

Exercises for programming with functions

Timothée Bonnet

March 14, 2019

Contents

1	Simplest functions	2
	Exercise 1 <i>Build a first function</i>	2
	Exercise 2 <i>Multiple values in output</i>	2
	Exercise 3 <i>Turning a piece of code into a function</i>	3
	Exercise 4 <i>Recover the central-limit theorem</i>	5
2	Funny things	7
2.1	The dot-dot-dot	7
	Exercise 5 <i>Understand the ...</i>	7
2.2	The <<- operator	8
	Exercise 6 <i>Understand the <<-</i>	8
2.3	Recursive functions	9
	Exercise 7 <i>What does this function do?</i>	9

Exercise 8 <i>Find a mistake in a family tree</i>	11
---	----

1 Simplest functions

* Exercise 1 Build a first function

Write a function that return the product of three arbitrary numbers together ($x*y*z$) provided by the user.

** Exercise 2 Multiple values in output

Write a function that return the product of three arbitrary numbers together ($x*y*z$) as well as their sum ($x+y+z$).

**** Exercise 3 Turning a piece of code into a function**

In a previous session we wrote the following piece of code to simulate p-values for a random effect in a mixed model:

```
RandomVariance <- 0
sampsize <- 500
nbblocks <- 30
pvals <- vector(length = 1000)
altpvals <- vector(length = 1000)
for (i in 1:1000)
{
  x <- rnorm(sampsize, mean = 4, sd=0.25)
  block <- sample(x = 1:nbblocks, size = sampsize, replace = TRUE)
  blockvalues <- rnorm(n = nbblocks, mean = 0, sd = sqrt(RandomVariance))
  y <- 8 - x + blockvalues[block] + rnorm(sampsize, 0, 1)
  dat <- data.frame(response = y, predictor = x, block=block)
  lm0 <- lm(response ~ 1 + predictor, data=dat)
  lmm0 <- lmer(response ~ 1 + predictor + (1|block), data=dat )
  (LRT0 <- anova(lmm0, lm0)) #mixed model must come first!
  pvals[i] <- LRT0$`Pr(>Chisq)`[2] # the p-value
  altpvals[i] <- 1-pchisq(LRT0$Chisq[2], 0.5) # a better p-value
}
pvals
altpvals
```

Turn this into a function where the arguments will let you control:

- The value of the simulated variance (RandomVariance)

- The number of simulations (above it is fixed to 1000)
- (optionally) The sample size and number of random levels (blocks)

And output:

- A vector of standard p-values from the LRT
- (optionally) a vector of “alternative” p-values (from the hand-made Chi-square)
- (optionally) histograms of the distribution of p-values

**** Exercise 4 Recover the central-limit theorem**

To simplify, the central-limit theorem states that the sum of independent random distributions tends towards a normal distribution even if the original distributions are non-normal. Let's write a function that shows that by randomly drawing distributions with random parameters, summing them, and comparing the sum to a Gaussian distribution of same mean and variance.

We can start from the following piece of code:

```

#how many numbers per distribution:
n=2000

# where we will save the sum of distributions:
output <- vector(length = n)

# three possible distributions to draw from:
rddistri <- list(rnorm,runif,rpois)

# drawing one number at random to tell which distribution we use:
choosedistri <- sample(1:length(rddistri), size = 1)

# a trouble is that the different distributions do not take the
#same number or type of arguments, so we need to have a special
#piece of code for each of them. For example, for the normal
#distribution (rnorm):
if(choosedistri==1)
{
  # Decide how you want to select mean and sd:
  output <- output + rddistri[[choosedistri]](n, mean=, sd=)
}
...

```

We may want to use a for loop. We could include also a plot or a statistical test (e.g., `shapiro.test`).

2 Funny things

2.1 The dot-dot-dot

* Exercise 5 Understand the ...

Consider these two functions:

```
halfmean1 <- function(x)
{
  mean(x)/2
}
halfmean2 <- function(x,...)
{
  mean(x,...)/2
}
```

They behave the same way in the first case, but differently in the second:


```
somedata <- c(10, 25, NA)

halfmean1(somedata)

## [1] NA

halfmean2(somedata)

## [1] NA

halfmean1(somedata, na.rm=TRUE)

## Error in halfmean1(somedata, na.rm = TRUE): unused argument (na.rm = TRUE)

halfmean2(somedata, na.rm=TRUE)

## [1] 8.75
```

What does ...do?

2.2 The <<- operator

* Exercise 6 Understand the <<-

We create two almost identical functions, `f0` and `f1` and add their output to an object `x`. Compare the output and explain what happens. What does `<<-` mean?

```
f0 <- function(x=2){  
  x <- x  
  y <- x+2  
  return(y)  
}
```

```
f1 <- function(x=2){  
  x <- x  
  y <- x+2  
  return(y)  
}
```

```
x <- rnorm(1000)  
f0()+x
```

```
x <- rnorm(1000)  
f1()+x
```

2.3 Recursive functions

**** Exercise 7 What does this function do?**

Consider the following function and try to understand what it does. (code by Dr. Koen van Benthem)

```

tree<-function(x,y,l,dir,n,nmax){
  if(n==0){
    return()
  }
  pos<-round(runif(1,1,199))
  colour=rainbow(200,start=0.2,end=0.6,v=0.6)[pos]
  lines(c(x,x+l*sin(dir)),c(y,y+l*cos(dir)),
        lwd=20*(n/nmax),col=colour)
  branches<-round(runif(1,2,4))
  for(i in 1:branches){
    angle<-dir+(-pi/6)+(pi/3)*(i-1)/(branches-1)
    l2<-runif(1,0.7,0.85)*l
    tree(x+l*sin(dir),y+l*cos(dir),l2,angle,n-1,nmax)
  }
}

```

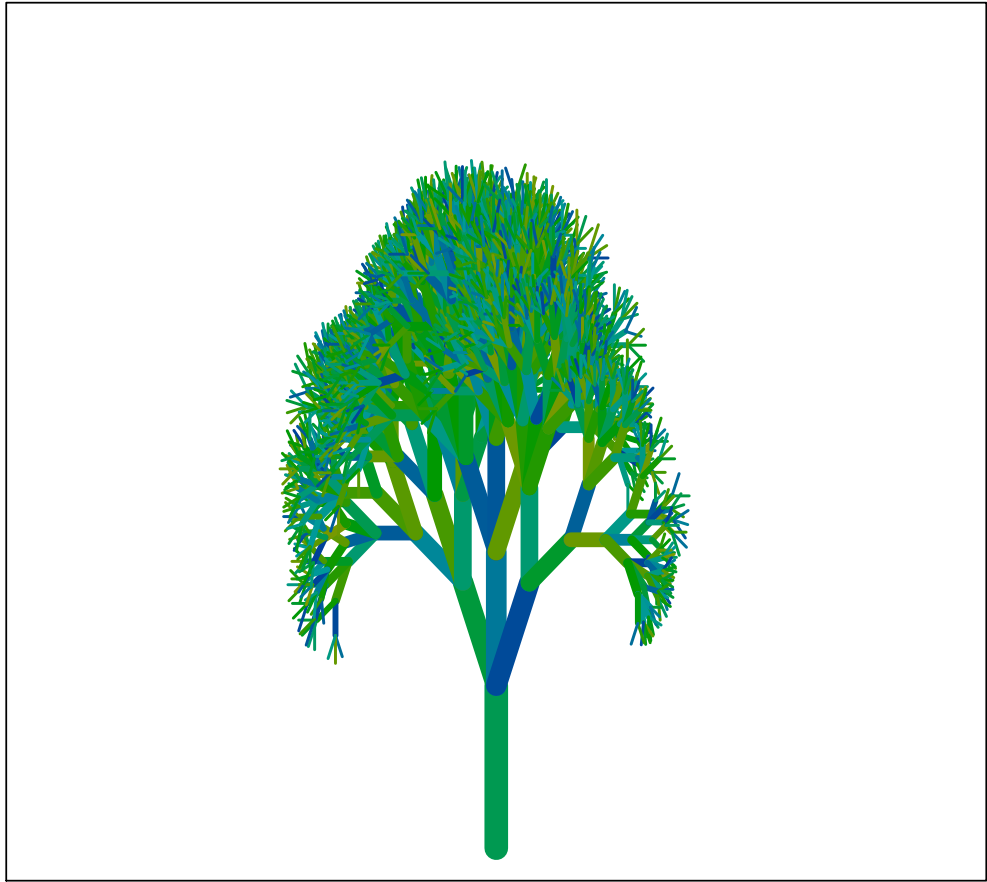
You can run the function and see if you had guessed right:

```

plot(0,0,type="n",xlim=c(-10,10),ylim=c(0,10),
     xaxt="n",yaxt="n",ylab="",xlab="")

tree(0,0,2,0,8,12)

```



What is special about this function? Why should you be careful not to give large values to the parameters `n`?

***** Exercise 8 Find a mistake in a family tree**

The file `wrongpedigree.csv` contains a pedigree (a family tree) containing a mistake: a function we tried to apply on this dataset informed us that the family tree was impossible without more information. We suspect that we have assigned an individual as its own ancestor (this can happen with genetic reconstruction of parentage). Use a

self-referencing function to scan the tree and find where the problem is.