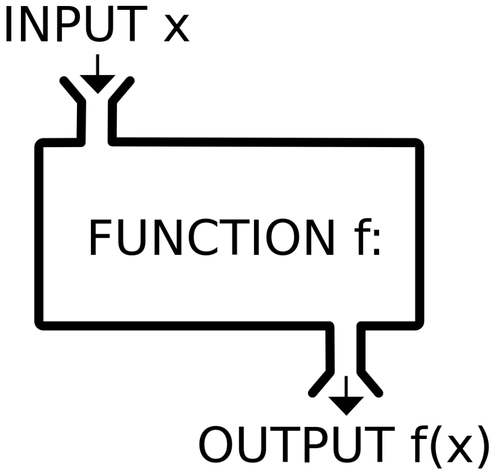
Defining and using functions in R

https://github.com/mbjoseph/r-intro-functions

Max Joseph

3/8/2017


INPUT x
FUNCTION f:
OUTPUT f(x)

A simple function

```r
c(1, 2, 3, 4, 5)
```

```
## [1] 1 2 3 4 5
```

Functions are objects too!

```
c
```

```
## function (...)  .Primitive("c")
```

Primitive functions

Functions written in R

```
sd
```

```
## function (x, na.rm = FALSE)
## sqrt(var(if (is.vector(x) || is.factor(x)) x else as.dou
##     na.rm = na.rm))
## <bytecode: 0x7fc4a3231870>
## <environment: namespace:stats>
```

Example: temperature conversion

```
x <- 30
y <- ((x - 32) * (5 / 9)) + 273.15
y
```

```
## [1] 272.0389
```

Example: temperature conversion

```
fahr_to_kelvin <- function(fahr) {
  kelvin <- ((fahr - 32) * (5 / 9)) + 273.15
  kelvin
}

fahr_to_kelvin(30)
```

```
## [1] 272.0389
```

Name

Arguments

```
fahr_to_kelvin <- function(fahr) {
  kelvin <- ((fahr - 32) * (5 / 9)) + 273.15
  kelvin
}
```

Body

```
There are only two hard things in Computer Science:
  cache invalidation and naming things.

-- Phil Karlton
```

What’s in a (function) name?

```
f()
my_func()
t_funk()
f2k()
convert_temperature()
fahr_to_kelvin()
```

Body

What your function **does**

```
fahr_to_kelvin <- function(fahr) {
  kelvin <- ((fahr - 32) * (5 / 9)) + 273.15
  kelvin
}
```

3 weird tricks to a great function body your physician doesn't want you to know!

1. Express intent
2. Be nice to humans
3. Self-contain your functions

1. Express your intent with meaningful names

```
fahr_to_kelvin <- function(fahr) {
  kelvin <- ((fahr - 32) * (5 / 9)) + 273.15
  kelvin
}
```

not

```
fahr_to_kelvin <- function(x) {
  y <- ((x - 32) * (5 / 9)) + 273.15
  y
}
```

2. Document what the function does for human readers

```
fahr_to_kelvin <- function(fahr) {
  # Convert temperature in fahrenheight to kelvin
  # args: fahr (numeric) - temp. in fahrenheight
  # returns: kelvin (numeric) - temp. in kelvin
  kelvin <- ((fahr - 32) * (5 / 9)) + 273.15
  kelvin
}
```

3. Make your functions self-contained

```
offset <- 273.15

fahr_to_kelvin <- function(fahr) {
  # Convert temperature in fahrenheight to kelvin
  # args: fahr (numeric) - temp. in fahrenheight
  # returns: kelvin (numeric) - temp. in kelvin
  kelvin <- ((fahr - 32) * (5 / 9)) + offset
  kelvin
}
```

^Bad

Environments

R associates each object with an **Environment**

By default, objects → **Global environment**

(demo)

Function environments

```
fahr_to_kelvin <- function(fahr) {
  kelvin <- ((fahr - 32) * (5 / 9)) + 273.15
  kelvin
}

fahr_to_kelvin(fahr = 100)
```

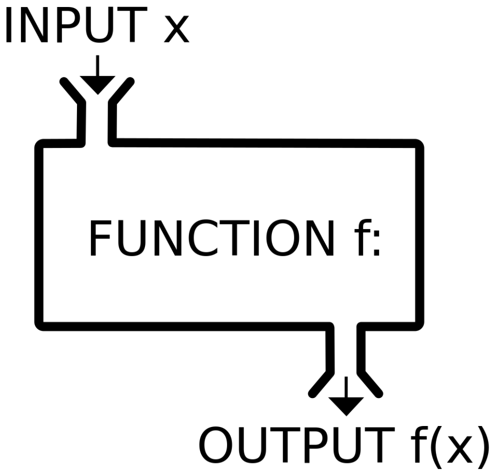
```
## [1] 310.9278
```

```
kelvin
```

```
## Error: object 'kelvin' not found
```

Self-contained functions

1. Act like functions



Self-contained functions

2. Are robust to the state of the global environment

```
offset <- 273.15

fahr_to_kelvin <- function(fahr) {
  # Convert temperature in fahrenheight to kelvin
  # args: fahr (numeric) - temp. in fahrenheight
  # returns: kelvin (numeric) - temp. in kelvin
  kelvin <- ((fahr - 32) * (5 / 9)) + offset
  kelvin
}
```

Review

Function parts:

- name
- arguments
- body

Best practices:

- use a good name
- document your function
- contain your function

But wait, there's more!

We haven't covered the **why** and **when** yet.

Short version:

- DRY (do not repeat yourself)
- modularity
- ease of maintenance