– Tidyverse design guide

https://twitter.com/mauro_lepore

Avoid hidden arguments

Hidden arguments make code harder to reason about, because to correctly predict the output you also need to know some other state

```
y <- 1
add <- function(x) {
  x + y
}
add(1)
#> [1] 2

y <- 10 ## It is hard to keep track of this

add(1)
#> [1] 11
```

Functions are easier to understand if the results depend only on the values of the inputs

How can I remediate the problem?

If you have an existing function with a hidden input:

1. Make sure the input is an explicit option.
2. Make sure it’s printed.

For example, take `prepare_data()`

The output depends on `data`, but it is hidden.

```
prepare_data <- function() {
  data <- read.csv(path)
  data[1:2, 1:2]
}

path <- tempfile()
readr::write_csv(mtcars, path)

prepare_data()
#>   mpg cyl
#> 1  21   6
#> 2  21   6
```

1. `prepare_data()` gains the explicit argument `data`

```
prepare_data <- function(data = read.csv(path)) {
  data[1:2, 1:2]
}

prepare_data()
#>   mpg cyl
#> 1  21   6
#> 2  21   6
```

2. prepare_data() now prints data

```
prepare_data <- function(data = read.csv(path)) {
  if (missing(data)) {
    message(
      "Using `data` with names: ", paste(names(data), collapse = ", ")
    )
  }
  data[1:2, 1:2]
}

prepare_data()
#> Using `data` with names: mpg, cyl, disp, hp, drat, wt, qsec, vs, am, gear, carb
#>   mpg cyl
#> 1  21   6
#> 2  21   6

prepare_data(read.csv(path))
#>   mpg cyl
#> 1  21   6
#> 2  21   6
```

But **data** should be supplied

Data arguments provide the core data. They are required, and are usually vectors and often determine the type and size of the output. Data arguments are often called `data`, `x`, or `y` – tidyverse design guide.

```
prepare_data <- function(data) {
  data[1:2, 1:2]
}

try(prepare_data())
#> Error in prepare_data() : argument "data" is missing, with no default

data <- read.csv(path)
prepare_data(data)
#>   mpg cyl
#> 1  21   6
#> 2  21   6
```

Some functions do need to depend on external state …

A function has hidden arguments when it returns different results with the same inputs in a surprising way

Surprising

```
getOption("stringsAsFactors")
#> [1] TRUE
data.frame(x = "a")$x
#> [1] a
#> Levels: a

old_options <- options(stringsAsFactors = FALSE)
on.exit(old_options)

getOption("stringsAsFactors")
#> [1] FALSE
data.frame(x = "a")$x
#> [1] "a"
```

Global options should not affect computation.

Not surprising

`read_csv(path)` depends not only on `path` but also on the contents of the file, but that is not surprising.

```
library(readr)
path <- tempfile()

write_csv(mtcars, path)

names(read_csv(path))
#>  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
#> [11] "carb"

write_csv(iris, path)

names(read_csv(path))
#> [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```