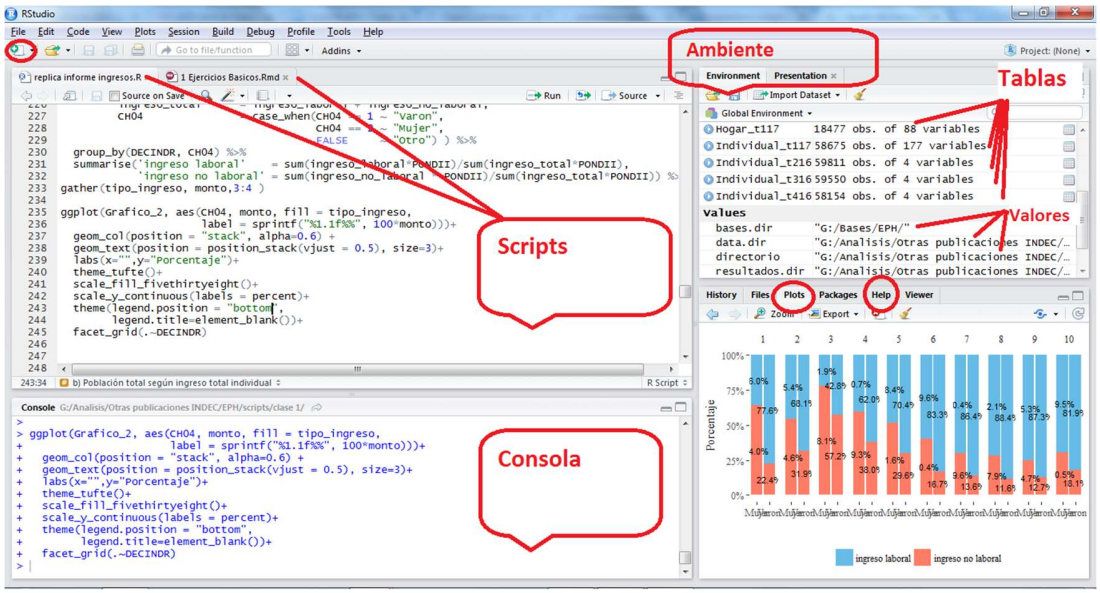
Introducción a R (2018, (Guido Weksler, Diego Kozlowski y Natsumi Shokida)

¿Que es R?

- Lenguaje para el procesamiento y análisis estadístico de datos
- Software Libre
- Sintaxis Básica: R base
- Sintaxis incremental: El lenguaje se va ampliando por aportes de Universidades, investigadores 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**



Pantalla Rstudio

- 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).

Diferencias con STATA y SPSS

- Gratuito
- Únicamente funciona por líneas de código (No hay botones para ejecutar comandos)
- Posibilita trabajar con múltiples bases de microdatos al mismo tiempo, sin mayor dificultad (No requiere abrir cada base, trabajarla por separado y luego cerrarla)
- Más potente
  - Totalmente automatizable

  - Aportes de usuarios
  - Extensible a otros lenguajes y usos (Presentación como esta, diseño de aplicaciones)
- Más veloz

Lógica sintáctica.

Definición de objetos[1]

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 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. De esta manera, si quiere definir al número 1 con el nombre "A", se escribe en el script lo siguiente:

```
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. Para obsevarlo u operar, se escribe el nombre del objeto ("A"). Al correr una linea con el nombre del objeto, la consola del programa muestra su contenido. Entre corchetes observamos el número de orden del elemento en cuestión[2].

```
A

## [1] 1

A+6

## [1] 7
```

---

[1] De ahora en más, se observará un rectangulo más oscuro que el otro que dan cuenta de lo que se escribe en el script y lo que devuelve R, respectivamente.

[2] De momento no se profundizará, pero está claro que el objeto "A" es un valor numérico, un objeto que da cuenta de un número. El hecho de que lo definamos de esta manera le permite a R leerlo de tal manera (podría decirse que le estamos diciendo a R cómo leerlo). Se sugiere que en lo que sigue, el alumno vaya prestando atención a la manera en que "piensa y razona" R, tanto más fácil será codificar cuanto más se entendida esta lógica de razonamiento del programa R.

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
```

R base

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

Existen diferentes tipos de datos básicos:

- numeric: valores numéricos, incluye decimales.
- integer: números enteros, no incluye decimales.
- character valores alfanuméricos, es decir, letras, números y signos mezclados.
- logical: valores lógicos, TRUE o FALSE.

Operadores lógicos:

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

Para dar un ejemplo de su funcionamiento, observe lo que se escribe en el script y la respuesta que da R. Primero se definen los objetos, luego se realizan las comparaciones lógicas, tal como sigue:

```
#Se redefinen los valores A y B
A <-  10
B  <-  20
#Se realizan 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.

Haga una pausa y ejecute la siguiente orden

```
class(A)
```

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

```
class(B)
```

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

```
class(C)
```

```
## [1] "logical"
```

Aquí le estamos preguntando a R (por medio del comando class()) lo siguiente: dado que se han definido los elementos A, B y C, ¿qué tipo de dato básico son?

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

#raiz cuadrada
E <- sqrt(9)
E

## [1] 3

#potencia
F <- 4^2
F

## [1] 16
```

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)`

Agunos ejemplos para comprender su funcionamiento:

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

```
paste("Apruebo", "Econometría", "con", "buena", "nota", sep = ";")

## [1] "Apruebo;Econometría;con;buena;nota"
```

En caso de que deseemos una separación de un espacio, se deberá escribir `sep = " "`, o simplemente omitir el argumento `sep`. El alumno podrá comprobarlo por sí mismo. Nótese que también se podrían definir ciertos objetos y luego realizar la concatenación, tal como sigue:

```r
A <- "Apruebo"
B <- "Econometría"
C <- "con"
D <- "buena"
E <- "nota"

paste(D, E, A, B, C)
```

```
## [1] "buena nota Apruebo Econometría con"
```

El alumno puede intentar ordenarlos tal como se hizo en la primera concatenación.

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

```r
A <- 4
sum(A +3+sqrt(9))
```

```
## [1] 10
```

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

```r
PromAcad <- mean(7, 6, 7, 10, 9, 1) #Nótese que no deben sumarse los elemento
s, dado que constituyen diferentes argumentos dentro de la función
PromAcad
```

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

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. Se pueden aprovechar para emprolijar el código y que la lectura sea más simple.
- 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, permitirá acceder a una determinada columna (o variable) 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.

Objetos:

Existen un gran cantidad de objetos distintos en R:

- Valores
- Vectores
- Data Frames
- Listas

Valores

Los valores y vectores pueden ser, como ya se ha mencionado, a su vez de distintas *clases*:

Numeric

```
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).

Vectores

Para crear un **vector** se utilizará 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+3
D

## [1] 4 7 9

#O también
E <- C + c(1,2,3)
E

## [1] 4 7 9
```

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

A continuación, se crea 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 (como ya se mencionó en el apartado de 'Caracteres especiales') por su número de orden, entre `[ ]`, tal como se observa a continuación:

```
E[2]

## [1] "Federico"
```

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

```
elemento2 <-  E[2]
elemento2

## [1] "Federico"
```

para **borrar** un objeto del ambiente de trabajo, se utilizará el comando *`rm()`*

```
rm(elemento2)
elemento2

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

Este comando es muy importante dado que permite mantener ordenado el ambiente, atendiendo a los objetos que se están utilizando o que se quieran utilizar en el momento más próximo.

También se puede cambiar los argumentos de un vector. Por ejemplo, si se quisiera cambiar el argumento 2 del vector `E` por “Pablo”, se podría hacer de la siguiente manera

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

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

O también redefinir el vector E, ahora atendienod a que se trata de “Pablo” y no de “Federico”. Como se dijo anteriormente, el mismo resultado a través de distintas vías.