R style guide

Andreas Alfons and Pieter Schoonees

Clean code is easier to read and understand, especially for people other than the author. Moreover, it is easier to find bugs in clean code.

The two code examples below are functionally identical but very different in coding style. Where is it easier to find the bug?

```
nrobs<-function(x){if(is.null(dim(x)))length(x)}else{nrow(x)}}
```

```
get_nr_observations <- function(x) {
  if (is.null(dim(x)))
    # vector
    length(x)
  } else {
    # matrix or data frame
    nrow(x)
  }
}
```

Give meaningful names to objects, functions and files

Aim to use short but meaningful names (which is not easy). Either use _ to separate words or use camel case, but stick to your choice and never mix the two. In general, we prefer to use _ to separate words.

- Object names should describe what information is stored.
- Function names should start with a verb wherever suitable.
- File names should end in `.R`.

Object names

```
# Good:
nr_products <- nrow(products)
model_fit <- lm(sales ~ ., data = products)

# Also good:
nrProducts <- nrow(products)
modelFit <- lm(sales ~ ., data = products)

# Bad:
nrproducts <- nrow(products)
mf <- lm(sales ~ ., data = products)

# Also bad:
nr_products <- nrow(products)
modelFit <- lm(sales ~ ., data = products)
```

Function names

```
# Good:
find_next <- function(clients) {
  # some stuff
}

# Also good:
findNext <- function(clients) {
  # some stuff
}

# Bad:
tmpfun <- function(clients) {
  # some stuff
}
```

File names

```
# Good:
predict_sales.R
utility_functions.R

# Also good:
predictSales.R
utilityFunctions.R

# Bad:
script.R
misc.R
```

Use spaces

Put spaces around operators. Always put a space after a comma and never before.

```
# Good:
products[sales <= 1000, ]
products[, "sales"]

# Bad:
products[sales<=1000,]
products[ ,"sales"]
```

Indent properly

In loops or control flow statements, indent your code by *two spaces*. Put the opening curly brace on the same line as the keyword and the closing curly brace on a new line.

```
# Good:
if (total_sales > 1000000) {
  result <- "great"
} else if (total_sales > 500000) {
  result <- "ok"
} else {
  result <- "disappointing"
```

```
}

# Bad:
if(total_sales > 1000000){ result <- "great"
}else
if (total_sales > 500000) {
    result <- "ok"
}
  else {
result <- "disappointing" }
```

Avoid long lines of code

Aim to limit your code to *80 characters* per line. This should allow other people to comfortably read your code without scrolling sideways in their editor.

Use comments

If you need to think about how to write a piece of code, that piece of code needs a comment. Also explain *why* the code works, not just what it does.