L01 - Basic R

Olivia Beck

May 16, 2023

Ethical Programming

"Ethical programming includes being honest about the extent to which you can be confident that your program is good, and adhering to quality control processes that ensure that if a mistake has been made – everybody makes mistakes – it is found and rectified before it causes harm."

-PERDITA STEVENS

Good programs….

- do what they are supposed to do
- are clearly written
  - Your code must be readable
  - Use comments!
- do not cause harm

R Terminology

RStudio = an add on to R which provides RStudio with better GUI

RStudio is an IDE.

- **R** is the programming language.
- **Rstudio** is a user interface to R that makes it easier to document your work, access nice features of R, and more.
  - You do not need Rstudio to use R, but it does make it much easier
- **CRAN** is a repository where the latest downloads of R (and legacy versions) are found in addition to source code for thousands of different user contributed R packages.
- **Packages** are collections of R functions, data, and compiled code in a well-defined format.
  - Packages need to be installed only once: `install.packages()`
  - Some packages are installed with R automatically: `stats`, `graphics`, `grDevices`, `utils`, `datasets`, `methods`, `base`
  - All other packages need to be installed
- **Libraries** are the directories where the packages are stored on your computer

Pacakages

- **Packages** are collections of R functions, data, and compiled code in a well-defined format, created to add specific functionality.
- R packages are stored/shared in a few different places
  - There are a set of standard (or base) packages which are considered part of the R source code and automatically available as part of your R installation
    - `base``stats`,`datasets`, `graphics`, `grDevices`, `utils`, `methods`
    - these packages are loaded automatically every time you open R or RStudio
  - **CRAN** is a repository where the latest downloads of R (and legacy versions) are found in addition to source code for thousands of different user contributed R packages
    - when we want to install packages from CRAN we use `install.packages(PackageName)`.
    - most of the packages we use in this class will come from CRAN
  - **GitHub** is a code storage/sharing platform where some people store their R packages

File Extensions

- **.R** is an R script
- **.Rmd** is a Rmarkdown document
  - The raw Rmarkdown document will have extension .Rmd
  - The rendered document will have the appropriate extension
-

File Type	Extension
PDF	.pdf
HTML	.html
Slidy	.html
Rnotebook	.nb.html
Word document	.docx
Excel document	.xlsx

Conventions (i.e. Style)

Remove slide. (check again)

- It's important to note that conventions are for the benefit of users & consumers of your code.
- **R will not "enforce" them for you** (but RStudio can help…see end of slides)
- Style will be graded on several assignments (using RStudio config)
    - I will always be clear in an assignment what stylistic objects I'm grading on. For example, in the Tidy Data Activity, I will tell you to have a descriptive file names and to use no spaces in the file, but I will not specify the exact names for each column.
    - A few things that I will always expect:
        - No spaces in file names
        - Names are always descriptive (files, variables, functions, column names in data frames)
    - What ever style guide you pick, stay consistent!
        - I personally like tidyverse styling convention:
            - file-path/file-name.extension
            - `function_name()`

Good Coding Practices

Assign vs. Equals

- Use “<-” (assign) not “=” (equals) when assigning variable/function values
    - In R, “<-” is used to assign variable/function names and “=” is used inside functions for arguments

```
# Good
x <- 1:5
quants.good <- quantile(x, probs = c(0.1, 0.9))

# Bad
x = 1:5
quants.bad = quantile(x, probs = c(0.1, 0.9))
```

See further.
Any example code to understand difference?

Naming conventions

- NEVER use spaces to name things
  - file paths/names should use "-". e.g. file-name.R
  - function names should use "_". e.g. `funciton_name()`
  - variable names should use "." e.g. `variable.name`
- Use lower case letters when possible
  - good: file-name.R, `function_name()`, `varaible.name`
  - bad: File-Name.R, `Funciton.Name()`, `VaRiAbLe.NaMe`
- SomePeopleLikeToUseCamelCase
- Anything you name should be meaningful. This includes file names, function names, variable names, etc.

Is this standard?

R is Case Sensitive.

Let's look at the `iris` data set:

```
head(iris)
```

Using Spaces

- Some coding languages respect white space, and others ignore it … R does both

(?)

When to use spaces

- Most of the time, R ignores white space. White space is often used to make code readable to the human eye.
- Place spaces around all operators (=, +, -, <-, etc.). The same rule applies when using = in function calls. Always put a space after a comma, and never before (just like in regular English).

```
# Good
average <- mean(feet / 12 + inches, na.rm = TRUE)

# Bad
average<-mean(feet/12+inches,na.rm=TRUE)
```

- Extra spacing (i.e., more than one space in a row) is okay (but not required) if it improves alignment of equal signs or assignments (<-).

```
# Good
list(
  total = a + b + c,
  mean  = (a + b + c) / n
)

longer_variable_name <- 5
```

When not to use spaces

- There are a small number of cases where R respects white space

1. :, :: and ::: can't have spaces around them.

```
# Good
x <- 1:10
base::get

# Bad
x <- 1 : 10
base :: get
```

1. When subsetting with `[]` , the first `[` cannot have a space before it.

```
#good
x[ 1 ]

#bad
x [ 1 ]
```

1. <- (assign) must NEVER have a space between the < and the -.

- x <- y (assign) V. x < - y (is x less than -y)

Lines of Code

- Strive to limit your code to 80 characters per line. This fits comfortably on a printed page with a reasonably sized font. If you find yourself running out of room, this is a good indication that you should encapsulate some of the work in a separate function.
    - Use the light gray line on the left side of the console in RStudio
- An opening curly brace should never go on its own line and should always be followed by a new line. A closing curly brace should always go on its own line, unless it's followed by else. Always indent the code inside curly braces. (don't worry about this right now, we will come back to it later)

```
# Good
if (y < 0 && debug) {
  message("Y is negative")
}

if (y == 0) {
  log(x)
} else {
  y ^ x
}
```

Indentation

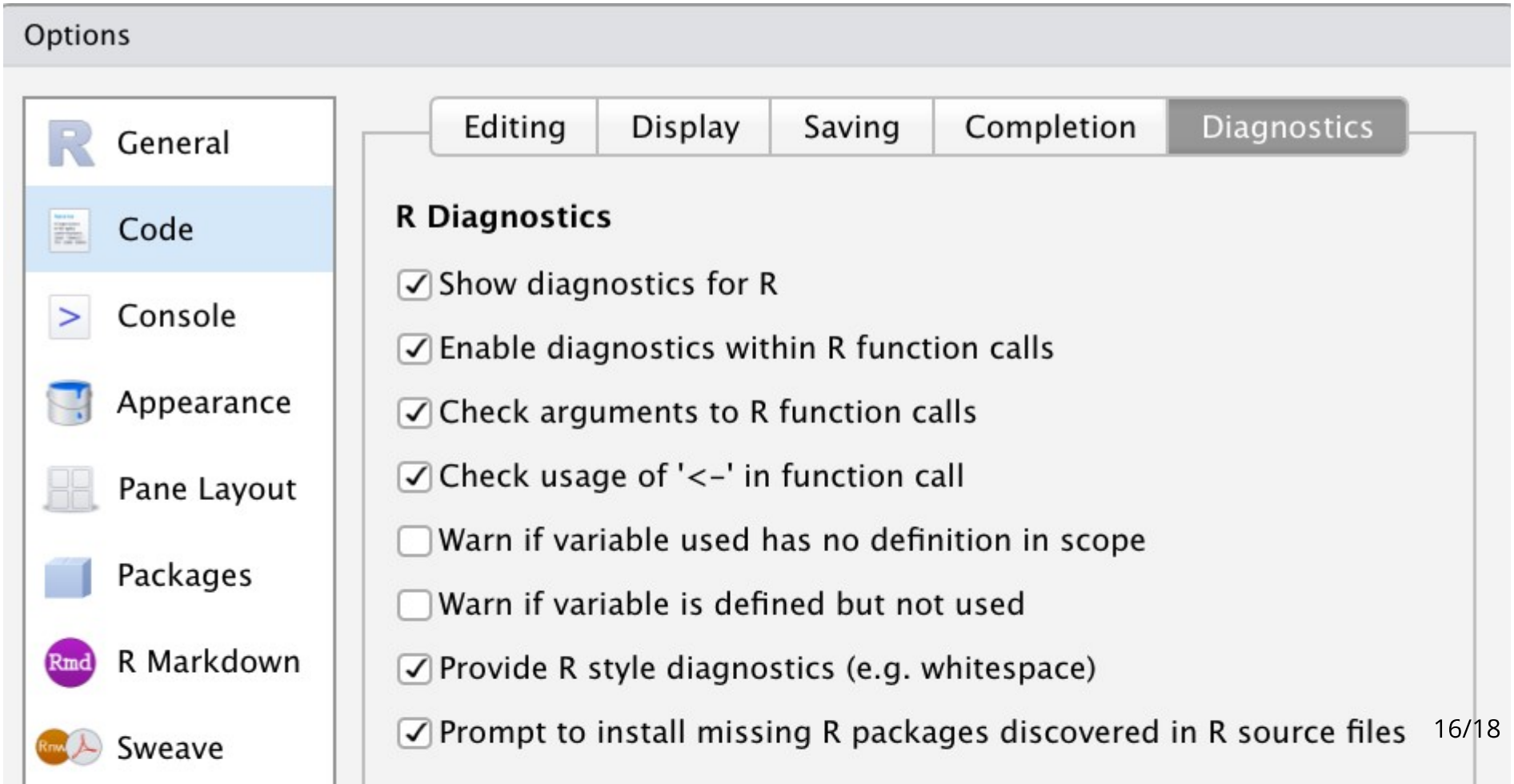
- When indenting your code, use two spaces. Never use tabs or mix tabs and spaces.
    - a tab is usually 12 spaces
    - in Rstudio you can set your tab button to be 2 spaces: Tools > Global Options > Code > Editing > General > Check “Insert spaces for tab” > Set tab width equal to 2 spaces
    - You can set up auto-indent by: Tools > Global Options > Code > Editing > General > Check “Vertically align arguments in auto-indent”
- The only exception is if a function definition runs over multiple lines. In that case, indent the second line to where the definition starts:

```
# Good
long_function_name <- function(a = "a long argument",
                               b = "another argument",
                               c = "another long argument") {
  # As usual code is indented by two spaces.
  return(a + b + c)
}
```

RStudio Help with Style

RStudio has some basic support built in (required for STAT 184), but more comprehensive support is provided by the `lintr` package (recommended).

RStudio >> Tools >> Global Options >> Code >> Diagnostics >> check nearly all boxes (both "warn if variable…" settings can be optional)



Additional Readings

- Advanced R by Hadley Wickham Ch. 22 http://adv-r.had.co.nz/Style.html
- How to Write Good Programs by Perdita Stevens Ch. 2 & 3

References

- https://hbctraining.github.io/Intro-to-R-flipped/lessons/04_introR_packages.html.
- https://www.statmethods.net/interface/packages.html
- https://hbctraining.github.io/Intro-to-R-flipped/lessons/04_introR_packages.html
- https://www.markdownguide.org/getting-started/
- https://rmarkdown.rstudio.com
- http://adv-r.had.co.nz/Style.html