Tools for biologists

Antonio Profico (antonio.profico@unipi.it)

Luca Rindi (luca.rindi@unipi.it)

How organizing an R script

A well-organized R script is essential for readability, reproducibility and collaboration with colleagues that are or not proficient in R.

1. Each script should answer one main question
2. Avoid “monster script”
3. If needed, split the workfkow into multiple scripts or sections.
    I. 01_data_cleaning_210126.R
    II. 03_data_analysis_230126.R
    III. 04_plots_150126.R

Hints:
- Separated words by “underscore” (_) (snake_case)
- Use the functions save() and load() to export and import data
- Add the date in the name of the file.

The functions save and load

Create an R studio project

Create a data frame

Save the data.frame using the function *save()*

e.g. save(yourobject, file = "path.rda")

Clean objects from the workspace

Load the object using the function *load()*

e.g. load(file = "path.rda")

How organizing an R script

4. Start with a header
5. Load packages at the beginning
6. Add customized R functions
7. Organize code into logical sections

```
# 1. Data import
# 2. Data cleaning
# 3. Exploratory analysis
# 4. Statistical analysis
# 5. Visualization
# 6. Export results
```

8. Avoid manual steps
9. Save, run, rerun

```
1  ######################################################################
2  ##### Mapping sexual dimorphism signal in the human cranium #####
3  ######################    R code      ##########################
4  ######################################################################
5
6  #load libraries
7  {⟷}
22
23 #load functions
24 {⟷}
350
351 #load data
352 {⟷}
361
362 #Figure 1 - R code
363 {⟷}
414
415 #Figure 2 - R code
416 {⟷}
444
445 #Figure 4 - R code
446 {⟷}
477
478 #Figure 5 - R code - frontal region
479 {⟷}
583
584 #Figure 6 - R code - module definition
585 {⟷}
608
609 #Figure 6 - nasal region module
610 {⟷}
698
699 #Figure 6 - frontal region module
700 {⟷}
786
787 #Figure 6 - mastoid process module
788 {⟷}
876
```

How organizing an R script

4. Start with a header
5. Load packages at the beginning
6. Add customized R functions
7. Organize code into logical sections

```
# 1. Data import
# 2. Data cleaning
# 3. Exploratory analysis
# 4. Statistical analysis
# 5. Visualization
# 6. Export results
```

8. Avoid manual steps
9. Save, run, rerun

```r
#Figure 1 - R code
{
PCA<-procSym(lset,sizeshape = TRUE)
PCscores<-PCA$PCscores
Variance<-PCA$Variance
PCx<-1
PCy<-2
xlim<-c(-1*max(abs(PCscores[,PCx])),max(abs(PCscores[,PCx])))
ylim<-c(-1*max(abs(PCscores[,PCy])),max(abs(PCscores[,PCy])))
Xlab<-paste("PC",PCx," ",round(Variance[PCx,2],2),"%",sep="")
Ylab<-paste("PC",PCy," ",round(Variance[PCy,2],2),"%",sep="")
dir.create("Figure_1")
t.test(PCscores[,1]~group)
t.test(PCscores[,2]~group)
summary(procD.lm(PCscores~group))
summary(procD.lm(PCscores~PCA$size))
summary(procD.lm(PCscores~PCA$size*group))
summary(procD.lm(PCscores[,1]~group))
summary(procD.lm(PCscores[,1]~PCA$size))
summary(procD.lm(PCscores[,2]~group))
summary(procD.lm(PCscores[,2]~PCA$size))


tiff("Figure_1/Fig. 1 - PCA_formspace.tiff",width = 1500,height = 1500,pointsize = 6,res = 300)
par(mar = c(5, 5, 5, 5))
plot(NA,xlim=extendrange(xlim),ylim=extendrange(ylim),xlab=Xlab,ylab=Ylab,asp=1,
     main="",cex.axis=1.5,cex.main=2.5,cex.lab=2,
     cex=2)
abline(v=0,lty=2,lwd=0.7)
abline(h=0,lty=2,lwd=0.7)

cols<-c("deepskyblue3","khaki2")
sex<-group
for(i in 1:2){
  sel<-which(sex==levels(as.factor(sex))[[i]])
  pchi<-sex[sel]
  pchi[pchi=="M"]<-19
  pchi[pchi=="F"]<-19
  mat<-PCscores[sel,c(PCx,PCy)]
  conv<-chull(mat)
  polygon(mat[c(conv,conv[1]),],col=adjustcolor(cols[i],alpha.f=0.3))
  points(mat,col=cols[i],pch=as.numeric(pchi),cex=1.5)
}
dev.off()
```

How organizing an R script

Start with a header

```
# 1. Data import
# 2. Data cleaning
# 3. Exploratory analysis
# 4. Statistical analysis
# 5. Visualization
# 6. Export results
```

8. Avoid manual steps

9. Save, run, rerun

```
#Figure 1 - R code
{
PCA<-procSym(lset,sizeshape = TRUE)
PCscores<-PCA$PCscores
Variance<-PCA$Variance
PCx<-1
PCy<-2
xlim<-c(-1*max(abs(PCscores[,PCx])),max(abs(PCscores[,PCx])))
ylim<-c(-1*max(abs(PCscores[,PCy])),max(abs(PCscores[,PCy])))
Xlab<-paste("PC",PCx," ",round(Variance[PCx,2],2),"%",sep="")
Ylab<-paste("PC",PCy," ",round(Variance[PCy,2],2),"%",sep="")
dir.create("Figure_1")
t.test(PCscores[,1]~group)
t.test(PCscores[,2]~group)
summary(procD.lm(PCscores~group))
summary(procD.lm(PCscores~PCA$size))
summary(procD.lm(PCscores~PCA$size*group))
summary(procD.lm(PCscores[,1]~group))
summary(procD.lm(PCscores[,1]~PCA$size))
summary(procD.lm(PCscores[,2]~group))
summary(procD.lm(PCscores[,2]~PCA$size))


tiff("Figure_1/Fig. 1 - PCA_formspace.tiff",width = 1500,height = 1500,pointsize = 6,res = 300)
par(mar = c(5, 5, 5, 5))
plot(NA,xlim=extendrange(xlim),ylim=extendrange(ylim),xlab=Xlab,ylab=Ylab,asp=1,
     main="",cex.axis=1.5,cex.main=2.5,cex.lab=2,
     cex=2)
abline(v=0,lty=2,lwd=0.7)
abline(h=0,lty=2,lwd=0.7)

cols<-c("deepskyblue3","khaki2")
sex<-group
for(i in 1:2){
  sel<-which(sex==levels(as.factor(sex))[[i]])
  pchi<-sex[sel]
  pchi[pchi=="M"]<-19
  pchi[pchi=="F"]<-19
  mat<-PCscores[sel,c(PCx,PCy)]
  conv<-chull(mat)
  polygon(mat[c(conv,conv[1]),],col=adjustcolor(cols[i],alpha.f=0.3))
  points(mat,col=cols[i],pch=as.numeric(pchi),cex=1.5)
}
dev.off()
```

What's wrong with this script?

```
library(ggplot2)
x <- read.csv("data.csv", sep=";", dec=",")
mean(x$size)
ggplot(x, aes(size, shape)) +
  geom_point()
x <- x[x$size > 0, ]
library(dplyr)
write.csv(x, "clean_data.csv")
# PCA
pca <- prcomp(x[,3:6], scale.=TRUE)
summary(pca)
setwd("/Users/student/Desktop/project")
library(vegan)
```

What's wrong with this script?

```
library(ggplot2)
x <- read.csv("data.csv", sep=";", dec=",")
mean(x$size)
ggplot(x, aes(size, shape)) +
  geom_point()
x <- x[x$size > 0, ]
library(dplyr)
write.csv(x, "clean_data.csv")
# PCA
pca <- prcomp(x[,3:6], scale.=TRUE)
summary(pca)
setwd("/Users/student/Desktop/project")
library(vegan)
```

No header or description
Working directory set in the middle of the script
Libraries loaded in different places
Data cleaning after analysis
Unclear object names (x)
No section structure
Hard-coded file paths

In class exercise

Load the data and prepare a clean and organized version of this script

An introduction to programming

R is an interpreted language; instructions are executed one at a time.

This allows us to:

1. Repeat a group of instructions (*for*, *while*, *repeat* loops)

2. Execute instructions in an alternative way (conditional statements)

Examples:

Ex. 1 Copy – rename – paste all files contained within a folder

Ex. 2 Data simulation (coin toss, bootstrap)

Ex. 3 Meta-analysis (e.g., systematic literature review, word cloud)

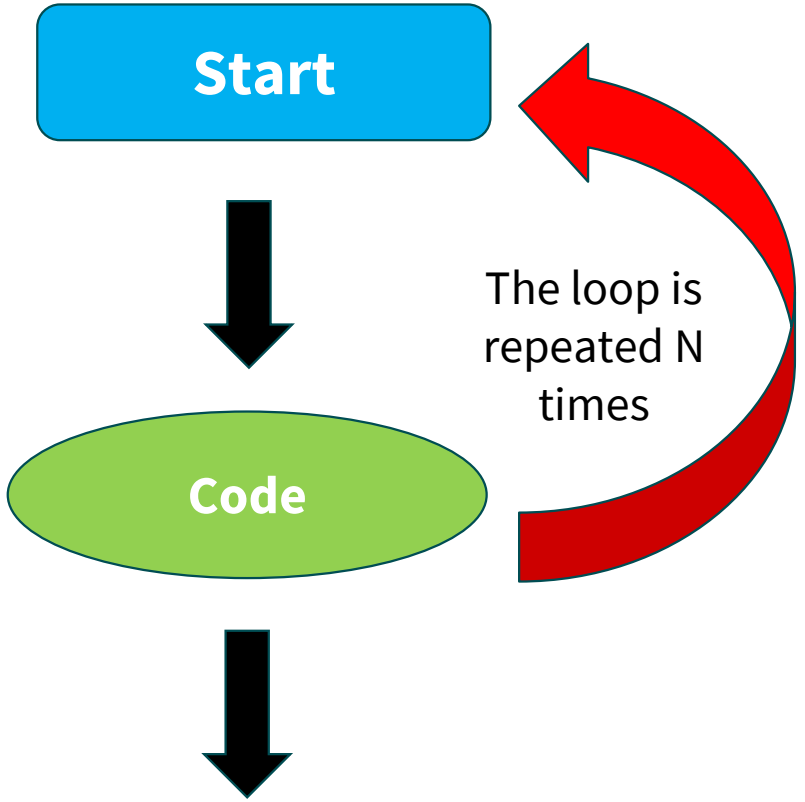
Create a loop

Create an empty R object (eg. extractions <- NULL)

Define how many times the loop will repeat (eg. 100, 1000, etc.)

Insert code to be repeated
#Flip a coin
Flip<-sample(c(“Heads", “Tails")

#Assign the result of the extraction, at each iteration, in the object extractions <- c(extractions, extractions)

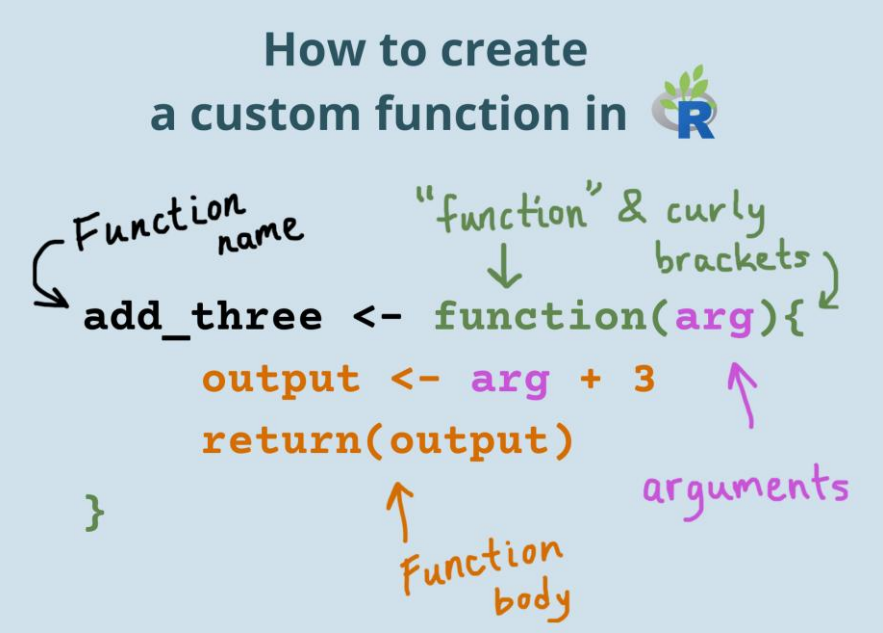


How writing an R function?

Why writing a function?

1. The function is not embedded in an existing R package
2. Avoid repeating the same code
3. Make analysis reproducible and repeatable
4. Convert workflows into reusable tools

How writing an R function?



Basic structure of a function

1. Use function()
2. Define input arguments
3. Write the R code body
4. Return the output

```
add_three <- function(arg) {
output <- arg + 3
return(output)
}
```

In class exercise

Load the data and prepare a clean and organized version of this script

How writing an R function?

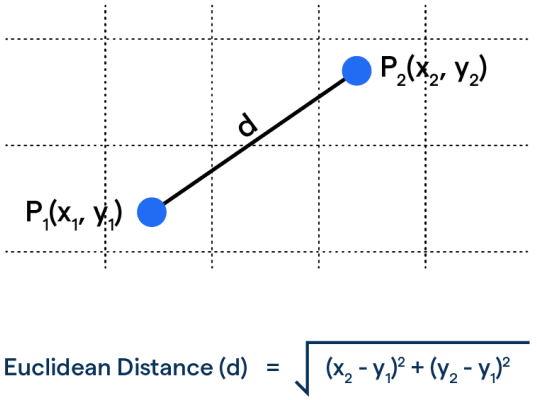
Goal: Write an R function that calculates the Euclidean distance between two points in a 2D space.

Hints:

1. A point in 2D space is defined by its coordinates: $P=(x,y)$
2. The Euclidean distance between two points

$$P_1 = (x_1, y_1) \text{ and } P_2 = (x_2, y_2)$$

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$



3. use sqrt() and ^2

The function locator

locator() is an interactive function that allows the user to click on a plot and retrieve x and y coordinates of the clicked points

1. locator(n) n = number of points to select
2. The output is a list with two elements:
    1. $x -> x coordinates
    2. $y -> y coordinates

```
plot(c(1:10),c(1:10))
pts<-locator(3)
pts$x
pts$y
```

Write a function - exercise

Write an R function to measure the distance between two points in an interactive way

1 – Generate a matrix with 25 rows and 2 columns

2 – Create a plot

3 – Use the function locator to select 2 points

4 – Calculate the Euclidean distance

5 – Return and print the value