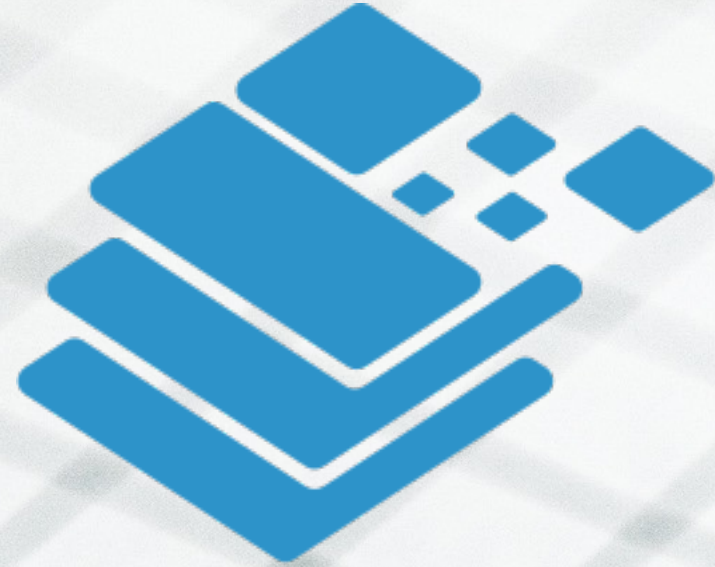




**Spatial
Lab
Analytics**

partimos pronto



**Spatial
Lab
Analytics**

ANÁLISIS DE DATOS CON R

Introducción al lenguaje R

Bastían Olea Herrera - baolea@uc.cl

Primera aproximación al lenguaje de programación R orientado al análisis de datos. Conceptos principales, elementos fundamentales del lenguaje, flujo de trabajo, y buenas prácticas.

01 **RStudio**

Aprendiendo a utilizar el entorno de desarrollo RStudio para trabajar con el lenguaje de programación R

02 **Objetos**

Creación y asignación de elementos persistentes en nuestro entorno de R

03 **Vectores**

Unidad básica del registro y procesamiento de datos en R

04 **Funciones**

Uso y creación de herramientas programáticas para trabajar con datos y acelerar tu flujo de trabajo



Studio[®]

01

Aprendiendo a utilizar el
entorno de desarrollo RStudio
para trabajar con el lenguaje de
programación R



R

- Lenguaje de programación
- Lanzado en 1993
- Usado por línea de comandos
- Desarrollado por R Core Team
- Basado en el lenguaje S (1976)
- Software libre



RStudio

- Entorno de desarrollo integrado (IDE) enfocado en R
- Lanzado en 2011
- Interfaz gráfica (ventanas y botones)
- Desarrollado por Posit (ex RStudio)
- Software libre





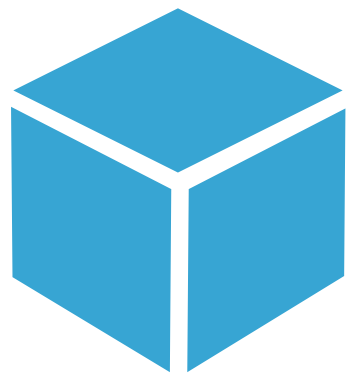
Script

Archivo de texto .R en el que escribimos nuestro código, en pasos, y siguiendo un orden lógico.



Consola

Es la forma directa de interactuar con R, un comando a la vez, con resultados efímeros.



Proyecto

Archivo .Rproj que marca nuestro espacio de trabajo: una carpeta específica reúne todas las piezas de nuestro análisis.



Script

1

Consola

2

Entorno

3

Archivos

4

RStudio

Go to file/function

Project: (None)

Environment History Connections Tutorial

262 MiB

Grid

R Global Environment

Name	Type	Le...	Size	Value
iris	data...	5	7.1...	150 obs. of...
penguins	tbl_df	8	16...	344 obs. of...

Files Plots Packages Help Viewer Pre

Home > R

Name	Size
cep-dashboard	
cep-procesamiento	
comisarias_chile	
comparador_mapas_chile	
corrupcion_chile	
datos_comunales_chile	
delincuencia_chile	
economia_chile	

1:1 (Top Level) R Script

Console Terminal Background Jobs

R 4.4.1 ~/

1	Adelie	Torgersen	39.1	18.7
2	Adelie	Torgersen	39.5	17.4
3	Adelie	Torgersen	40.3	18
4	Adelie	Torgersen	NA	NA
5	Adelie	Torgersen	36.7	19.3
6	Adelie	Torgersen	39.3	20.6
7	Adelie	Torgersen	38.9	17.8
8	Adelie	Torgersen	39.2	19.6
9	Adelie	Torgersen	34.1	18.1
10	Adelie	Torgersen	42	20.2

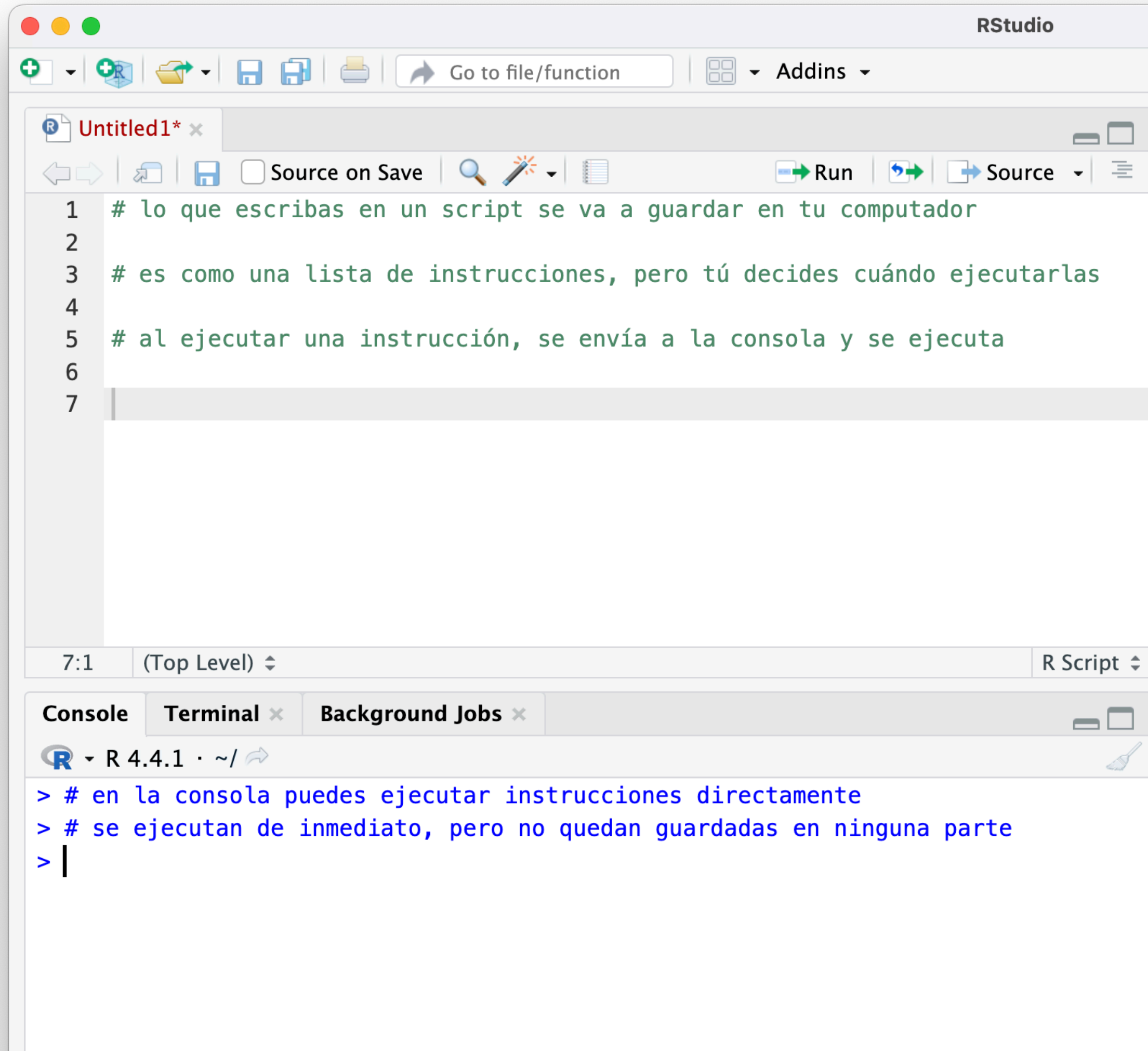
i 334 more rows

- 1 Panel de scripts:** aquí tenemos nuestros archivos de texto con nuestro código. Podemos tener varias pestañas de distintos archivos de texto. Ejecutamos el código poniendo el cursor en la línea que deseemos y presionando *comando + enter*, o el botón *Run*.
- 2 Panel de consola:** en la consola se imprimen los resultados que arroja R a partir del código que ejecutamos en los scripts. También podemos ejecutar código directamente en la consola.
- 3 Panel de entorno:** acá veremos los objetos que vayamos creando, que pueden ser números, texto, tablas de datos, funciones, gráficos y otros.
- 4 Panel de archivos:** en este panel podemos navegar los archivos y carpetas de nuestro proyecto y/o computador.



Consola y Scripts

- Todos los **comandos** se ejecutan por medio de la **consola**
- Pero al ejecutar código en la consola, el código no se guarda!
- Para guardar el código, escribimos en **scripts**
- Escribimos las instrucciones en un script, y el script se encarga de pasarlos a la consola y ejecutarlos
- El panel de scripts tiene botones que nos ayudan a ejecutar código



Operaciones básicas

- Podemos realizar cualquier operación matemática en la **consola** de RStudio o en un script.
- Para **ejecutar** un comando, pon el cursor de texto en la línea o expresión que desees ejecutar, y presiona el botón *Run*, o las teclas *control + enter*.
- El resultado de todas las operaciones aparece en la **consola**.

```
2 + 2 #suma  
> [1] 4
```

```
50 * 100 #multiplicación  
> [1] 5000
```

```
4556 - 1000 #resta  
> [1] 3556
```

```
6565 / 89 #división  
> [1] 73.76404
```

```
10^4 #potencias  
> [1] 10000
```


Operaciones básicas

- Los **comentarios** nos permiten poner texto en cualquier parte del script sin que afecte los cálculos.
- También podemos poner un comentario al final de una línea sin que afecte el código

```
1 + 1 + 1 + 1
```

```
# comentario: quizás esto debería  
# ser de otra forma, porque  
# la verdad quedó bien mal...
```

```
1 * 4 # así queda mucho mejor
```

Tipos de datos

- Las cosas que podemos hacer con los objetos de nuestro entorno depende del tipo que sean. En R existen varios tipos:

- **Numéricos**

1 2 3 4 5.1 5.2 5.333

- **Caracter** (texto)

"ésta es una cadena de texto"

- **Lógicos** (verdadero o falso)

TRUE FALSE TRUE

y muchos más!

`objetos <- 4`

02

Creación y asignación de
elementos persistentes en
nuestro entorno de R

Objetos

Podemos guardar cualquier expresión de R como un objeto. A esto le llamamos **asignación**.

Para asignar un objeto, usamos el operador de asignación: `<-`

```
cifra <- 4
```

En este ejemplo, el objeto **cifra** es creado a partir del dato **4**

Objetos

- Al asignar algo, creamos o modificamos un objeto con el valor que le estamos asignando.
- Al ejecutar un objeto, obtenemos su valor en la consola.

```
edad <- 31
```

```
edad
```

```
> [1] 31
```

```
año <- 2024
```

```
año
```

```
> [1] 2025
```

```
animal = "gato"
```

```
animal
```

```
> [1] "gato"
```

Entorno

- Al crear un objeto, éste aparece en nuestro entorno de R.
- El entorno de R son todos los objetos que vayamos creando y que R va manteniendo en su memoria.
- Para poder usar algo, tiene que estar en nuestro entorno!



Operaciones con objetos

- Podemos usar el objeto creado para lo que queramos.
- Podemos pensar en crear objetos como asignar variables, y usar estas variables para llevar a cabo operaciones.

```
edad <- 31
edad
> [1] 31
```

```
año <- 2024
año - edad
> [1] 1993
```

```
perros = 2
gatos = 3
animales <- perros + gatos
animales
> [1] 5
```

```
presupuesto = 100000
pizza = 15000
presupuesto - pizza * 10
> [1] -50000
```

Comparaciones

- Podemos realizar **comparaciones** entre distintos datos, y recibiremos una respuesta de verdadero/falso.

```
1 == 1  
> [1] TRUE
```

```
2 == 6  
> [1] FALSE
```

```
2 + 2 == 4  
> [1] TRUE
```

```
2 + 2 != 3  
> [1] TRUE
```

```
100 > 4  
> [1] TRUE
```


Comparaciones con objetos

- Como los objetos pueden contener datos, podemos usar esos datos para compararlos con otros datos, u otros objetos.
- Las comparaciones son el principio que luego nos permitirá filtrar datos, crear variables, y más.

```
edad >= 18  
> [1] TRUE
```

```
mínimo <- 35
```

```
edad > mínimo  
> [1] FALSE
```

```
año == 2024  
> [1] FALSE
```

```
año != 2024  
> [1] TRUE
```

```
fecha2 <- 1980
```

```
fecha2 > año  
> [1] TRUE
```

```
vectores <- c(1, 2, 3)
```

03

Unidad básica del registro y
procesamiento de datos en R

Vectores

- Los vectores corresponden a secuencias de elementos, como una cadena de valores.
- Estos elementos son de un mismo tipo (numérico, carácter, lógico)
- Son la forma más básica de registrar e interactuar con observaciones o casos

```
vectores <- c(1, 2, 3)
```

```
c(1, 2, 3, 4, 5, 6)
```

```
c("a", "b", "c", "d")
```

```
c(TRUE, FALSE, FALSE)
```

```
c(0.2, 0.1, -0.0, -0.1)
```


Vectores

operaciones

- Podemos realizar operaciones matemáticas sobre los vectores
- La operación se aplicará a cada elemento del vector

```
numeros <- c(11, 12, 13, 14)
```

```
numeros + 100  
> [1] 111 112 113 114
```

```
numeros2 <- numeros + 800  
numeros3 <- numeros2 * 56  
numeros3  
  
> [1] 45416 45472 45528 45584
```

Vectores

comparaciones

- También podemos realizar comparaciones sobre los valores de un vector

```
edades <- c(54, 34, 65, 21, 32)
```

```
edades > 40
```

```
> [1] TRUE FALSE TRUE FALSE FALSE
```

```
valores <- c(1.01, 1.13, 1.02, 1.20, 1.11)
```

```
valores > 1.05
```

```
> [1] FALSE TRUE FALSE TRUE TRUE
```

```
valores == 1.02
```

```
> [1] FALSE FALSE TRUE FALSE FALSE
```

Operadores

- Los operadores son símbolos que nos permiten realizar operaciones específicas:

& especifica que las dos condiciones deben cumplirse

| indica que cualquiera de las condiciones debe cumplirse

%in% significa que los valores deben estar dentro del conjunto

```
valores <- c(1.01, 1.13, 1.02, 1.20, 1.11)
```

```
valores > 1.05 & valores < 1.15  
[1] FALSE TRUE FALSE FALSE TRUE
```

```
valores > 1.12 | valores < 1.05  
> [1] TRUE TRUE TRUE TRUE FALSE
```

```
animales <- c("serpiente", "perro",  
"gato", "rata", "gallina", "pez")
```

```
mamíferos <- c("vaca", "perro", "caballo",  
"gato", "humano", "rata")
```

```
animales %in% mamíferos  
> [1] FALSE TRUE TRUE TRUE FALSE FALSE
```


Subconjuntos

- Los corchetes nos permiten filtrar un vector usando su posición.
- Los resultados de las comparaciones son vectores de tipo lógico. Podemos usar estos vectores lógicos para *filtrar* vectores, dejando sólo los que cumplen las condiciones.

```
animales[5]  
> [1] "gallina"
```

```
animales[3:5]  
> [1] "gato"      "rata"      "gallina"
```

```
animales[animales %in% mamíferos]  
> [1] "perro" "gato"  "rata"
```

```
valores[valores > 1.1]  
> [1] 1.13 1.20 1.11
```

```
valores[valores > 1.05 & valores < 1.15]  
> [1] 1.13 1.11
```

funciones() { ... }

04

Uso y creación de herramientas
programáticas para trabajar con
datos y acelerar tu flujo de
trabajo

Funciones

- Las funciones son pequeños programas que nos permiten realizar distintas operaciones.
- Las funciones tienen **argumentos**, que son los inputs que recibe la función
- Si no sabemos cómo usar una función, podemos buscar su nombre en el panel de Ayuda de RStudio, o escribir el nombre de la función con un signo de interrogación antes: `?funcion`

`mean()` # calcular promedio

`median()` # calcular mediana

`sum()` # sumar elementos

`min()` # valor mínimo

`max()` # valor máximo

`typeof()` # tipo del objeto

`is.na()` # evaluar si es missing

`as.numeric()` # convertir a numérico

Funciones

- Para crear una función, usamos la función `function()`
- Dentro de esta función se especifican los inputs, y luego en el cuerpo se define lo que hará la función.
- Dentro del **cuerpo** de la función podremos usar los inputs para realizar lo que necesitemos.
- Lo último que hagamos en la función será lo que la función **retorna**; es decir, su output

```
calculo1 <- function(input1) {  
  # cuerpo  
  input1 * 1000  
}
```




**Spatial
Lab
Analytics**

Soluciones en análisis de datos

www.spatiallab.cl