Introduction to Functional Programming in R

Daniel Emaasit

February 27, 2016

Note: This is from *Functional Programming in R* <https://github.com/Emaasit/Functional-Programming-in-R> by Daniel Emaasit

## Introduction

R is a functional programming language, meaning that it has tools for creating and manipulating functions.

## First class Functions

R has first class functions. You can do anything with functions that you can do with vectors, like:  
\* Assign them to variables \* Store them in lists \* Pass them as arguments \* Create them inside functions \* Return them as a result of a function

## Functional Components

All R functions are made up of 3 components:

* the body
* the arguments/formals
* the environment

## By printing the function  
f <- function(n, mean, sd){  
 ## Generate n random normal observations with mean = mean and std dev = sd  
 rnorm(n, mean, sd)  
}  
  
print(f) ## if environment is not shown, it means it was created in the Global Env

## function(n, mean, sd){  
## ## Generate n random normal observations with mean = mean and std dev = sd  
## rnorm(n, mean, sd)  
## }

body(f)

## {  
## rnorm(n, mean, sd)  
## }

formals(f)

## $n  
##   
##   
## $mean  
##   
##   
## $sd

environment(f)

## <environment: R\_GlobalEnv>

attributes(f)

## $srcref  
## function(n, mean, sd){  
## ## Generate n random normal observations with mean = mean and std dev = sd  
## rnorm(n, mean, sd)  
## }

class(f)

## [1] "function"

f(10, 1, 1)

## [1] 1.47668699 0.71672888 -0.97502539 2.72914763 0.05783114  
## [6] 0.40099066 1.00783939 0.55827609 -0.05060957 0.80287827

### Primitive Functions

Functions (only found in base package) whose formals(), body(), and environment() are all NULL. Like:

sum

## function (..., na.rm = FALSE) .Primitive("sum")

formals(sum)

## NULL

body(sum)

## NULL

environment(sum)

## NULL

## Scoping

The set of rules that governs how R looks up the values of a symbol. Two types of scopint:

* Lexical Scoping: Looks up symbol values based on how functions were nested when they were created, not how they are nested when they are called
* Dynamic Scoping: Used in select functions to save typing during interactive analysis

Four basic principles of R’s Lexical Scoping

* name masking
* functions vs variables
* a fresh start

j <- function() {  
 if (!exists("a")) {  
 a <- 1  
 } else {  
 a <- a + 1  
 }  
 print(a)  
}  
  
j()

## [1] 1

* dynamic lookup

**Closures**: Functions created by other functions

## Every Operation is a function call

## Function Arguments

## Special Calls

## Return Values