Clean Code Examples

Anthea Alberto

2023-02-17

Principles of clean code

Code for people, not machines

"Always assume that others will read your source code." (Mayer, 2022, p.56)

```
# Bad!
xxx <- 10000
yyy <- 0.1
zzz <- 1:10

for (iii in zzz) {
  print(xxx*(1+yyy)^iii)
}
```

```
## [1] 11000
## [1] 12100
## [1] 13310
## [1] 14641
## [1] 16105.1
## [1] 17715.61
## [1] 19487.17
## [1] 21435.89
## [1] 23579.48
## [1] 25937.42
```

While it does work (i.e. it produces a resonable output) is difficult to understand what the code above does.

```
# Better:
investements <- 10000
yearly_return <- 0.1
years <- 1:10

for (year in years) {
  print(investements*(1+yearly_return)^year)
}
```

```
## [1] 11000
## [1] 12100
## [1] 13310
## [1] 14641
## [1] 16105.1
## [1] 17715.61
## [1] 19487.17
## [1] 21435.89
```

```
## [1] 23579.48
## [1] 25937.42
```

This is better! Giving the variables proper names makes the purpose of the code and its output easier to understand.

Use the right names

This principle is similar to the one above – it's a lot easier to make sense of code when variables, data frames and functions have descriptive names. To use an example from *The Art of Clean Code*, say you want to program a currency converter. The code in the chunk below is taken from StackOverflow (slightly amended to reflect conversion rates on Jan 3rd, 2023 and include CHF).

```
currencyCon <- function(x, from = "USD", to = "EUR"){
  # assign values: 1 usd in each currency
  values <- c(1.000000, 0.945799, 0.831797, 130.411928, 0.934267)
  # names attribute
  names(values) <- c("USD", "EUR", "GBP", "YEN", "CHF")
  # calculate (convert from into USD, and divide to)
  values[to] / (values[from] / x)
}
```

```
# Testing
currencyCon(1, "USD", "EUR")
```

```
##      EUR
## 0.945799
```

```
currencyCon(1, "EUR", "EUR")
```

```
## EUR
##   1
```

```
currencyCon(1, "GBP", "YEN")
```

```
##      YEN
## 156.7834
```

```
currencyCon(100, "CHF", "USD")
```

```
##      USD
## 107.0358
```

currencyCon is a more meaningful name than just con() or fun() or something along those lines, because it concisely describes what the function does.
Now let's say you want a simpler function that transforms USD to EUR. The way it is done in the chunk below works, but is not ideal

```
dollar_to_euro <- function(x){
  x*0.945799
}
```

First, the name is not unambiguous: *dollar* will likely be associated with USD in most of the world, but Canadians, Australians and people in Hong Kong among others also use a currency called dollar, which makes the name problematic. It's better to use something like usd_to_eur().
Furthermore, it's discouraged to use numbers in functions, but rather define things like conversion rates beforehand. This will make changing it later easier, because you won't have to go through the entire function to find the number to adjust.

```
conv_rate <- 0.945799

usd_to_euro <- function(x){
  x*conv_rate
}

usd_to_euro(1)
```

```
## [1] 0.945799
```

conv_rate is not exactly unambiguous, so the chunk above could be improved further still.

Adhere to standards and be consistent

If you go back to the code chunks above, you'll notice some inconsistencies in terms of naming conventions. In particular, the conversion function from StackOverflow uses a capital C to distinguish the two words from each other (*currency* and *Con*), whereas in the other chunks I've primarily used underscores for variable and function names.
Different people prefer different