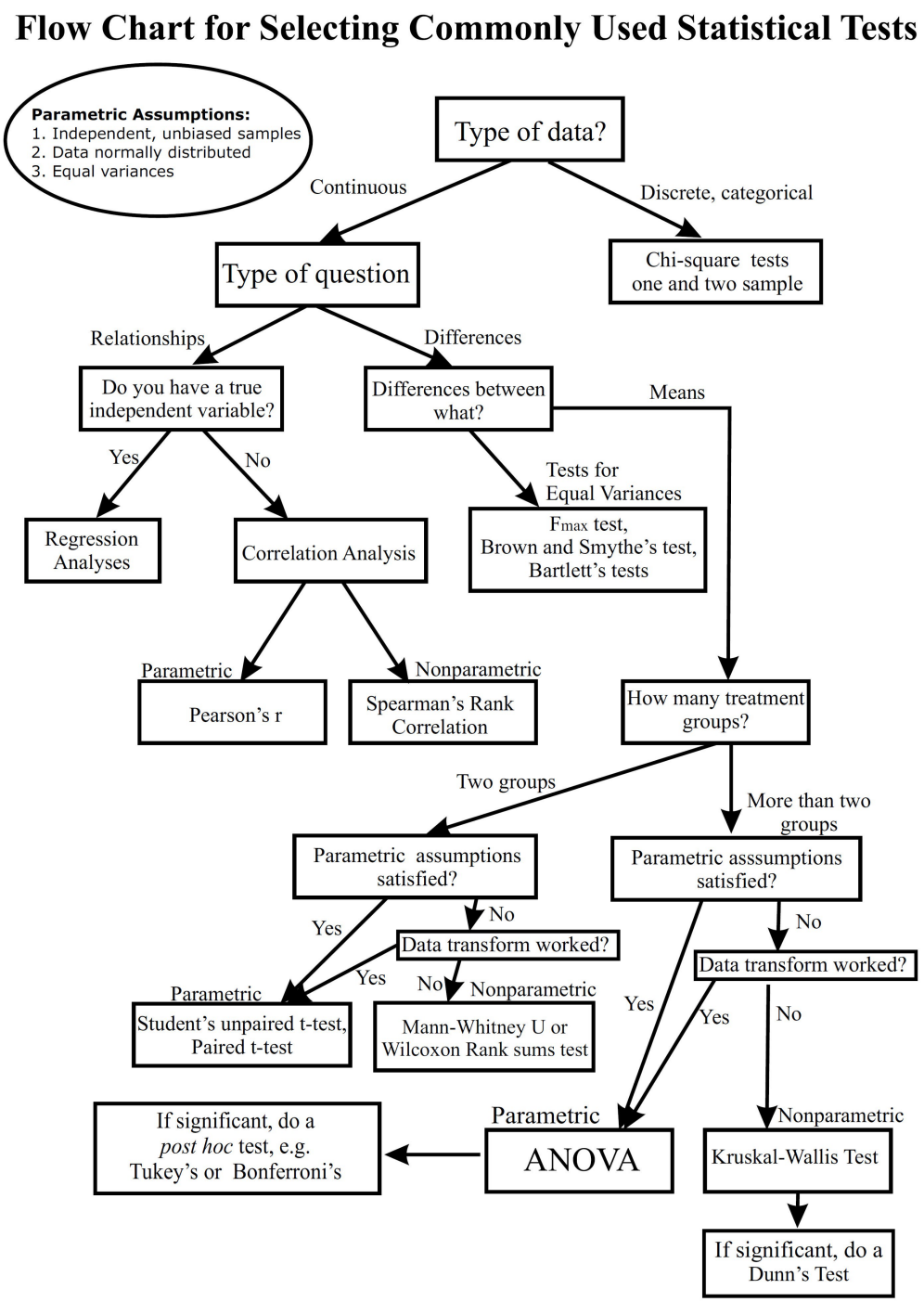
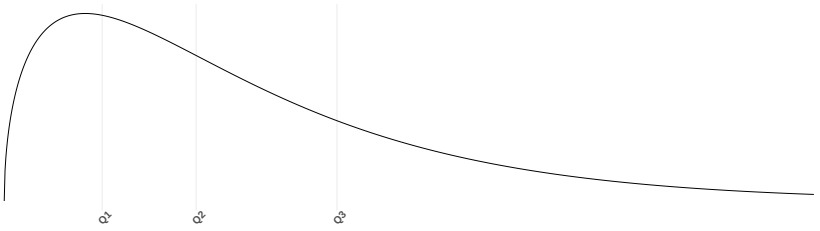


Figure 3.1: fuente: http://abacus.bates.edu/~ganderso/biology/resources/statistics.html

3.1.3.2 Cuantiles

Así como dijimos que la mediana es el valor que deja al 50% de los datos de un lado y al 50% del otro, podemos generalizar este concepto a cualquier X%. Esto son los cuantiles. El cuantil x, es el valor tal que queda un x% de la distribución a izquierda, y 1-x a derecha.

Algunos de los más utilizados son el del 25%, también conocido como Q_1 (el *cuartil* 1), el Q_2 (la mediana) y el Q_3 (el *cuartil* 3), que deja el 75% de los datos a su derecha. Veamos como se ven en la distribución de arriba



3.1.3.3 desvío estándar

- El *desvío estándar* es una medida de dispersión de los datos, que indica cuánto se suelen alejar de la media.

3.1.4 Gráficos estadísticos

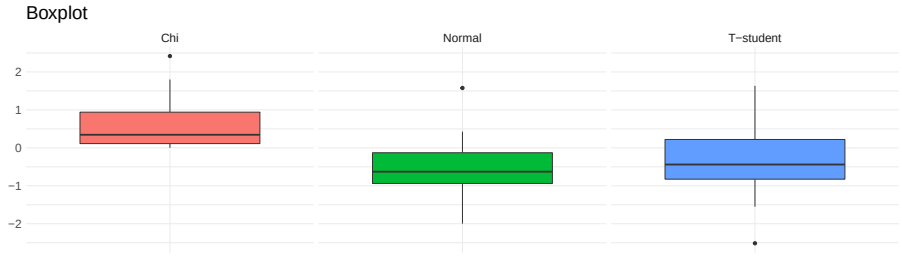
Cerramos la explicación con algunos gráficos que resultan útiles para entender las propiedades estadísticas de los datos.

3.1.4.1 Boxplot

El Boxplot es muy útil para describir una distribución y para detectar outliers. Reúne los principales valores que caracterizan a una distribución:

- Q_1
- Q_2 (la mediana)
- Q_3
- el *rango intercuarítlico* $Q_3 - Q_1$, que define el centro de la distribución
- Outliers, definidos como aquellos puntos que se encuentran a más de 1,5 veces el rango intercuartílico del centro de la distribución.

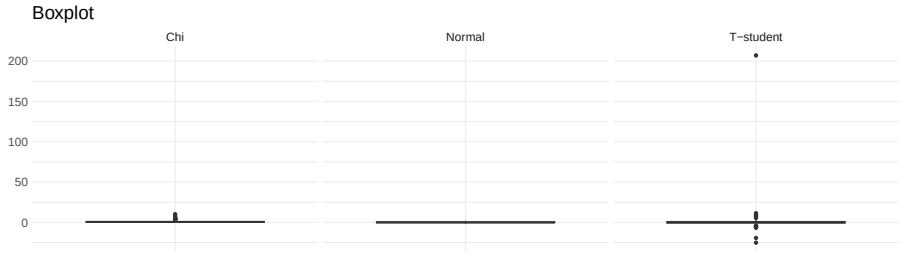
veamos qué pinta tienen los boxplot de números generados aleatoriamente a partir de tres distribuciones que ya vimos. En este caso, sólo tomaremos 15 valores de cada distribución



Algunas cosas que resaltan:

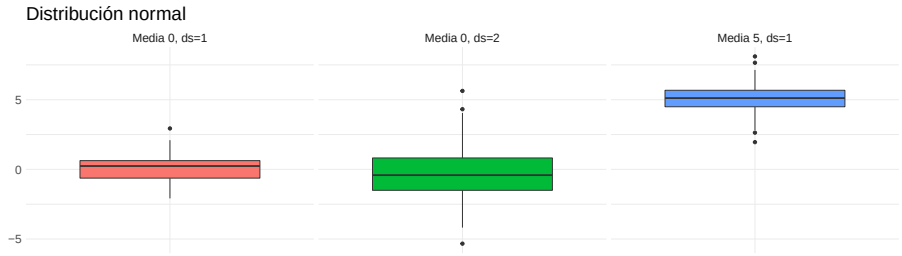
- la distribución χ^2 no toma valores en los negativos.
- La normal esta más concentrada en el centro de la distribución

Podemos generar 100 números aleatorios en lugar de 15:



Cuando generamos 100 valores en lugar de 15, tenemos más chances de agarrar un punto alejado en la distribución. De esta forma podemos apreciar las diferencias entre la distribución normal y la T-student.

También podemos volver a repasar qué efecto generan los distintos parámetros. Por ejemplo



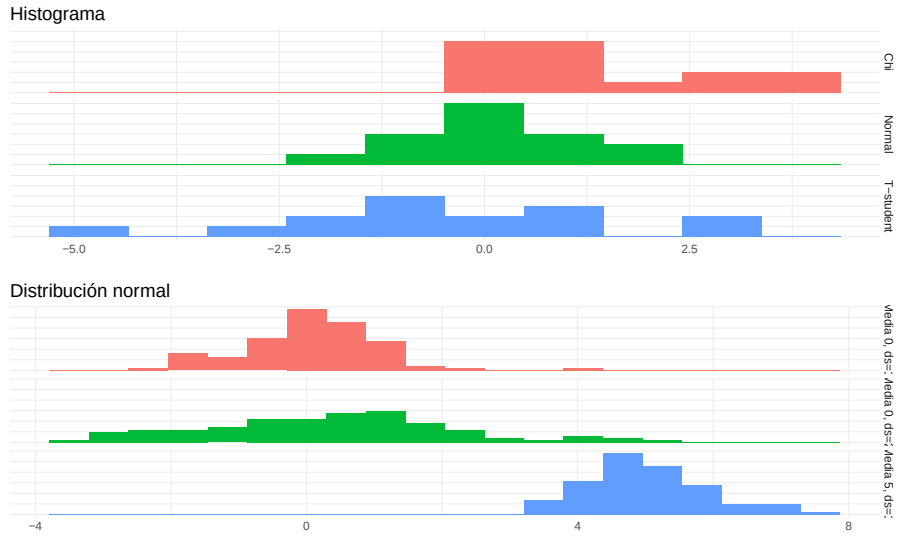
3.1.4.2 Histograma

Otra forma de analizar una distribución es mediante los histogramas:

- En un histograma agrupamos las observaciones en rangos fijos de la variable y contamos la cantidad de ocurrencias.

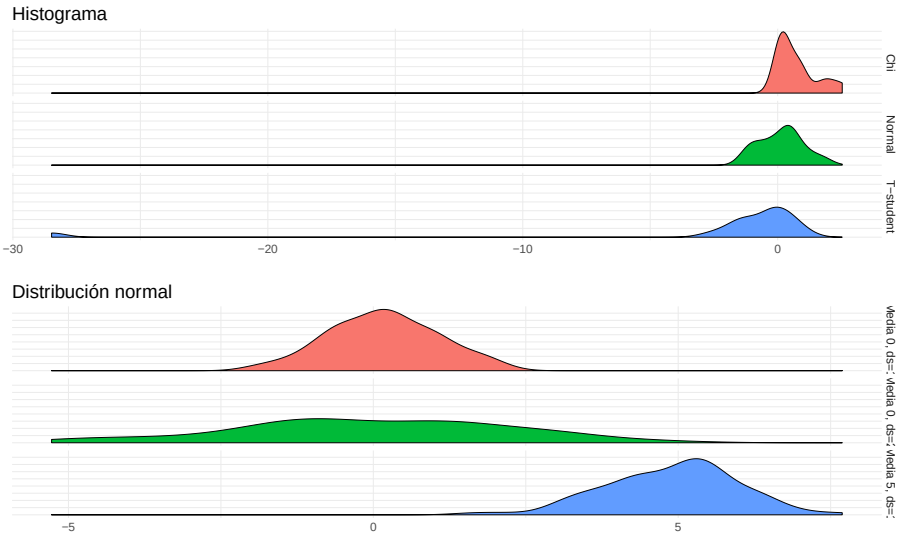
- Cuanto más alta es una barra, es porque más observaciones se encuentran en dicho rango

Veamos el mismo ejemplo que arriba, pero con histogramas



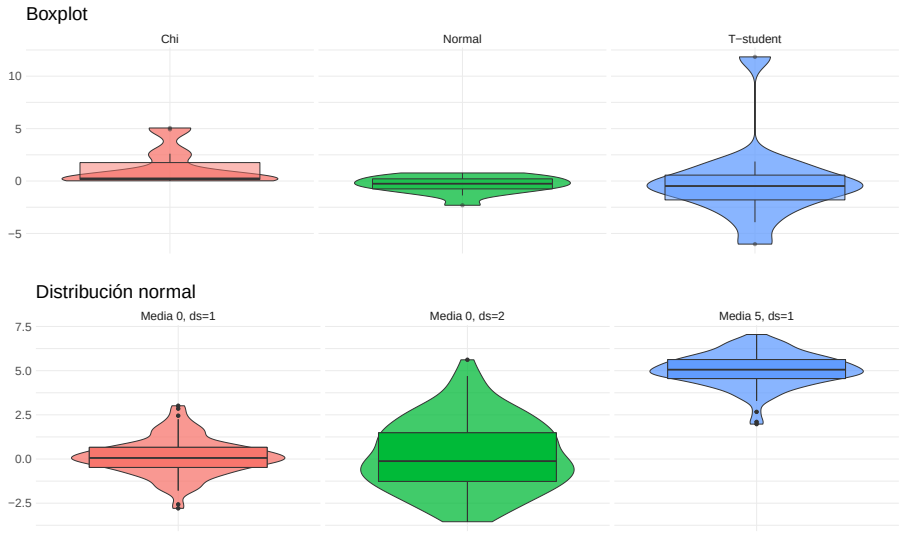
3.1.4.3 Kernel

Los Kernels son simplemente un suavizados sobre los histogramas



3.1.4.4 Violin plots

Combinando la idea de Kernels y boxplots, se crearon los violin plots, que simplemente muestran a los kernels duplicados



3.1.5 Bibliografía de consulta

Quién quiera profundizar en estos temas, puede ver los siguientes materiales:

- https://seeing-theory.brown.edu/
- https://lagunita.stanford.edu/courses/course-v1:OLI+ProbStat+Open_Jan2017/about
- Jay L. Devore, "Probabilidad y Estadística para Ingeniería y Ciencias", International Thomson Editores. https://inferencialitm.files.wordpress.com/2018/04/probabilidad-y-estadistica-para-ingenieria-y-ciencias-devore-7th.pdf

3.2 Práctica Guiada

3.2.1 Generación de datos aleatorios

Para generar datos aleatorios, usamos las funciones

`-rnorm` para generar datos que surgen de una distribución normal `-rt` para generar datos que surgen de una distribución T-student `-rchisq` para generar datos que surgen de una distribución Chi cuadrado

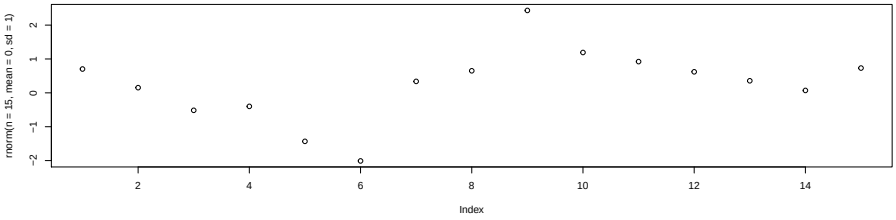
> pero antes, tenemos que fijar la *semilla* para que los datos sean reproducibles

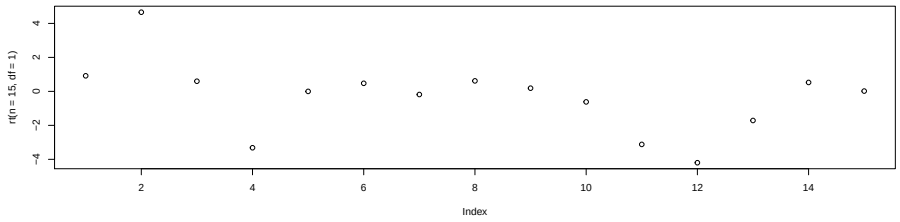
```
##  [1] -1.20706575  0.27742924  1.08444118 -2.34569770  0.42912469
##  [6]  0.50605589 -0.57473996 -0.54663186 -0.56445200 -0.89003783
## [11] -0.47719270 -0.99838644 -0.77625389  0.06445882  0.95949406
```

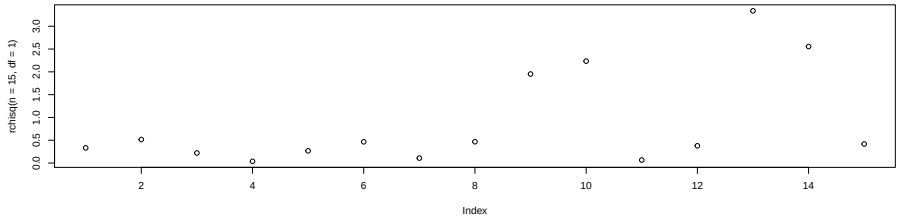
```
##  [1] -0.363717710 -1.603466805 -0.388596796 -0.588007490  0.007839245
##  [6] 14.690527710 -1.863488555  0.022667470 -2.084247299 -0.249237745
## [11] -1.311594174 -3.569055208 -2.490838240 -3.848779244 -4.271087169
```

```
##  [1] 0.5317744 1.4263809 4.2797098 0.2184660 0.6923773 0.0455256 3.1902100
##  [8] 0.2949942 0.5403827 0.1543732 0.8639196 0.1417290 1.1386091 0.2966193
## [15] 0.5110879
```

Para poder ver rápidamente de qué se tratan los valores, podemos usar el comando plot







Noten que el eje X es el índice de los valores, es decir que no agrega información.

3.2.2 Tests

Utilicemos ahora datos reales.

los datos salen de https://data.buenosaires.gob.ar/dataset/femicidios

Vamos a ver ahora las estadisticas de Buenos Aires sobre la cantidad de femicidios por grupo etario. Es interesante preguntarse si hay más femicidios para cierto rango etario.

```
## # A tibble: 19 x 3
##     anio cantidad_femicidios grupo_edad
##    <dbl> <chr>               <chr>
##  1  2015 1                   0 - 15
##  2  2015 2                   16 - 20
##  3  2015 5                   21 - 40
##  4  2015 3                   41 - 60
##  5  2015 -                   61 y más
##  6  2015 1                   Ignorado
##  7  2016 2                   0 - 15
##  8  2016 3                   16 - 20
##  9  2016 4                   21 - 40
## 10  2016 1                   41 - 60
## 11  2016 2                   61 y más
## 12  2016 2                   Ignorado
## 13  2017 …                   0 - 15
## 14  2017 …                   16 - 20
## 15  2017 …                   21 - 40
## 16  2017 …                   41 - 60
## 17  2017 …                   61 y más
## 18  2017 …                   Ignorado
## 19  2017 9                   TOTAL
```

fijense que las estadísitcas no estan desagregadas por rango etario para 2017, que en caso de que haya 0 femicidios pusieron '-' en lugar de 0. Además, como tenemos pocos datos, es mejor hacer un test que compare sólamente dos grupos.

Vamos a reorganizar la información para corregir todas estas cosas

```
## # A tibble: 2 x 2
##   grupo_edad cantidad_femicidios
##   <chr>                   <dbl>
## 1 0-40                       17
## 2 41 y más                    6
```

Con esta tabla de contingencia podemos hacer un test de hipótesis.

¿Cuál usamos? Nos fijamos en el machete, o googleamos, y vemos que como queremos comparar la cantidad de casos por grupos categóricos, tenemos que usar el test Chi.

- H_0 No hay asociación entre las variables
- H_1 Hay asociación entre las variables

La idea es que tenemos dos variables: El rango etario y la cantidad de femicidios

```
##
##  Chi-squared test for given probabilities
##
## data:  femicidios$cantidad_femicidios
## X-squared = 5.2609, df = 1, p-value = 0.02181
```

noten que el resultado lo dan en términos del p-valor. Como el valor es bajo, menor a 0.05, entonces podemos rechazar que no existe relación. O en otros términos, pareciera que la diferencia es significativa estadísticamente.

3.2.3 Descripción estadística de los datos

Volveremos a ver los datos de sueldos de funcionarios

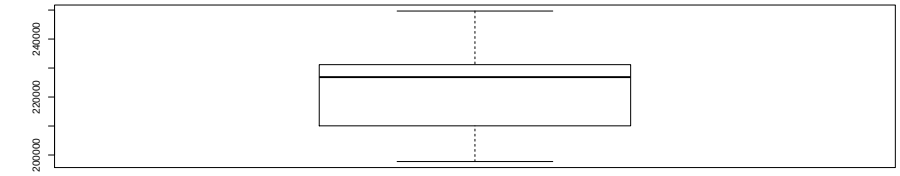
Con el comando `summary` podemos ver algunos de los principales estadísticos de resumen

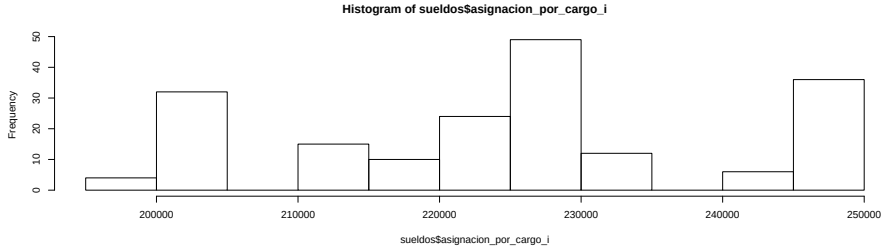
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  197746  210061  226866  225401  231168  249662
```

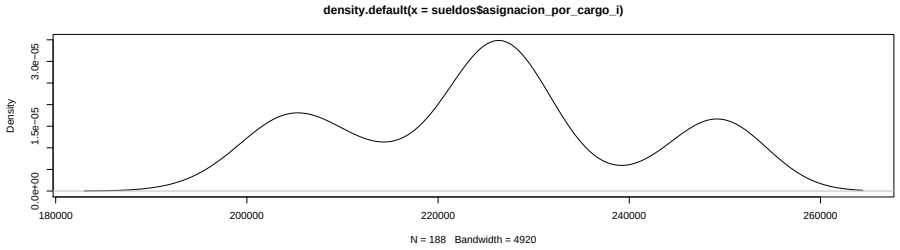
3.2.4 Gráficos estadísticos

No nos vamos a detener demasiado a ver cómo hacer los gráficos de resumen, porque la próxima clase veremos como realizar gráficos de mejor calidad. Como los presentados en las notas de clase

A modo de ejemplo, dejamos los comandos de R base para realizar gráficos






density.default(x = sueldos$asignacion_por_cargo_i)
Density
0.0e+00
1.5e−05
3.0e−05
180000
200000
220000
240000
260000
N = 188 Bandwidth = 4920