R desde Cero

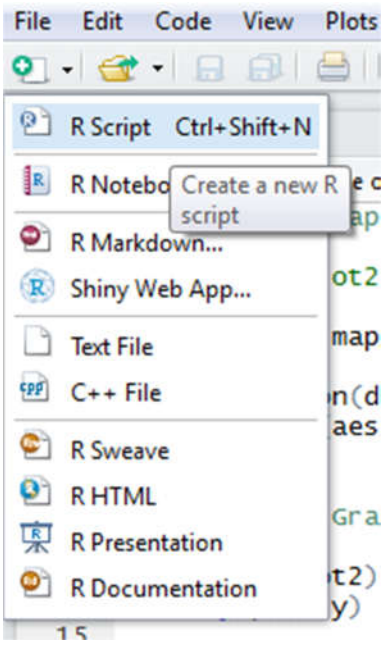
Practica Nº 1.

RStudio

Buscar el ícono de RStudio 

e ingresar en el programa. Buscar el panel de la Consola y ubicar el prompt (>).

Hacer click sobre el ícono  y seleccionar la opción: R Script:



Usando la consola

Escribir en el prompt (>):

```
> 1 + 100                    y presionar enter
```

¿Qué resultado mostró?

_____

R te mostrará la respuesta, precedido de un "[1]". No te preocupes por esto por ahora, lo explicaremos más adelante. Por ahora piensa en eso como parte de la salida.

Ahora escribe:

```
> 1 +                          y presiona enter
```

¿Qué pasa en la consola?

_____

Si escribes un comando incompleto R esperará a que lo completes y por eso muestra el signo + y no el prompt (>). Cada vez que presionas Enter y R te muestra un "+" en lugar de ">", significa que está esperando que completes el comando. Si deseas cancelar un comando, simplemente presiona "Esc" y RStudio te devolverá el ">" prompt.

Ahora escribe:

```
> 1 +  "A"                        y presiona Enter
```

¿Qué pasa en la consola?

_____

Una parte un poco confusa de R es cómo informa errores, advertencias y mensajes. El tema predeterminado en RStudio colorea errores, advertencias y mensajes en rojo, lo que hace que parezca que hiciste algo mal. Sin embargo, **ver texto rojo en la consola no siempre es malo.**

R desde Cero

Sigamos haciendo cuentas

Cuando usas R como calculadora, el orden de las operaciones es el mismo que has aprendido en la escuela.

De mayor a menor precedencia:

Paréntesis: (,)

Exponente: ^ o **

División: /

Multiplicación: *

Suma: +

Resta: -

Ahora escribe:

```
> 3 + 5 * 2                          y presiona enter
```

Resultado: _____

Usa paréntesis para agrupar las operaciones a fin de forzar el orden de la evaluación o para aclarar lo que deseas hacer.

Ahora escribe:

```
> (3 + 5) * 2                          y presiona enter
```

Resultado: _____

Escribe las siguientes líneas de código:

```
> (3 + (5 * (2 ^ 2)))
```

Resultado: _____

```
> (3 + (5 * (2 ^ 2))) # Esta ecuación corresponde al libro 3.
```

Resultado: _____

```
> # Esta ecuación corresponde al libro 3.
```

Resultado: _____

El texto de la segunda línea de código se llama "comentario". Todo lo que sigue después del símbolo hash (o numeral) # es ignorado por R cuando se ejecuta el código. Sirve para documentar que estamos haciendo, muy útil para que otros usen nuestro código….o para nuestros futuros yo….por ejemplo dentro de 6 meses

Funciones matemáticas

R tiene muchas **funciones** matemáticas integradas. Para llamar a una función, **simplemente escribimos su nombre seguido de paréntesis ( ).** Todo lo que escribas dentro de los paréntesis se llaman **argumentos** de la función, probemos:

```
> sin(1)  # función trigonométrica
```

Resultado: _____

```
> log(1)  # logaritmo natural
```

Resultado: _____

```
> log10(10) # logaritmo en base-10
```

Resultado: _____

```
> exp(0.5) # e^(1/2)
```

Resultado: _____

R desde Cero

Comparando

Podemos realizar comparaciones en R, probemos estas líneas de código en la consola:

```
> 1 == 1  # igualdad (observa dos signos iguales, se lee como
"es igual a")
```

Resultado: _____

```
> 1 != 2  # desigualdad (leída como "no es igual a")
```

Resultado: _____

```
> 1 < 2  # menor que
```

Resultado: _____

```
> 1 <= 1  # menor o igual que
```

Resultado: _____

```
> 1 > 0  # mayor que
```

Resultado: _____

```
> 1 >= -9 # mayor o igual que
```

Resultado: _____

R desde Cero

Paquetes

Por código

Es posible agregar funciones a R escribiendo un paquete u obteniendo un paquete escrito por otra persona. Hay más de 10,000 paquetes disponibles en CRAN (la red completa de archivos R). R y RStudio tienen funcionalidad para administrar paquetes:

- Puedes ver qué paquetes están instalados escribiendo `installed.packages()`
- Puedes instalar paquetes escribiendo `install.packages("nombre_de_paquete")`
- Puedes actualizar los paquetes instalados escribiendo `update.packages()`
- Puedes eliminar un paquete con `remove.packages("nombre_de_paquete")`
- Puedes hacer que un paquete esté disponible para su uso con `library(nombre_de_paquete)`

Vamos a cargar para su uso el paquete **tydiverse:**

```
> library(tydiverse)
```

Instala el siguiente paquete: **gapminder**

_____-

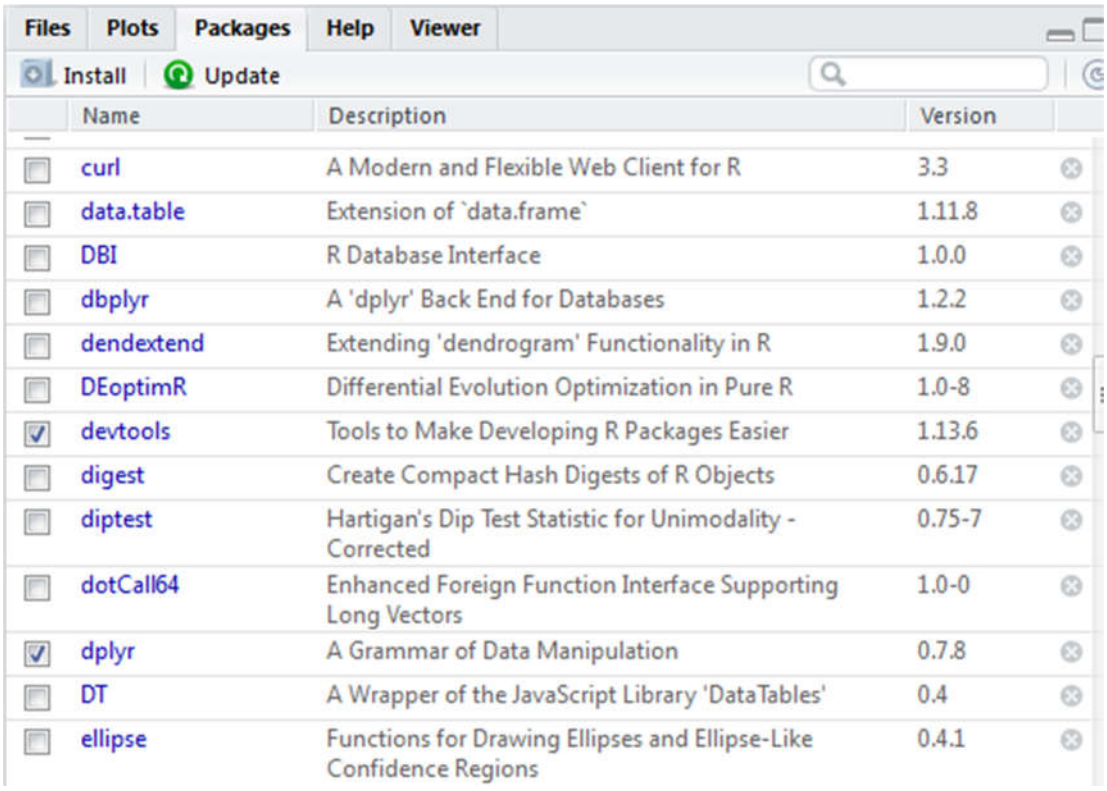
Ahora cárgalo para que esté disponible para su uso:

_____-

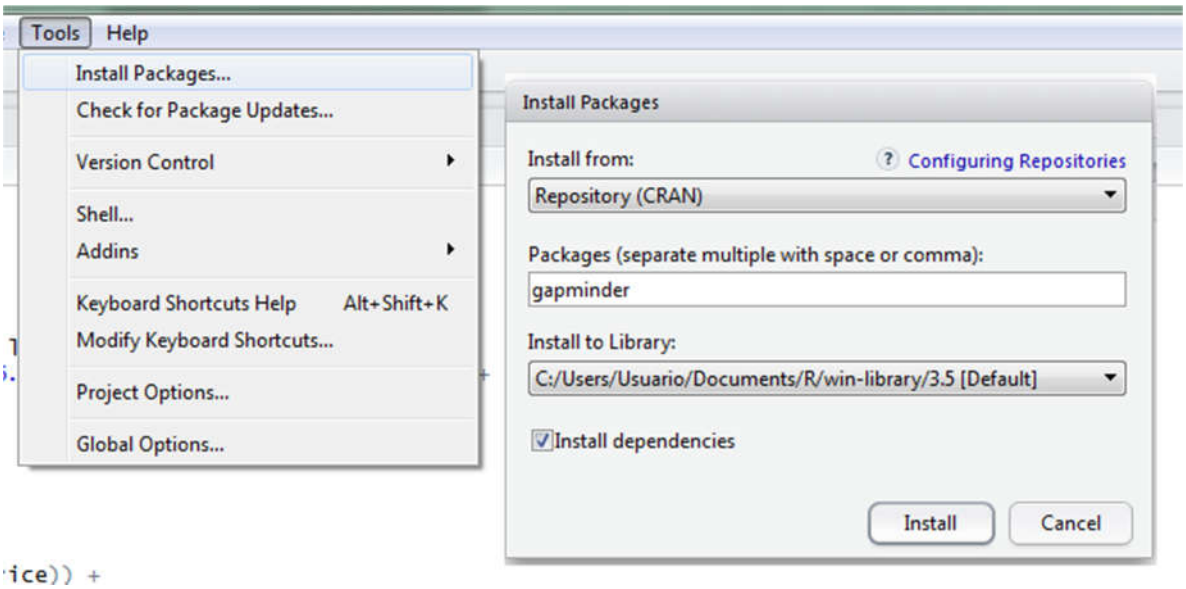
Haciendo lo mismo con los menú de RStudio:

Cargando un paquete: en el panel Paquetes, buscar el paquete a cargar y hacer click en la casilla correspondiente, por ejemplo: **devtools** y **dplyr**

R desde Cero



Instalando un paquete:



R desde Cero

Variables y asignaciones

Podemos almacenar valores en variables usando el operador de asignación **<-**, veamos un ejemplo, escribe:

```
> x <- 1/40
```
Resultado: _____

Observa que la asignación no muestra el valor. En cambio, lo almacena para más adelante en algo llamado **variable**. `x` ahora contiene el **valor** `0.025`. Escribe:

```
> x
```

Resultado: _____

Más precisamente, el valor almacenado es una *aproximación decimal* de esta fracción, llamado número de coma flotante o **floating point**.

Busca la pestaña `Environment` en uno de los paneles de RStudio, y verás que `x` y su valor han aparecido. Nuestra variable `x` se puede usar en lugar de un número en cualquier cálculo que espere un número, escribe:

```
> log(x)
```

Resultado: _____

Tener en cuenta que las variables pueden reasignarse, es decir, puedes cambiar el valor almacenado en la variable, escribe:

```
> x <- 100
> x
```

Resultado: _____

`x` tenía el valor `0.025` y ahora tiene el valor `100`.

También, los valores de asignación pueden contener la variable asignada, ejecuta el siguiente código:

```
> x <- x + 1 # observa cómo RStudio actualiza la descripción
de x en la pestaña superior derecha
> y <- x * 2
```

El lado derecho de la asignación puede ser cualquier expresión de R válida. La expresión del lado derecho *se evalúa por completo* antes de que se realice la asignación.

También es posible utilizar el operador = para la asignación (es la práctia común en la mayoría de los lenguajes de programación):

```
> x = 1/40
```

Esta forma es menos común entre los usuarios R. Lo más importante es **ser consistente** con el operador que usas. Ocasionalmente hay lugares donde es menos confuso usar <- que =, y es el símbolo más común usado en la comunidad. Entonces la recomendación es usar <-.

Nombrando variables

Los nombres de las variables pueden contener letras, números, guiones bajos y puntos. **No pueden comenzar con un número ni contener espacios en absoluto.** Existen diferentes convenciones para nombres largos de variables, estos incluyen

- puntos.entre.palabras
- guiones_bajos_entre_palabras
- MayúsculasMinúsculasParaSepararPalabras

Lo que uses depende de ti, pero **sé consistente**.

De los siguientes ejemplos, ¿Cuáles son nombres de variables válidas en R?

```
min_height
max.height
_age
.mass
MaxLength
min-length
2widths
celsius2kelvin
```

Ayuda

Dentro de RStudio

Escribir en la consola:

```
> ?help # observa cómo RStudio actualiza la pantalla del panel Ayuda
```

Resultado: _____

Para buscar ayuda en operadores especiales, usa comillas, por ejemplo, escribir en la consola:

```
>?"<-"
```

Muchos paquetes vienen con "viñetas": tutoriales y documentación de ejemplo extendida. Sin ningún argumento, `vignette()` listará todas las viñetas disponibles para todos los paquetes instalados; `vignette(package="package-name")` listará todas las viñetas disponibles para `package-name`, y `vignette("vignette-name")` abrirá la viñeta especificada.

Si un paquete no tiene viñetas, generalmente puedes encontrar ayuda escribiendo `help("package-name")`

Pide ayuda sobre el paquete ggplot2

Código: _____

¿dónde se mostró la ayuda?

_____

_____

Consulta que viñetas hay disponibles para el paquete ggplot2

Código: _____

¿dónde se mostraron los resultados?

_____

_____

Consulta la primera viñeta que aparece como resultado:

Código: _____

¿Sobre que es esa viñeta?

_____

_____

Si no estás seguro de en qué paquete está una función, o cómo se escribe específicamente, puedes hacer una búsqueda difusa: `??function_name, escribe en la consola:`

```
> ??ggplot
```

¿qué aparece en la ventana de ayuda?

_____

_____

Usemos la función > sessionInfo() para ver los detalles de nuestra sesión de R Actual:

```
> sessionInfo()
```

Buscar la ayuda para la función `c`. ¿Qué tipo de vector crees que crearás si evalúas lo siguiente?:

```
c(1, 2, 3)
c('d', 'e', 'f')
c(1, 2, 'f')
```

Resultado:

_____

_____

Buscar la ayuda para la función **paste**. ¿Para que sirve el argumento **sep**? ¿Existe un argumento **collapse**?

_____

_____

_

Usa la ayuda para encontrar una función (y sus parámetros asociados) que puedas usar para cargar datos de un archivo csv en los cuales las columnas están delimitadas con "\ t" (tab) y el punto decimal es un "." (punto).

Esta comprobación para el separador decimal es importante, especialmente si estás trabajando con colegas internacionales ya que diferentes países tienen diferentes convenciones para el punto decimal (i.e. coma vs. punto). sugerencia: usa **??csv** para buscar funciones relacionadas con csv.

Fuera de R Studio

En google realiza esta búsqueda:

hacer un gráfico de lineas en R

¿Qué resultados arroja?, ¿Puedes ver el nombre de alguna función o paquete en los títulos de los resultados?

Ahora realiza la siguiente búsqueda en Google:

hacer un gráfico de lineas en R ggplot2

¿Qué resultados arroja?, ¡Los resultados cambiaron?, ¿Puedes ver el nombre de alguna función o paquete en los títulos de los resultados?

Visita las siguientes páginas:

https://www.rstudio.com/resources/cheatsheets/

https://community.rstudio.com/

https://stackoverflow.com/

R desde Cero

Comunidades - Opcional

1. Ingresa en https://www.meetup.com/es-ES/rladies-santa-rosa/
2. Registrate como usuario de meetup
3. Registrate como miembro del grupo de R-Ladies Santa Rosa.

Ejercicios adicionales

¿Cuál será el valor de cada variable después de cada comando en el siguiente programa?

```
mass <- 47.5
```
Resultado: _____

```
age <- 122
```
Resultado: _____

```
mass <- mass * 2.3
```
Resultado: _____

```
age <- age - 20
```
Resultado: _____

Escribe un comando para **comparar la variable mass con age**. ¿Es la variable **mass** más grande que **age**?

Resultado: _____

Referencias

1. https://swcarpentry.github.io/r-novice-gapminder-es/
2. https://moderndive.com/index.html
3. https://flor14.github.io/Fundamentos_de_R/