**R Legacy Training**

University of New Mexico

Department of Biology

2020

Joseph R. Stinziano, Ph.D

josephstinziano@gmail.com

github.com/jstinzi

@JosephStinziano

josephstinziano.com

**Contents**

[Core Concepts 3](#_Toc31371418)

[Homework ideas 3](#_Toc31371419)

[Principles of clean and transparent coding 4](#_Toc31371420)

[R projects, version control through git and GitHub 4](#_Toc31371421)

[Using Nested Working Directories 6](#_Toc31371422)

[Using R Markdown for Troubleshooting 6](#_Toc31371423)

[Using Conditionals 6](#_Toc31371424)

[Using for Loops 6](#_Toc31371425)

[Building Functions That Work 6](#_Toc31371426)

# Core Concepts

1. Principles of clean & transparent coding (Easy)
2. R projects, version control through git and GitHub (Easy)
3. Using nested working directories (Easy)
4. R markdown for troubleshooting (Easy – Medium)
5. Using conditionals (Easy – Medium)
6. Using for loops (Easy – Medium)
7. Building functions that work (Medium)
8. Building function output structures (Medium)
9. Meta-programming – generalizing your code (Hard)
10. Dealing with unknown file types (Hard)
11. Building R packages (Medium – Hard)
12. Unit testing (Medium)
13. Working with large datasets (Medium – Hard)

# Homework ideas

1. Bring a script you’ve written for your data analysis, then apply principles of clean and transparent coding. Provide the script and data to everyone else to try out.
2. Build a function from scratch that automates part of your data analysis. Provide function and sample data to everyone else to try.
3. Build a bifurcated function using conditionals.
4. I will provide a broken function and some data to analyze – you must fix it.
5. Build a nested loop function.
6. Build a function that takes other functions as inputs.
7. Build a function with generalized arguments.
8. Read in a file type of unknown origin, knowing only the meta-data on its structure.
9. Build an R package with functions that streamline the analysis of your own data.
10. Come up with a unit testing scheme to ensure that your functions are operating properly whenever you make a change.

# Resources

The following resources are very helpful in developing advanced expertise in R. They are listed in an order that builds on the previous book. While building R packages could happen *before* reading Advanced R and R Inferno, your R package will be far more elegant after reading these.

R for Data Science by Garrett Grolemund and Hadley Wickham

<https://r4ds.had.co.nz/>

R Inferno by Patrick Burns

<https://www.burns-stat.com/pages/Tutor/R_inferno.pdf>

Advanced R by Hadley Wickham

<https://adv-r.hadley.nz/>

R Packages by Hadley Wickham

<http://r-pkgs.had.co.nz/>

# Principles of clean and transparent coding

Writing clean and transparent code is fundamental for fundamental reasons:

1. Easier to remember what the code does when you haven’t looked at it in months
2. Easier for others to understand what the code does and how it works
3. Easier to troubleshoot
4. Takes less time to polish the code for publication
5. Higher chance that other people will actually use your code

To write clean and transparent code, there are several principles to consider:

1. Consistency: standardize your variable and function names. Use the same names across R scripts, and if certain variables have different units, then add an additional component to the variable name to signify the change in units (e.g. if gsw means stomatal conductance to water in mol m-2 s-1, when expressing it in mmol, you may want to use gsw\_mmol)
2. Modularity: whenever possible, reduce your code to the minimum functional unit. This may cause the code to run slower (e.g. running many small for loops rather than one big one), but when it comes to troubleshooting and error catching, it will be easier to locate the source of the error and interpret what went wrong.
3. Exhaustive annotations: use frequent comments to describe what the code is doing. This will help you remember the what and why of a piece of code you’ve written. It will also help you catch previous errors and misconceptions as you advance in your programming

Clean and transparent code starts with data collection. When labelling treatments and samples, develop or adopt a naming convention and stick to it for the duration of the project (or, ideally, for the rest of your life unless you come across a better version).

# R projects, version control through git and GitHub

R projects will save you a lot of time – they are a way to keep everything organized and simplify the use of working directories. When you create an R project, every time you open it up, it will bring you to the last file you were working on, while setting your working directory to the file folder for the project. Given that research often involves multiple simultaneous projects, this makes it straight forward to keep all of your code and data straight.

git, GitHub, GitHub Desktop

# Using Nested Working Directories

It is easy to put all of your data files into one folder and let them be. But this is lazy, and potentially becomes extremely complicated

# Using R Markdown for Troubleshooting

# Using Conditionals

# Using for Loops

# Building Functions That Work