Style Guide for R

DataCamp

The main purpose of the DataCamp's style guidelines is to build out a course library that has a consistent look and feel for students.

The DataCamp style guide is heavily inspired by the style guide created by Hadley Wickham. It shows small differences to better match educational needs, and introduces some new elements that are specifically relevant for content in the DataCamp format.

Notation and naming

Object names

- Variable names and function names are lowercase.

```
# good
datacamp

# bad
DataCamp
```

- Use an underscore (_) to separate words within a name.

```
# good
learn_r_tutorial

# bad
LearnRTutorial
learn.r.tutorial
```

- Variable names should be nouns.
- Function names should be verbs.
- Aim for concise and meaningful names.
- Avoid using names of existing functions and variables. Else, this can confuse new R enthusiasts.

Syntax

Assignment

- Use `<-` for assignment, not `=`.

```
# Good
datacamp <- "Cool"

# Bad
datacamp = "Not Cool"
```

Spacing

- Place spaces around all infix operators (`=`, `+`, `-`, `<-`, etc.).
- Place spaces around `=` when used in function calls.
- Always put a space after a comma, and never before (just like in regular English).
- Exception: `:`, `::`, and `:::` do not need spaces.

```
# Good
average <- mean(feet / 12 + inches, na.rm = TRUE)
x <- 1:10
base::get

# Bad
average<-mean(feet/12+inches,na.rm=TRUE)
x <- 1 : 10
base :: get
```

- Place a space before left parentheses, except in a function call.

```
# Good
if (debug) do(x)
plot(x, y)

# Bad
if(debug)do(x)
plot (x, y)
```

- No spaces around code in parentheses or square brackets (unless there's a comma, see above).

```
# Good
mtcars[5, ]

# Bad
mtcars[5,]
```

Curly braces

- Opening curly braces never continue on their own line and are always followed by a new line.
- Closing curly braces always continue on their own line, unless followed by an else statement.

```
# Good
if (grade > 10) {
  message("Student passed")
}

# Bad
if (grade>10)
{ message("Student passed") }
if (grade>10) { message("Student passed") }
```

It's ok to leave very short statements on the same line:

```
if (y < 0 && debug) message("Y is negative")
```

Look and feel of Interactive Content

Writing Style

- Use American English.
- Correct punctuation should be present in: Assignment, Hint, Instructions, Submission Correctness Tests.
- Use the "you"-form to give chapters a personal touch. Limit the we-ing.
- Do a double check on spelling mistakes.

Assignment, Instructions, Hint

- R code is placed between backticks (e.g. `r_code`, `r_function_name()`) inside normal text.
- Package names are also written in code style, using backticks.
- Functions are written with `()` and inside backticks: `r_function_name()`.
- Blocks of code are formatted using three backticks:

````
# Good
```
create_code_block <- function() {
  print("OK")
}
```

# Bad
`
createcodeblock <- function() {
  print("OK")
}
`
````

- Emphasise non-code words via *italics*. Use **bold** to introduce new concepts.
- Place mathematical expressions between `$` signs. It will be compiled as LaTeX output.

```
# Good
$ a_1 = b_1 + c_1 $

# Bad
a_1 = b_1 + c_1
```

Sample Code and Solution Code

- Use no `;` at the end of a line.
- Use double quotes for R strings `"`.

```
# Good
authors = c("Ross Ihaka", "Robert Gentleman")

# Bad
authors = c('Ross Ihaka', 'Robert Gentleman');
```

- The first argument shall not be called by its name, typically other arguments should be named.

```
# Good
points(point, cex = .5, col = "dark red")
```

```
# Bad
points(x = point, cex = .5, col = "dark red")
```

- The sample code should contain three underscores as placeholders.

```
# Good
# Assign 5 to x
x <- ___

# Bad
# Assign 5 to x
x <- _

# Also bad
x <-
```

Commenting guidelines

- Start each comment on a new line. The comment should begin with the comment symbol and a single space: `#`.
- If you have multiple sentences in your comment, put a `.`. If you have one sentence comment, no `.` required.

```
# This is a good comment
# This is also a good comment. But this time with two sentences.

# This is a bad comment.
# This is a bad comment. But this time with two sentences
```

- To refer to variables or functions inside comments, don't use backticks or quotes:

```
# This is a good way to refer to a variable x or the function my_fun()

# This is a bad way to refer to a variable `x` or the function 'my_fun()'
```