Writing Functions

R Programming Structures

Gaston Sanchez

R Coding Compendium

Donate

About

How to write functions?

- Always start simple with test toy-values
- Work first on what will be the body of the function
- Check out each step of the way
- Don't try to do much at once
- Create the function (i.e. encapsulate the body) once everything works
- Don't write long functions: write short / small functions (preferably less than 10 lines of code)

Variance Function Example

R has the `var()` function, but for sake of illustration let's ignore this.

The sample variance is given by the following formula:

$$var(x) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

- x a variable
- n number of values in x
- $\bar{x}$ mean of x-values

Variance Function Example

```r
# start simple
x <- 1:5

# get working code
sum((x - mean(x))^2) / (length(x) - 1)
```

```
## [1] 2.5
```

```r
# test it: compare it to var()
var(1:5)
```

```
## [1] 2.5
```

Variance Function Example

```r
# encapsulate your code
variance <- function(x) {
  sum((x - mean(x))^2) / (length(x) - 1)
}

# check that it works
var(1:10)
```

```
## [1] 9.166667
```

Variance Function Example

```
# then consider less simple cases
variance(runif(10))
```

```
## [1] 0.07852319
```

```
variance(rep(0, 10))
```

```
## [1] 0
```

```
variance(c(1:9, NA))
```

```
## [1] NA
```

Variance Function Example

```r
# adapt it gradually
variance <- function(x, na.rm = FALSE) {
  if (na.rm) {
    x <- x[!is.na(x)]
  }
  sum((x - mean(x))^2) / (length(x) - 1)
}

variance(c(1:9, NA), na.rm = TRUE)
```

```
## [1] 7.5
```

Writing Functions

When writing functions:

- Choose meaningful names of functions
- Preferably a verb
- Choose meaningful names of arguments
- Think about the users (who will use the function)
- Think about extreme cases
- If a function is too long, maybe you need to split it

Choosing names of functions

```r
# vaoid this
f <- function(x, y) {
  x + y
}

# this is better
add <- function(x, y) {
  x + y
}
```

Choosing names of functions

Give meaningful names to arguments:

```
# avoid this
area_rect <- function(x, y) {
  x * y
}

# this is better
area_rect <- function(length, width) {
  length * width
}
```

Names of functions

Even better: give default values (whenever possible)

```
area_rect <- function(length = 1, width = 1) {
  length * width
}

# default
area_rect()

# specifying argument values
area_rect(length = 10, width = 2)
```

Meaningful Names to Arguments

Avoid this:

```
# what does this function do?
ci <- function(p, r, n, ti) {
  p * (1 + r/p)^(ti * p)
}
```

This is better:

```
# OK
compound_interest <- function(principal, rate,
                              periods, time) {
  principal * (1 + rate/periods)^(time * periods)
}
```

Meaningful Names to Arguments

```r
# names of arguments
compound_interest <- function(principal = 1, rate = 0.01,
                              periods = 1, time = 1) {
  principal * (1 + rate/periods)^(time * periods)
}

compound_interest(principal = 100, rate = 0.05,
                  periods = 5, time = 1)

compound_interest(rate = 0.05, periods = 5,
                  time = 1, principal = 100)

compound_interest(rate = 0.05, time = 1,
                  periods = 5, principal = 100)
```

Documenting Functions

Documenting Functions

Also add a short description of what the arguments should be like. In this case, the description is outside the function

```
# function for adding two numbers
# x: number
# y: number
add <- function(x, y) {
  x + y
}
```

Documenting Functions

In this case, the description is between `<-` and `function()`

```
add <-
  # function for adding two numbers
  # x: number
  # y: number
  function(x, y) {
  x + y
}
```

Documenting Functions

In this case, the description is inside the function

```
add <- function(x, y) {
  # function for adding two numbers
  # x: number
  # y: number
  x + y
}
```

Documenting Functions

In this case, the description is inside the function

```
# description of arguments
compound_interest <- function(principal = 1, rate = 0.01,
                              periods = 1, time = 1)
{
  # principal = Principal Amount
  # rate = Annual Nominal Interest Rate as a decimal
  # time = Time Involved in years
  # periods = number of compounding periods per unit time
  principal * (1 + rate/periods)^(time * periods)
}
```

Roxygen Comments

One interesting option to document functions is by using **roxygen comments**

```r
#' @title Compound Interest
#' @description Calculates future value with compound interest
#' @param principal Principal Amount
#' @param rate Annual Nominal Interest Rate as a decimal
#' @param time Time Involved in years
#' @param periods Number of compounding periods per unit time
#' @return future value
compound_interest <- function(principal = 1, rate = 0.01,
                              periods = 1, time = 1)
{
   principal * (1 + rate/periods)^(time * periods)
}
```

Donation

If you find any value and usefulness in this set of slides, please consider making a one-time donation in any amount (via paypal). Your support really matters.

Donate

https://www.paypal.com/donate?business=ZF6U7K5MW25W2&currency_code=USD