

# Practice IV. Work in groups. Functions to compute sectorial indices

---

DRA. ANNA BOQUÉ CIURANA  
DR. JON XAVIER OLANO POZO

# Objectives of Practice IV



Apply R programming to compute sectorial indices.



Work collaboratively in small groups.



Learn how to use both package-based and custom R functions.

# Why Use R for This Analysis?

---



R IS OPEN-SOURCE,  
REPRODUCIBLE, AND  
COMMUNITY-SUPPORTED.



IDEAL FOR WORKING WITH TIME  
SERIES, SPATIAL DATA, AND  
CLIMATE INDICATORS.



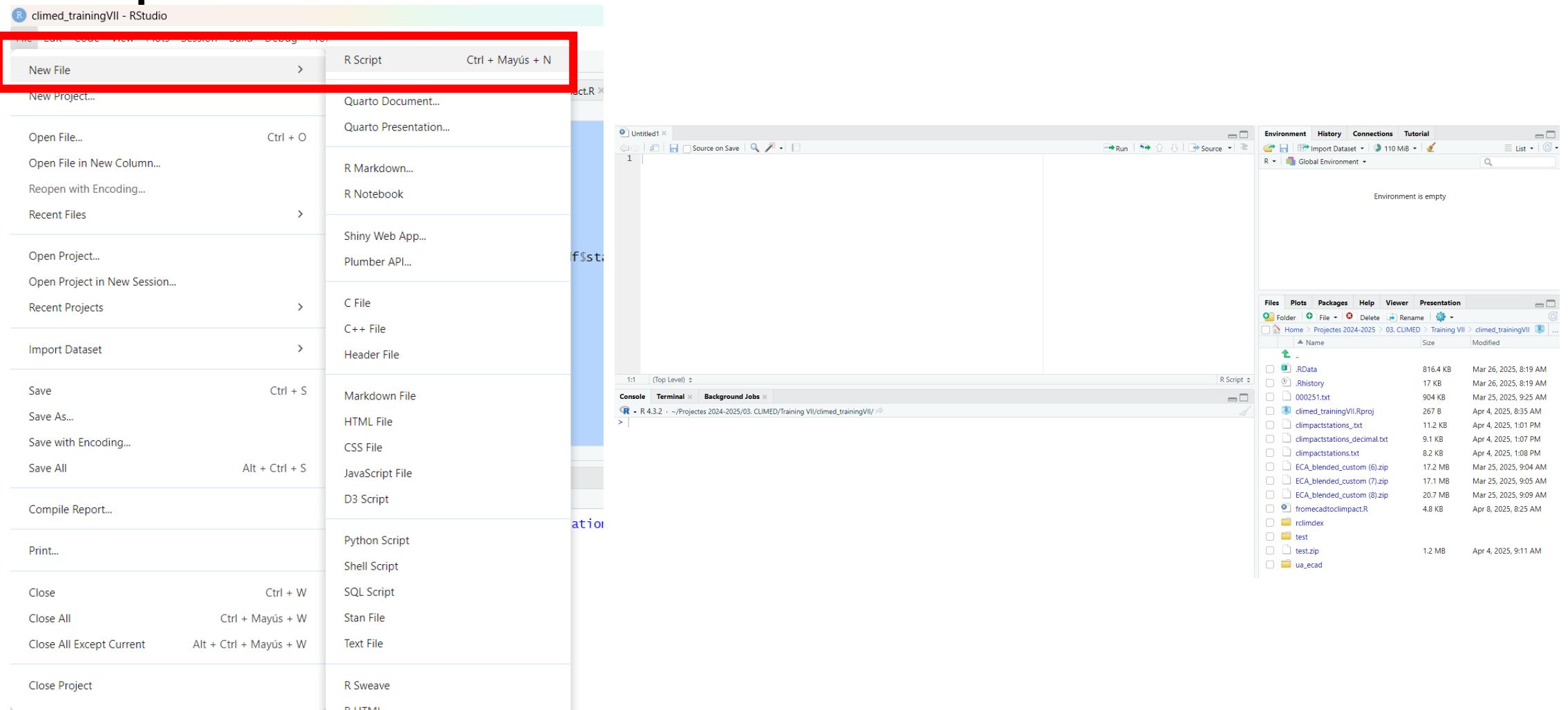
COMPATIBLE WITH CLIMATE-  
SPECIFIC PACKAGES AND  
GENERAL DATA ANALYSIS TOOLS.

---

## Hands-on practice

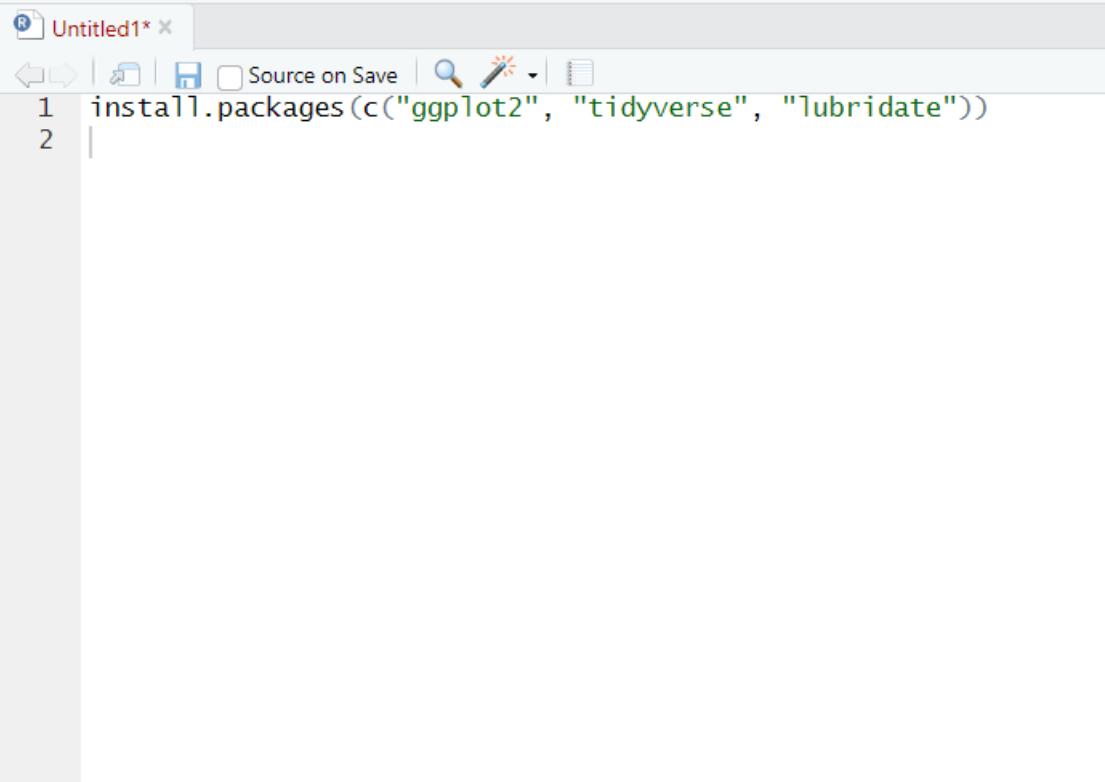
1. Install and load packages
2. Load and prepare data
3. Apply a predefined function
4. Create a custom function
5. Visualization of the results

# 0. Prepare R-Studio



# 1. Install and load packages

- # Install packages
- install.packages(c("tidyverse", "lubridate"))
- # Load librarys
- library(tidyverse)
- library(lubridate)



```
R Untitled1* 
install.packages(c("ggplot2", "tidyverse", "lubridate"))
|
```

## 2. Load Data

```
library(ggplot2)
library(tidyverse)
library(lubridate)
library(readr)
climate_data<- read_table("test/000251.txt")
View(climate_data)
```

The screenshot shows the RStudio interface. In the top-left, there's an R script editor window titled "Untitled1" containing the code above. To its right is the "Environment" tab of the top navigation bar. On the far right, the "Connections" tab is active, showing a dropdown menu with options like "From Text (base)...", "From Text (readr)...", "From Excel...", etc., with "From Text (readr...)" highlighted.

Below the tabs, there's a toolbar with icons for "File", "Plots", "Packages", "Help", and "Viewer". Under "File", there are buttons for "Folder", "File", "Delete", and "Rename". The "Viewer" tab is selected. The main workspace below shows a preview of the first 50 entries of the "climate\_data" dataset. The "Import Options" section at the bottom of the dialog includes fields for "Name" (set to "X000251"), "Skip" (set to 0), and "Delimiter" (set to "Whitespace"). A red box highlights the "Import Options" section and the preview area. The preview table has columns labeled 1900, 01, 01\_1, 0, 0\_1, and -5.5.

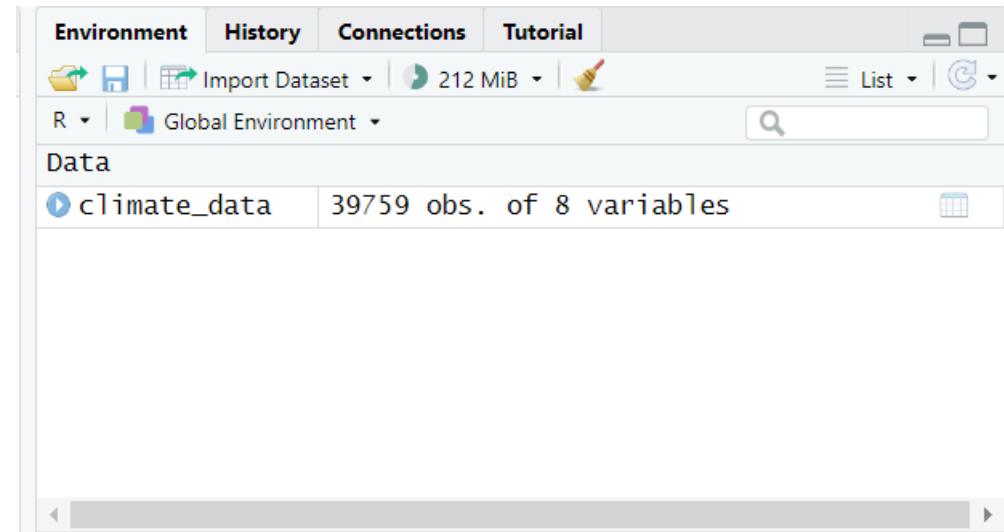
The R console at the bottom displays the same code as the script editor, with the last two lines highlighted in blue:

```
4 library(readr)
5 climate_data<- read_table("test/000251.txt")
6 View(climate_data)
```

# 3. Apply predefined function

- 3.1 Prepare data
- # Extract year and TX
- climate\_data\$year <- climate\_data[[1]]  
# Column 1 = year
- climate\_data\$tx <- climate\_data[[5]]  
# Column 5 = TX

```
9 View(climate_data)
10
11
12 climate_data$year <- climate_data[[1]]
13 climate_data$tx   <- climate_data[[5]]
14 |
```



# 3. Apply predefined function

- 3.2 Create function for yearly mean and apply
- ```
mean_annual_tx <- function(data) {
```
- ```
  data %>%
```
- ```
  group_by(year) %>%
```
- ```
  summarise(mean_tx = mean(tx, na.rm = TRUE))
```
- }
- #Apply function
- ```
annual_avg_tx <- mean_annual_tx(climate_data)
```
- # See result
- ```
print(annual_avg_tx)
```

```
16 mean_annual_tx <- function(data) {  
17   data %>%  
18   group_by(year) %>%  
19   summarise(mean_tx = mean(tx, na.rm = TRUE))  
20 }
```

The screenshot shows the RStudio interface with the 'Global Environment' tab selected. It lists the 'climate\_data' dataset and the 'mean\_annual\_tx' function.

```
22 # Apply  
23 annual_avg_tx <- mean_annual_tx(climate_data)  
24 |  
25 # See  
26 print(annual_avg_tx)  
24:1 (Top Level) ↴
```

The screenshot shows the RStudio interface with the 'Console' tab selected. It displays the command 'annual\_avg\_tx <- mean\_annual\_tx(climate\_data)' and its output, which is a tibble with columns 'year' and 'mean\_tx'. The output table has 112 rows, starting from 1900 to 1909.

year	mean_tx
1900	15.6
1901	15.8
1902	14.8
1903	15.7
1904	15.0
1905	15.6
1906	15.8
1907	14.2
1908	14.7
1909	16.0

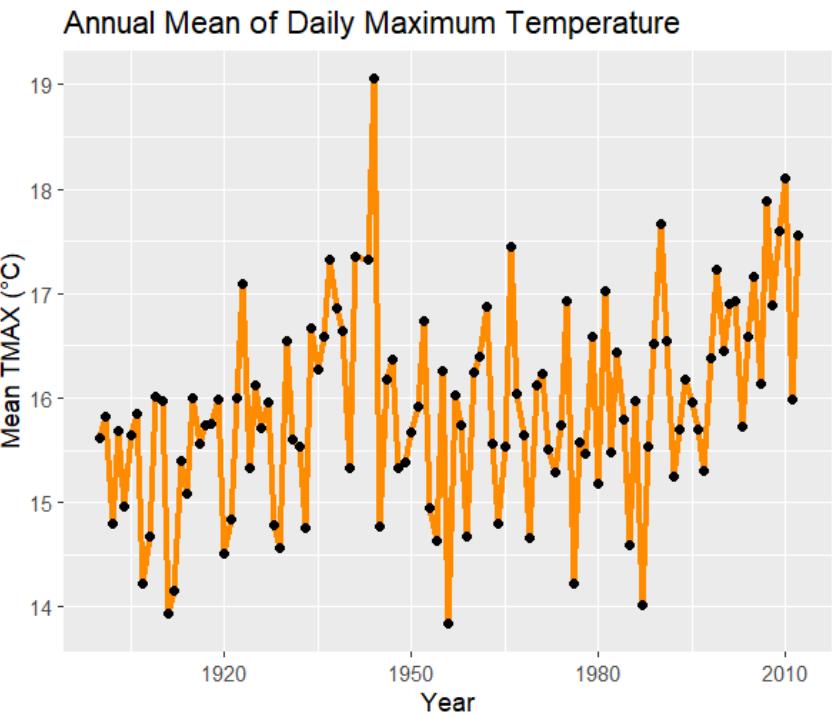
The screenshot shows the RStudio interface with the 'Global Environment' tab selected. It lists the 'annual\_avg\_tx' dataset and the 'mean\_annual\_tx' function.

# 3. Apply predefined function

- 3.2 Visualize
- library(ggplot2)
- ggplot(annual\_avg\_tx, aes(x = year, y = mean\_tx)) +
- geom\_line(color = "darkorange", size = 1.2) +
- geom\_point() +
- labs(title = "Annual Mean of Daily Maximum Temperature",  
x = "Year", y = "Mean TMAX (°C)")

```
30 ggplot(annual_avg_tx, aes(x = year, y = mean_tx)) +  
31   geom_line(color = "darkorange", size = 1.2) +  
32   geom_point() +  
33   labs(title = "Annual Mean of Daily Maximum Temperature",  
34     x = "Year", y = "Mean TMAX (°C)")  
35
```

24:1 (Top Level) ↗



---

## 4. Sectorial functions

- 1. Agriculture: Dry Spells

```
climate_data$prec <- climate_data[[4]]
```

```
count_dry_spells <- function(prec, threshold = 1, min_length = 5) {  
  rle_obj <- rle(prec < threshold) # TRUE si precipitación < 1 mm  
  sum(rle_obj$values & rle_obj$lengths >= min_length)  
}  
dry_spells <- count_dry_spells(climate_data$prec)
```

```
# Mostrar resultado  
print(paste("Number of dry spells:", dry_spells))
```



## 4. Sectorial functions

- 1. Agriculture: Dry Spells

```
climate_data$tmax <- climate_data[[5]]  
  
count_hot_days <- function(temp, threshold = 35) {  
  sum(temp > threshold, na.rm = TRUE)  
}  
  
hot_days <- count_hot_days(climate_data$tmax)  
  
  
print(paste("Number of hot days:", hot_days))
```

---

## 4. Custom functions

- rain\_mm <- function(rain\_mm) { if (rain\_mm > 50) {  
return(5) } else if (rain\_mm >= 31 && rain\_mm <= 50) {  
return(4) } else if (rain\_mm >= 21 && rain\_mm <= 30) {  
return(3) } else if (rain\_mm >= 11 && rain\_mm <= 20) {  
return(2) } else if (rain\_mm >= 0 && rain\_mm <= 10) {  
return(1) } else { return(0) }}
- climate\_data\$prec <- climate\_data[[4]]
- mm<-sapply(climate\_data\$prec ,FUN=cp\_rain\_mm )

[View](#)