# DfE Best Practice for R:: CHEAT SHEET



### Software



R Studio Write code in the **RStudio** IDE



Use **Git** to version-control your code and analysis



Use GitHub / AZURE DevOps to collaborate with other people

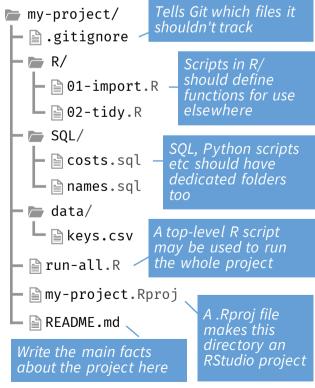
### **Projects**

### **PROJECT CREATION**

- **Create** a new project in RStudio using File > New Project > New Directory
- **Do** put projects in C:\Users\your-name\Documents
- **Don't** put projects in C:\Users\vour-name\OneDrive -Department for Education\Documents

#### PROJECT STRUCTURE

Projects are folders containing a file with the extension .Rproj. Projects should be structured something like this:



NB, usethis::use description()+ usethis::use namespace() will turn this structure into a package!

### **Packages**

Packages should be loaded in one place with successive calls to library()



Use the **tidyverse** for normal wrangling, plotting etc



Use tidymodels for modelling and machine learning



Use {shiny}, {bslib} and {bs4Dash} for app development



Use <u>r-lib</u> packages like {rlang}, {cli} and {glue} for low-level programming

GitHub stars are a good proxy for a package's quality. Not sure whether to use a package? If it has >200 stars on GitHub it's probably okay!

### Getting Help



### **CREATE A REPREX**

- A minimal, reproducible example should demonstrate the issue as simply as possible
- Copy your example code and run reprex::reprex() to embed errors/messages/outputs as comments
- Use your reprex in a question on Teams or Stackoverflow

```
print("Hello " + "world!")
#> Error in "Hello " + "world!": non-
numeric argument to binary operator
```



This reprex minimally demonstrates an error when attempting to use + for Python-style string concatenation

#### **ETIOUETTE WHEN ASKING OUESTIONS**

Ignore messages or warnings	Ensure your code only fails where you're expecting it to
Include big files	Use dput() or tibble::tribble() to include a data sample
Post screenshots of your code	Use reprex::reprex() and paste your code as text
Don't	Do

### **Databases**

- Use {DBI} and {odbc} to connect to SOL
- Use **helper functions** to create connections

```
connect to db <- function(db) {</pre>
 DBI::dbConnect(
    odbc::odbc(), Database = db,
    # Hard-code common options here
        # Connect using the helper
         con <- connect to db("DWH PL")</pre>
```

## **Learning More**

- · Common data science tasks: use R for Data Science (2e)
- Developing packages: use R Packages (2e)
- Advanced programming: use Advanced R (2e)
- Creating apps: use Mastering Shiny





#### **Functions** WRITING FUNCTIONS: WORKFLOW

```
• Write functions to reduce repetition
  or increase clarity
```

- Write many **small** functions that **call** each other
- Define functions in **dedicated scripts** with corresponding names

#### **NAMING CONVENTIONS**

```
√ Good (verb-like)

                    * Bad (noun-like)
compute totals()
                    totals getter()
fit model()
                    modeller_func()
import datasets() project data()
```

```
a <- complex operation on a
                              1. Repetitive,
b <- complex operation on b
                                  complex code;
c <- complex operation on c
                                  purpose clarified
d <- complex operation on d
                                  by comments
```

```
operate_on <- function(x) {</pre>
   complex operation on x
```

abstracted into **functions** 

2. Complex logic

```
a <-operate on(a)
b <-operate on(b)
c <-operate on(c)</pre>
d <-operate on(d)</pre>
```

3. Repetition reduced; clearer code; less need for comments

### Styling

For other styling guidance, refer to the Tidyverse style guide

#### **NAMING THINGS**

- Use lower\_snake\_case for most objects (functions, variables etc)
- **Title Snake Case** may be used for column names
- Use only **syntactic** names where possible (include only *numbers*, *letters*, *underscores* and *periods*, and don't start with a number)

### # Good (lower\_snake\_case everywhere): $\leftarrow$ function(x) x + 1 first letters <- letters[1:3]</pre> iris sample <- slice sample(iris, n = 5)</pre> # Bad (non-syntactic, doesn't use snake\_case): `add 1` $\leftarrow$ function(x) x + 1 FirstLetters <- letters[1:3]</pre> iris.sample <- slice sample(iris, n = 5)</pre>

#### **WHITESPACE**

- Add spaces after commas and around operators like |>, %>%, +, -, \*, /, = and <-
- Indentation increases should always be by *exactly* 2 spaces
- Add linebreaks when lines get longer than 80 characters.
- When there are many arguments in a call, give each argument its own line (including the first one!)

```
# Good (lots of spaces, indents always by +2):
df <- iris |>
 mutate(
    Sepal.Area = Sepal.Width * Sepal.Length,
    Petal.Area = Petal.Width * Petal.Length
# Bad (inconsistent spacing and indentation):
df<-iris |>
 mutate(Sepal.Area=Sepal.Width*Sepal.Length,
      Petal.Area=Petal.Width*Petal.Length)
```