**STYLE – what?**

Programming *style* is a set of *rules or guidelines* used when writing ... code.

https://en.wikipedia.org/wiki/Programming_style

---

Style concerns everything: files, functions, objects, arguments, names, spacing, indenting, assignment, quotes, comments, pipes, capitalization, punctuation, and so on...
STYLE – why?

... following a particular programming style will help programmers read and understand source code conforming to the style, and help to avoid introducing errors.

https://en.wikipedia.org/wiki/Programming_style
STYLE – where?

There is no agreed upon style in the R environment. But choose one and stick to it.

Tidyverse Styleguide
http://style.tidyverse.org/

rOpenSci Packaging Guide
https://github.com/ropensci/onboarding/blob/master/packaging_guide.md

Google’s Styleguide
https://google.github.io/styleguide/Rguide.xml
... documentation is written text ... that accompanies ... or is embedded in ... code. It either explains how [the code] operates or how to use it ...

https://en.wikipedia.org/wiki/Software_documentation

```
1 add <- function(x, y) {
2   # Add x and y together and automatically
3   # return the result
4   x + y
5 }
```

```
1 #' Add together two numbers.
2 #'
3 #' @param x A number.
4 #' @param y A number.
5 #' @return The sum of \code{x} and \code{y}.
6 #' @examples
7 #' add(1, 1)
8 #' add(10, 1)
9 add <- function(x, y) {
10   x + y
11 }
```
DOCUMENTATION – why?

Code should be easy to understand

That means that it should be written to minimise the time it would take for someone else to understand it.

The purpose of commenting is to help the reader know as much as you did when you wrote the code and everything was fresh in your memory.

That reader could very well be future you. And we all know past you won’t answer emails.
DOCUMENTATION – where?

Tidyverse Styleguide
http://style.tidyverse.org/

rOpenSci Packaging Guide
https://github.com/ropensci/onboarding/blob/master/packaging_guide.md

R packages
http://r-pkgs.had.co.nz/

The Art of Readable Code
https://www.amazon.com/Art-Readable-Code-Practical-Techniques/dp/0596802293
Git – what?

Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development...

https://en.wikipedia.org/wiki/Git
Data science is software development...
...and you need to use git every time.
Git and GitHub
http://r-pkgs.had.co.nz/git.html

Version Control with Git and SVN

A successful Git branching model
http://nvie.com/posts/a-successful-git-branching-model/
In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.

https://en.wikipedia.org/wiki/Unit_testing
TEST – why?

Testing makes sure that your code works – and that it works the way you want it to.

Testing gives you fewer bugs, a better code structure, easier restarts and more robust code.

CRAN checks also greatly improve your package.
Chapter: Testing
http://r-pkgs.had.co.nz/tests.html

testthat package
https://github.com/r-lib/testthat

devtools package
https://www.rstudio.com/products/rpackages/devtools/
DOCKER – what?

Docker is a computer program that performs operating-system-level virtualization also known as containerization.

Docker uses resource isolation to allow independent "containers" to run within a single OS, avoiding the overhead of starting and maintaining virtual machines (VMs).
DOCKER – why?

Docker can create the exact environment that you need for your analysis.

Once build you can use this image every time you need to run the given analysis.

The processes will be totally isolated and software you run or update will have no effect outside of the given container.

And you can chain them together in powerful docker-recipes using docker compose.
DOCKER – where?

Rocker
https://github.com/rocker-org/rocker

Docker tutorial
https://docs.docker.com/get-started/

Dockerhub
https://hub.docker.com/explore/