R Programming For Natural Resource Professionals

Lecture 2: Data Structures, Data Types, Finding Help, Coding Etiquette

Paper discussions

- Group assignments

Paper discussions

- Open this link now: https://bit.ly/3GVPwuV
- Talk with your group to identify one or two thoughts, questions, and epiphanies that resonate then record them in the Google Doc.
- Read through the list as it updates.

Homework reminders

- Be sure to cite sources, including your colleagues.
- Remember to comment your code.
  - In many cases, line-by-line commenting is appropriate
- Follow instructions exactly.
- Homework will be returned via email. Grades will be on Canvas.
  - Read through the R markdown that I return. It includes notes from me (JH: …)

Topics for today

- Data types
- Data structures
- Finding help
- Coding etiquette
- Workflows

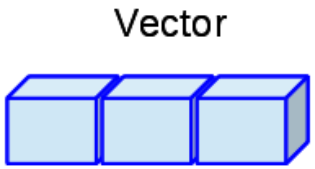
Data types

Data type	Example	Verify
Logical	TRUE, FALSE	`a <- TRUE` `class(a)`
Numeric	42.1, 4, 2215	`b <- 23.4` `class(b)`
Integer	2L, 24L, 0L	`c <- 2L` `class(c)`
Character	“a”, “good”, “perch”	`d <- “perch”` `class(d)`
Complex	1+4i	`e <- “1+4i”` `class(e)`

Topics for today

- Data types
- Data structures
- Finding help
- Coding etiquette
- Workflows

Data structures



Vector: A sequence of items of the same type.

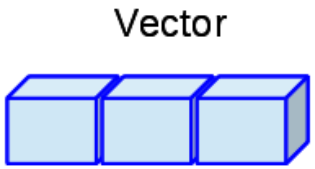
Most basic data structure in R

Items in a vector can be accessed using `[]`

Length of vector displayed using `length()`

```
> apple <- c("red", "green", "yellow")
> apple
> class(apple)
> length(apple)
> apple[2]
```

Data structures

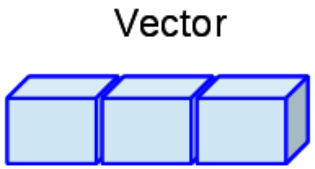


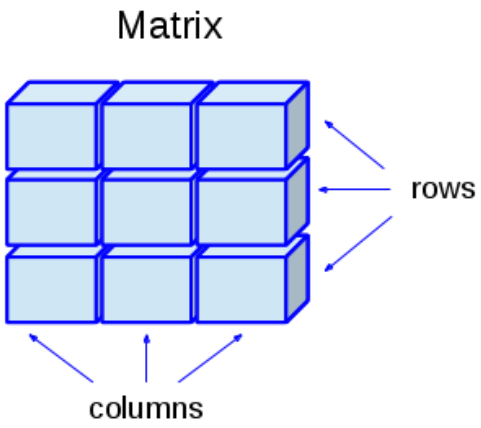
<u>Factor</u>: A common type of vector used in plotting and modeling.

Forces values of the vector into categories

```
> apple <- c("red", "green", "yellow")
> factor_apple <- as.factor(apple)
> factor_apple
> str(factor_apple)
```

Data structures

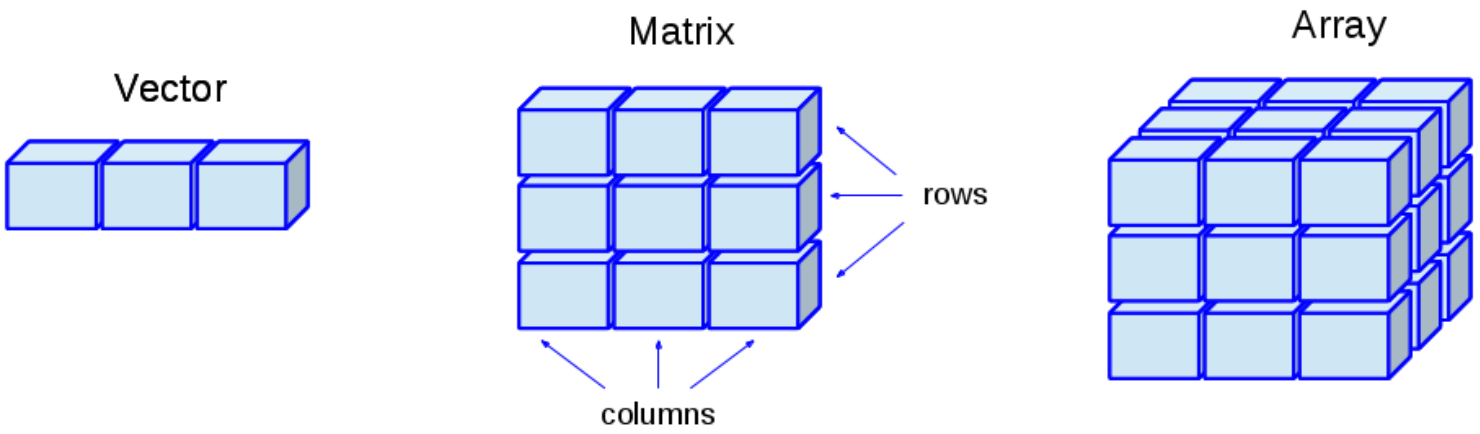




<u>Matrix</u>: A two-dimensional array

```
> matrix1 <- matrix(c("a", "a", "b", "c", "c", "a"),
nrow = 2, ncol = 3, byrow = TRUE)
> matrix1
```

Data structures



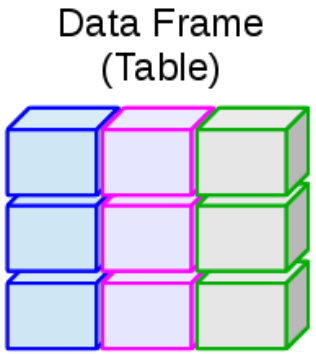
<u>Array</u>: A multidimensional matrix

Any number of dimensions.

The array function's dim attribute to creates to specify dimensionality.

```
> array1 <- array(c("green", "yellow"), dim = c(3,3,2))
> array1
```

Data structures

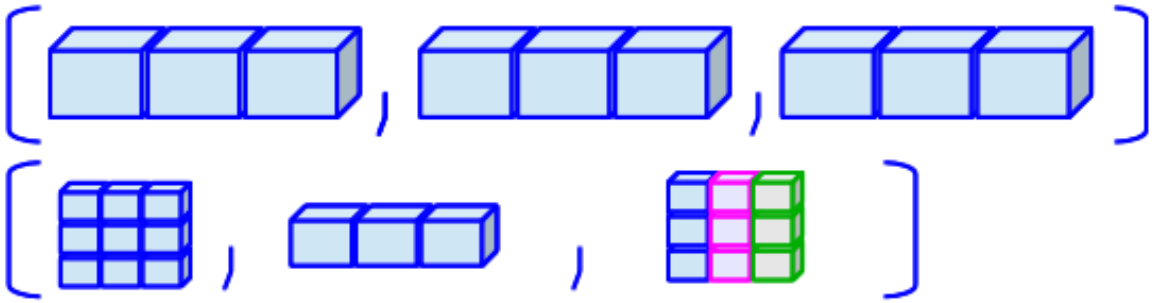


<u>Data frame</u>: A table, similar to a matrix, but each variable (column) can be a different data type

```
> df1 <- data.frame(gender = c("Male", "Female", "Female"), height = c(152, 171.5, 165), weight = c(81, 93, 83), age = c(42, 38, 26))
> df1
> str(df1)
> summary(df1)
```

Data structures

Lists



<u>Lists</u>: Can contain many different types of elements, such as vectors, arrays, data frames, or even other lists.

```
> list1 <- list(c(2,5,3), 21.3, “tree”))
> str(list1)
> list1[3]
> list1[[3]]
> list1[[1]][[2]]
```

Topics for today

- Data types
- Data structures
- Finding help
- Coding etiquette
- Workflows

Finding help

R's built-in help pages

```
> ?mean
```

Files | Plots | Packages | Help | Viewer

Refresh Help Topic

R: Arithmetic Mean | Find in Topic

mean {base} R Documentation

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)

## Default S3 method:
mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments

`x`	An R object. Currently there are methods for numeric/logical vectors and date, date-time and time interval objects. Complex vectors are allowed for `trim = 0`, only.
`trim`	the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
`na.rm`	a logical value indicating whether `NA` values should be stripped before the computation proceeds.
`...`	further arguments passed to or from other methods.

Value

If `trim` is zero (the default), the arithmetic mean of the values in `x` is computed, as a numeric or complex vector of length one. If `x` is not logical (coerced to numeric), numeric (including integer) or complex, `NA_real_` is returned, with a warning.

If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations deleted from each end before the mean is computed.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

Finding help

<u>Online resources</u>

Stack Overflow

Package vignettes on CRAN

- https://cran.r-project.org/web/packages/vegan/index.html

R-bloggers

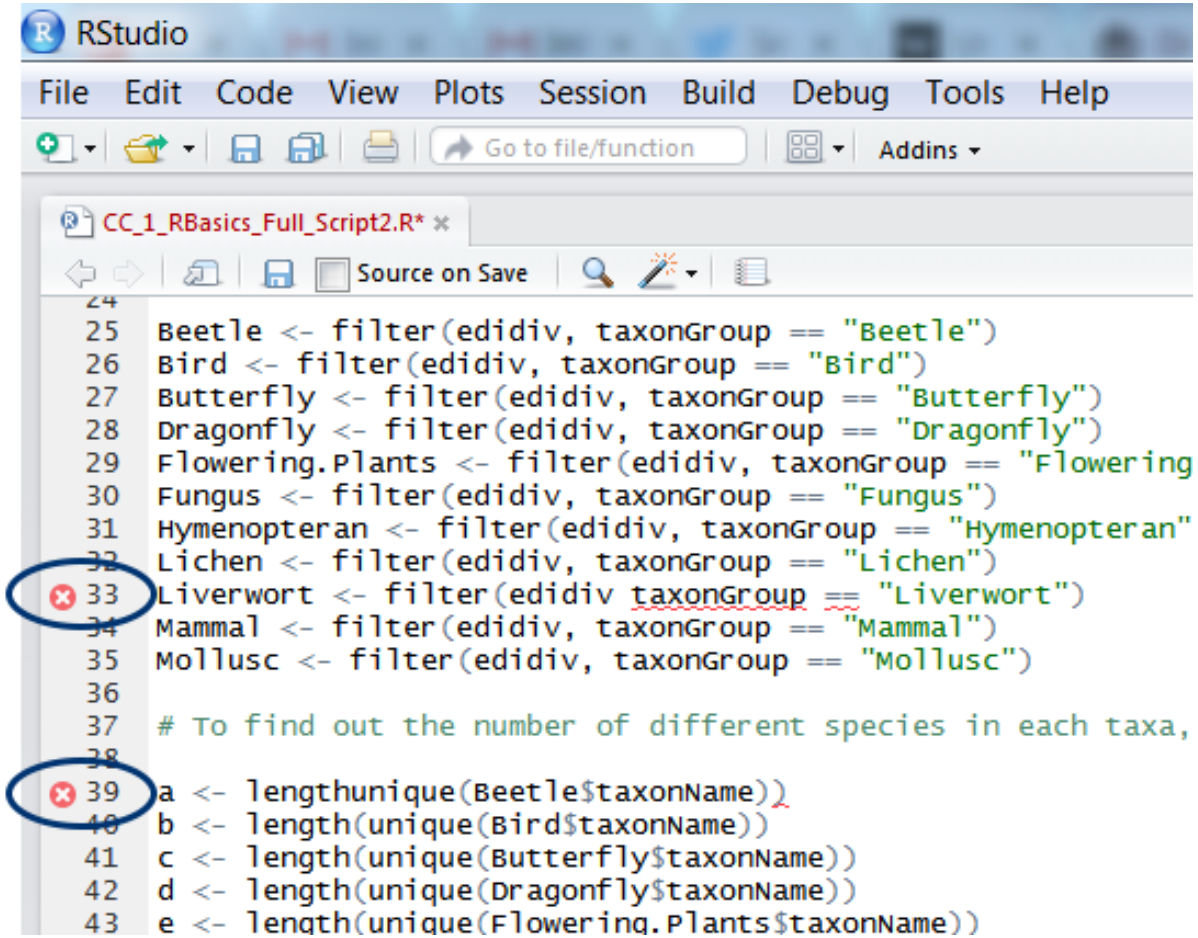
Package's GitHub sites

Finding help

Realtime Error Checking in Rstudio

Syntax errors are common and easy to make.

- Open bracket or parentheses
- Missing commas
- Extra character or other typo



Common errors

Error: Could not find function ‘functionName’

Likely solution: Package containing function not loaded. `library(packageName)`

Error: There is no package called ‘packageName’

Likely solution: Package isn’t installed

Error: object ‘objectName’ not found

Likely solution: Check your environmental panel that object is loaded. Check for typos.

Error: unexpected symbol in ‘lineOfCode’

Likely solution: A forgotten or extra comma, bracket, etc.

Topics for today

- Data types
- Data structures
- Finding help
- Coding etiquette
- Workflows

Coding Etiquette

File names should be meaningful

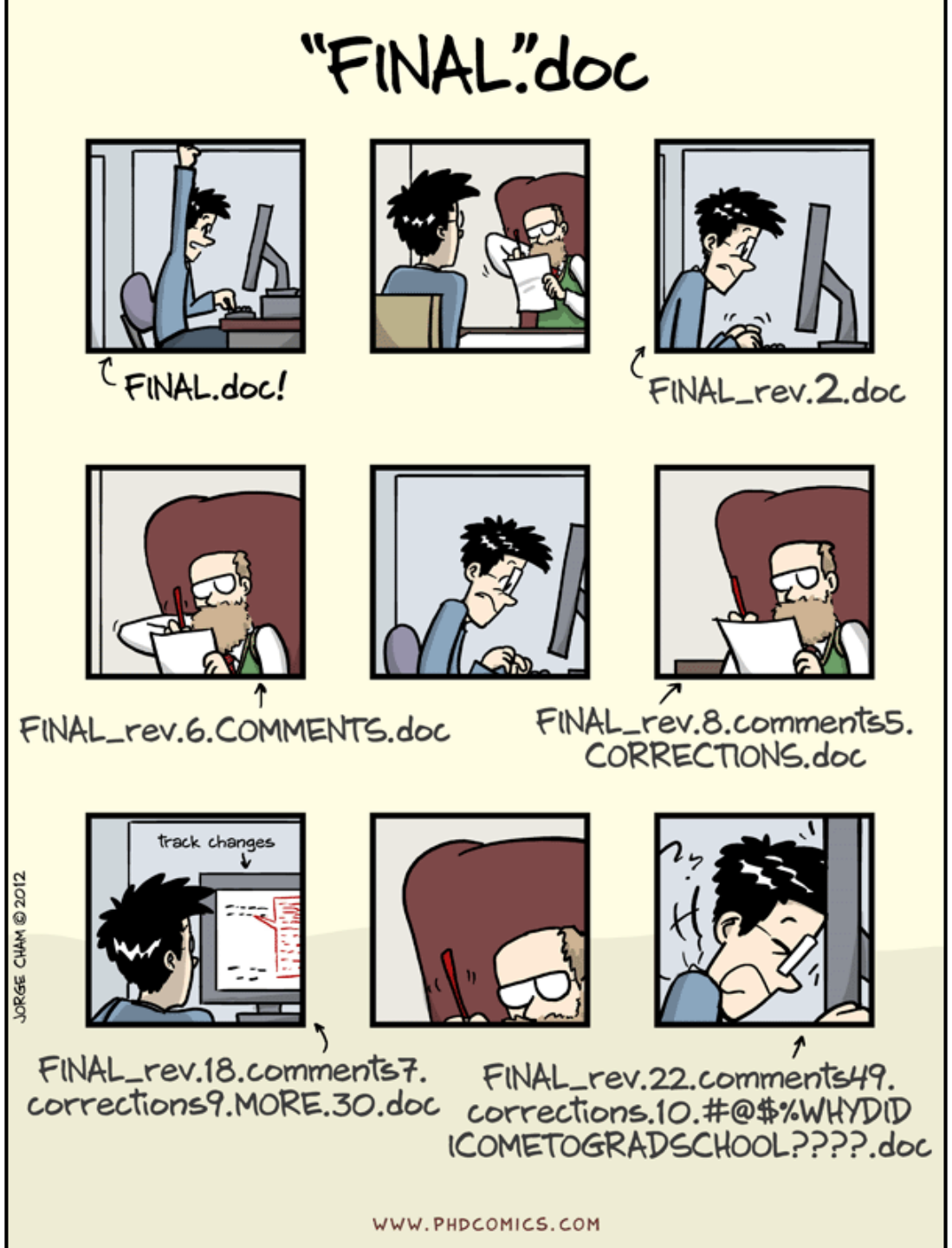
- dataWrangling.R
- repeatedMeasuresModel.R
- modelSelection.R

Coding Etiquette

If files must be run in sequence, give them numerical prefixes:

- 1_dataWrangling.R
- 2_modelSelection.R
- 3_plotting.R

Word from the wise- no matter how much you think it is appropriate, never name anything “final.”



Coding Etiquette

Function names should be **verbs**

Object names should be **nouns**

Good function names	Good object names
`parse_data`	`green_bay_env`
`generate_boxplots`	`mccurdy_woodlot_dbh`

Coding Etiquette

Do not assign names to existing functions

For example:

- `c`
- `T`
- `F`
- `data`

Coding Etiquette

Put **space** around all operators and after commas.

Just like English. Make your code readable.

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

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

Coding Etiquette

Put **space** around all operators and after commas.

Just like English. Make your code readable.

Exception to the rule!

`base::mean`

Coding Etiquette

Opening curly brackets do not go on new line but are followed by a new line.

Closing curly brackets always go on a new line.

```
if (y < 0){
    message("Y is negative")
}
```

Coding Etiquette

Assignment should always be done using “<-”, never “=“

Good: `x <- 10`

Bad: `x = 10`

Coding Etiquette

Case: lots of case options. Just be consistent.

- lower_snake
- UPPER_SNAKE
- lowerCamelCase
- UpperCamelCase
- kebab-case

Coding Etiquette

Use commenting frequently

Titling scripts:

```
##############################
#### Length frequency plots ####
##############################
```

In R Studio: `#####` generates a collapsible code block

Coding Etiquette Exercise

```
x=330/12

c <- 1972

if (y < 0){message("Y is negative")}
```

Topics for today

- Data types
- Data structures
- Finding help
- Coding etiquette
- Workflows

Workflows

Goal: Portable code

- R scripts and data files recreate the environment

R Studio facilitates a project-oriented workflow to help with this

Workflows

Projects include:

- Input data
- R scripts
- A default home directory
  - Enables writing of relative paths (e.g., data/species_counts.csv)

One folder = one project

Workflows

Projects directory structures can vary, but a good starting point is:

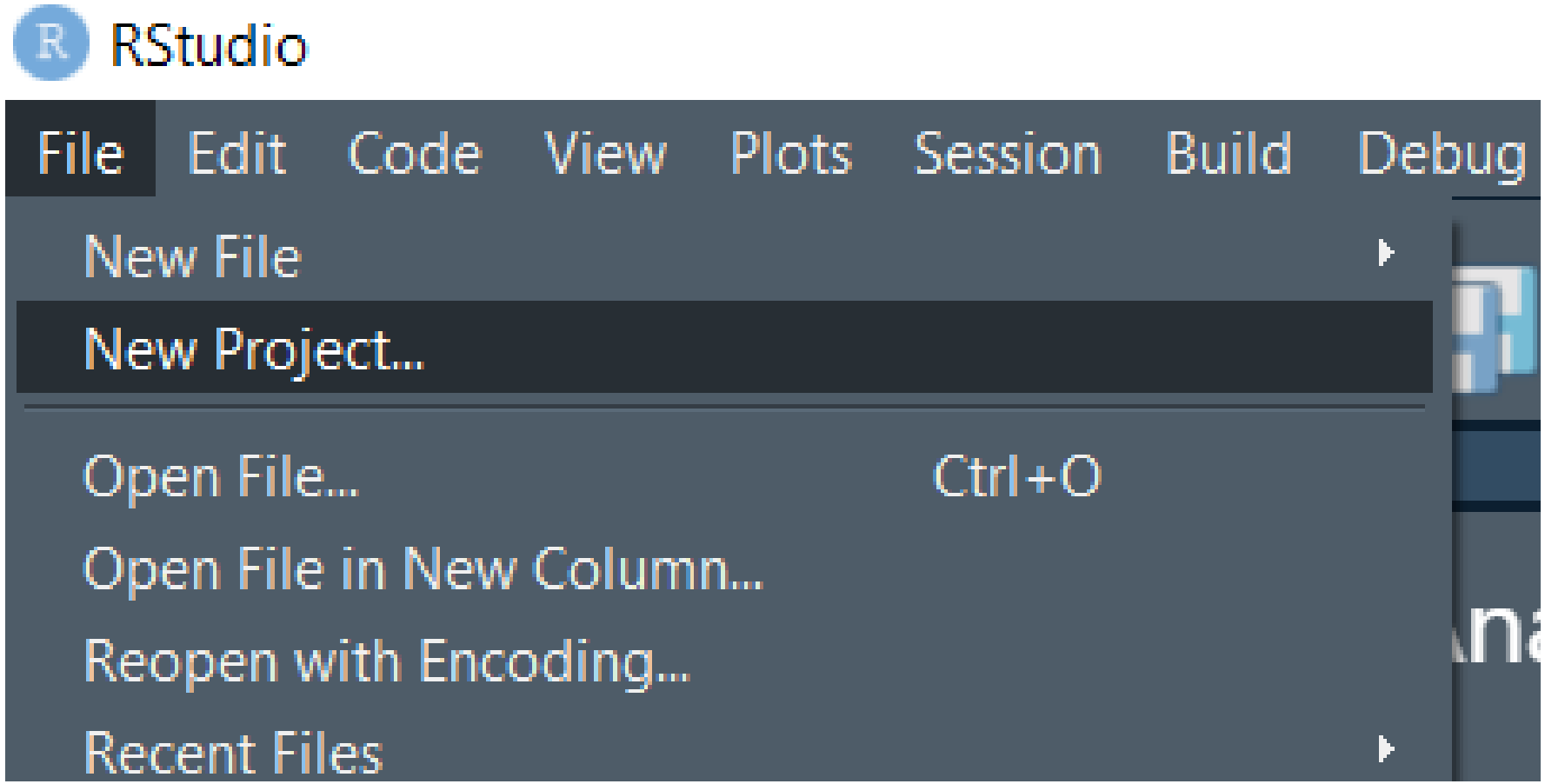
1. data folder: all input data and metadata
2. doc folder: manuscript and other documents
3. figs folder: all output figures
4. output folder: intermediate outputs (e.g., wrangled data)
5. R folder: R scripts
6. reports folder: R Markdown files that generate reports

Workflows

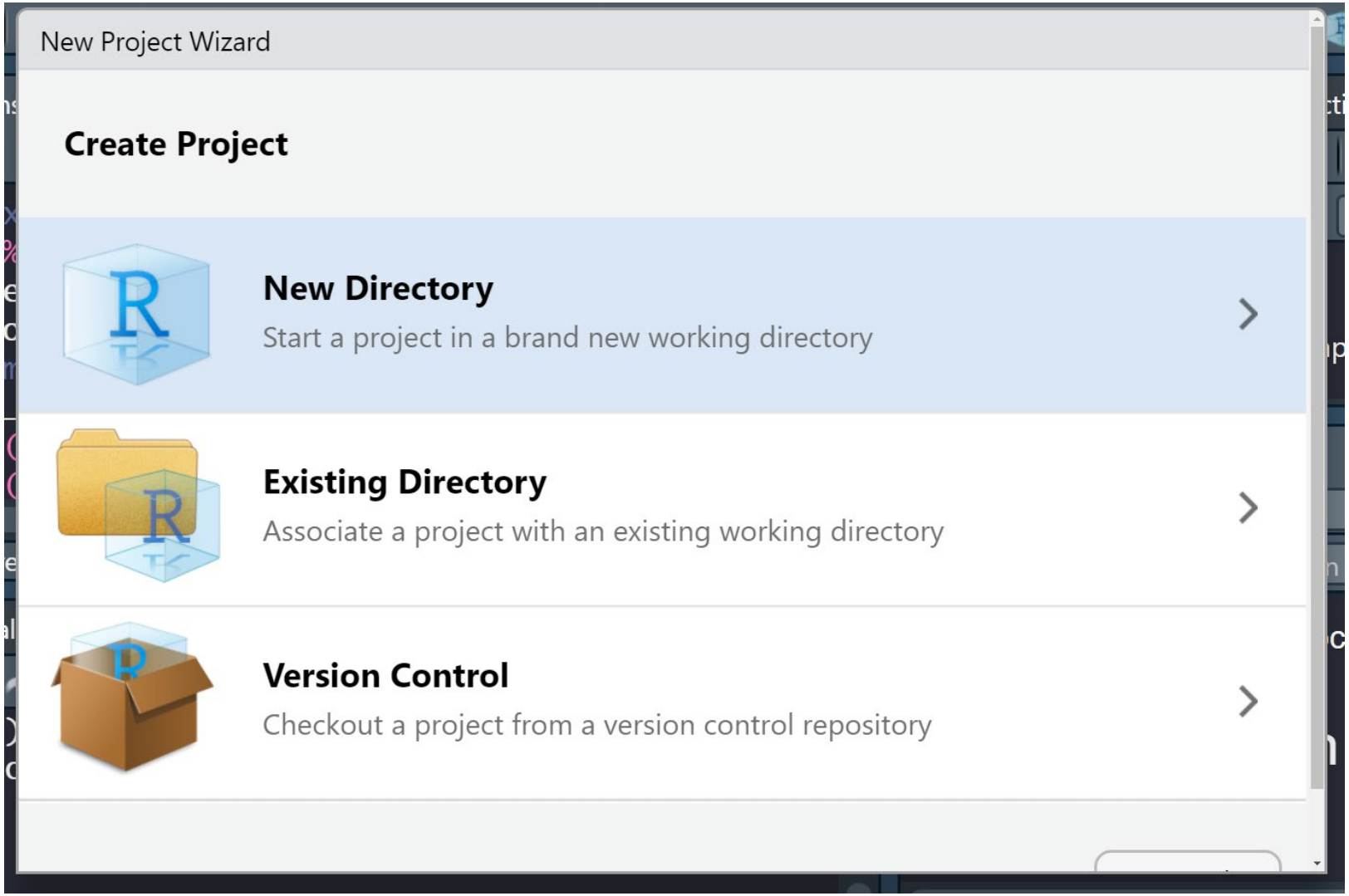
Projects should contain scripts organized for a logical workflow:

1. Load and merge data (Always work from your raw data!)
2. Data wrangling
3. Data analyses
4. Generate outputs such as tables and figures

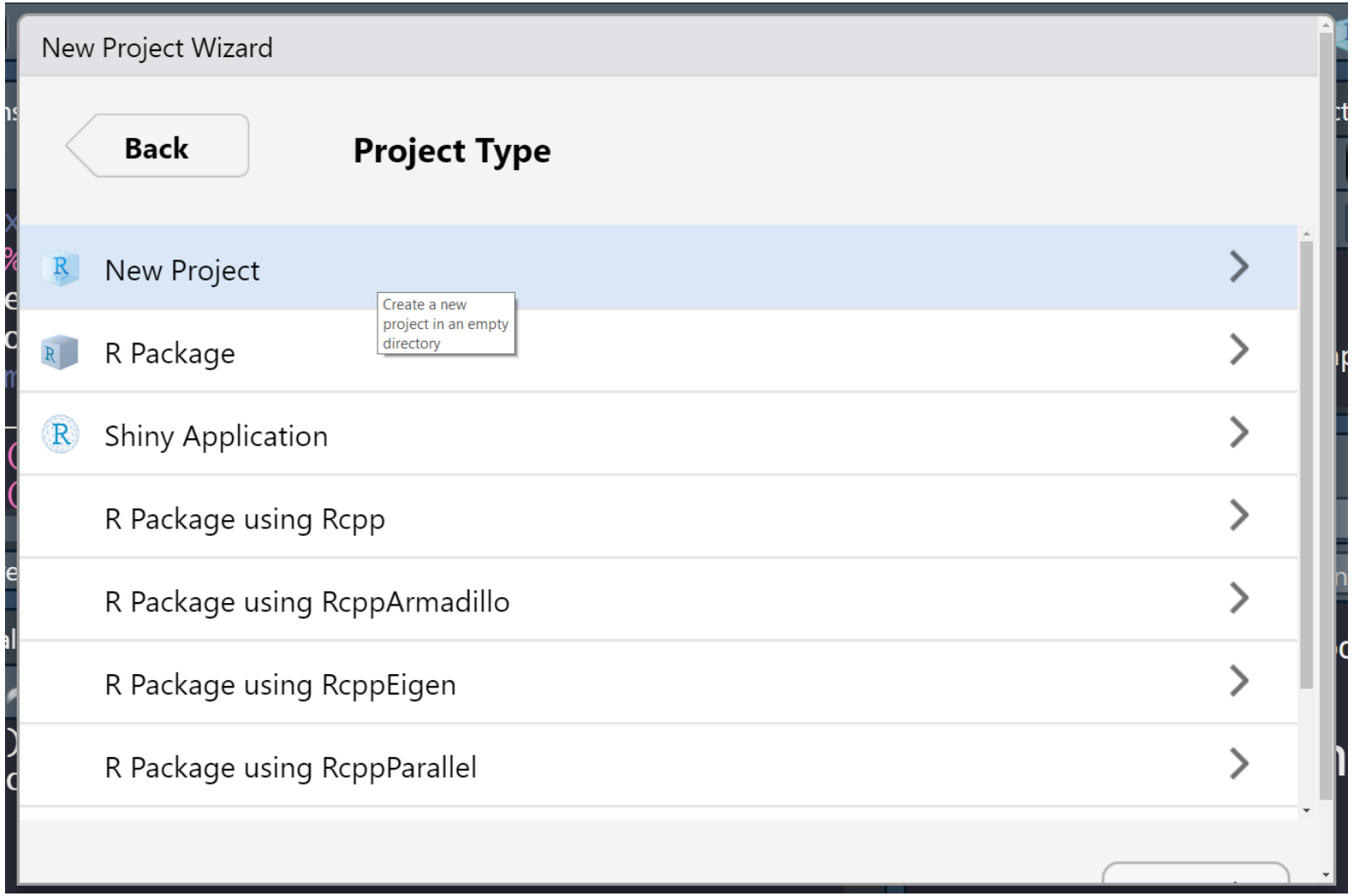
Create project for RforNatRes



Create project for RforNatRes



Create project for RforNatRes



Create project for RforNatRes