

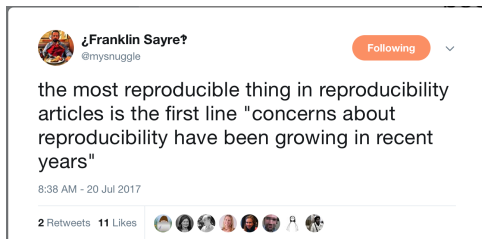
Developing a Workflow to Maximize Reproducibility and Research Impact: Managing Data, Computer Code, and Projects for Success

Althea A. ArchMiller & John R. Fieberg

7/12/2017

Developing a Workflow to Maximize Reproducibility and Research Impact: Managing Data, Computer Code, and Projects for Success

Althea A. ArchMiller & John R. Fieberg

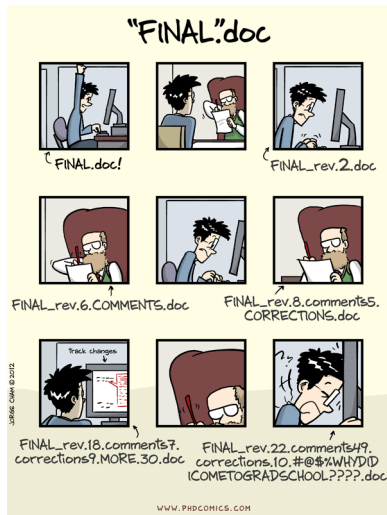


- ▶ Program R <www.r-project.org> (required)
- ▶ RStudio <www.rstudio.com/products/RStudio> (required)
- ▶ Git <git-scm.com> (optional)
- ▶ GitHub account <github.com> (optional)

Why worry about reproducibility?

Working towards future reproducibility makes my code easier for my collaborators (and me) to read, run, and debug today, and that's why I think reproducibility is a **win-win for all researchers.**"

-Althea



Why worry about reproducibility?



Vince Buffalo

@vsbuffalo



 Follow

Managing your projects in a reproducible fashion doesn't just make your science reproducible, it makes your life easier.

- ▶ make your life easier! Now, and in the future
- ▶ collaborations
- ▶ broader research impact
- ▶ increased citations
- ▶ transparency
- ▶ grant and journal requirements

“[Reproducibility] provides security, saves time, and forces me to be more thoughtful about my workflow.” - Ethan Young

Is my research reproducible?

What formats are your research documents stored in?

Is my research reproducible?

What formats are your research documents stored in?

- ▶ .csv
- ▶ .txt
- ▶ .pdf
- ▶ .html
- ▶ .R/.Rdata
 - ▶ YES - these are considered "reproducible"

Is my research reproducible?

What formats are your research documents stored in?

- ▶ .csv
- ▶ .txt
- ▶ .pdf
- ▶ .html
- ▶ .R/.Rdata
 - ▶ YES - these are considered "reproducible"
- ▶ .doc/.docx
- ▶ .sas
- ▶ .xls/.xlsx
- ▶ any other proprietary file format
 - ▶ NO - these are not "reproducible"

Is my research reproducible?

- ▶ Is your code linear?
 - ▶ Clear environment often and at beginning of script
 - ▶ Each program should focus on one main task or analysis
 - ▶ Don't rely on manual commenting/uncommenting

Is my research reproducible?

- ▶ Is your code linear?
 - ▶ Clear environment often and at beginning of script
 - ▶ Each program should focus on one main task or analysis
 - ▶ Don't rely on manual commenting/uncommenting

```
# Here, I want to know what variables are significant?  
# I will try each variable in turn  
lm.out <- lm(weight ~ height, data = trial.data)  
remove(lm.out) # clear previous lm.out for each  
               # new lm() definition above  
  
# Is the relationship significant?  
# (If not, clear and try a new variable)  
summary(lm.out)
```

Is my research reproducible?

- ▶ Are your files easily shared with others?
 - ▶ Organized directory structure
 - ▶ Files relatively linked
 - ▶ Well-documented & commented
 - ▶ Consistency in coding practices

“The point of having style guidelines is to have a common vocabulary of coding so people can concentrate on *what* you are saying, rather than on *how* you are saying it.” - Google’s R Style Guide

Is my research reproducible?

Do you treat your data as read-only?

- ▶ Don't use Excel, etc., to manipulate raw data
- ▶ Use an R script for data processing
 - ▶ Process data in one script, then save for loading into subsequent scripts
- ▶ When archiving, provide raw data and processing code not just final tables

Workshop Outline

The goal for this workshop is to help you develop the tools to develop a workflow to maximize reproducibility, collaborations, and research impact.

1. RStudio Projects for organizing data, code, and output

Workshop Outline

The goal for this workshop is to help you develop the tools to develop a workflow to maximize reproducibility, collaborations, and research impact.

1. RStudio Projects for organizing data, code, and output
2. R-Markdown and R-Oxygen with knitr for documenting your code and creating reproducible reports

Workshop Outline

The goal for this workshop is to help you develop the tools to develop a workflow to maximize reproducibility, collaborations, and research impact.

1. RStudio Projects for organizing data, code, and output
2. R-Markdown and R-Oxygen with knitr for documenting your code and creating reproducible reports
3. GitHub for version-control, collaborating and archiving

1. RStudio Projects for organizing data, code, and output

1: Introduction to RStudio

RStudio is a GUI for R that eases

- ▶ programming
- ▶ debugging
- ▶ data organization
- ▶ data processing
- ▶ project management
- ▶ program version-controlling and archiving
- ▶ plotting
- ▶ R library installation and updating

1: Introduction to RStudio

RStudio is a GUI for R that eases

- ▶ programming
- ▶ debugging
- ▶ data organization
- ▶ data processing
- ▶ project management
- ▶ program version-controlling and archiving
- ▶ plotting
- ▶ R library installation and updating
- ▶ Reproducibility!

1. Introduction to RStudio

Think about a typical research project, maybe a dissertation chapter or an experiment that you've managed from data collection through publication. What are typical **folders** that you've used?

1. Introduction to RStudio

Think about a typical research project, maybe a dissertation chapter or an experiment that you've managed from data collection through publication. What are typical **folders** that you've used?

- ▶ Raw data
- ▶ Processed data
- ▶ Analysis scripts
- ▶ Paper/Manuscript-related documents
- ▶ Sharing documents (“transmittals”)
- ▶ Metadata
- ▶ Maps or other deliverables

1. Introduction to RStudio

Think about a typical research project, maybe a dissertation chapter or an experiment that you've managed from data collection through publication. What are typical **folders** that you've used?

- ▶ Raw data
- ▶ Processed data
- ▶ Analysis scripts
- ▶ Paper/Manuscript-related documents
- ▶ Sharing documents ("transmittals")
- ▶ Metadata
- ▶ Maps or other deliverables

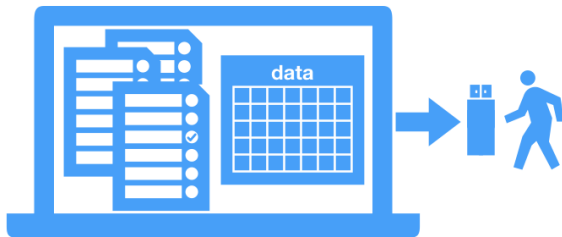
RStudio Projects provide an opportunity for you to organize and manage all of these types of folders in **one place** in a way that **relatively links** everything together and **eases sharing**.

1: Introduction to RStudio



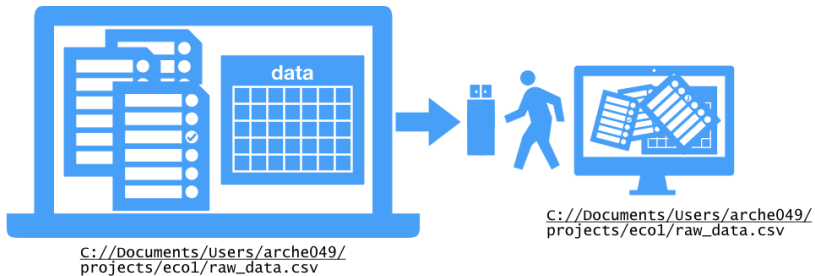
C://Documents/Users/arche049/
projects/ecol/raw_data.csv

1: Introduction to RStudio

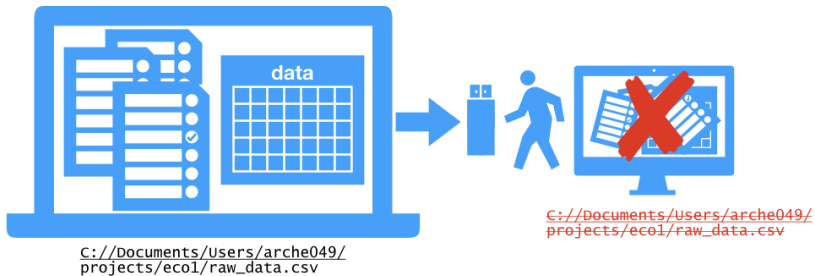


C://Documents/Users/arche049/
projects/ecol/raw_data.csv

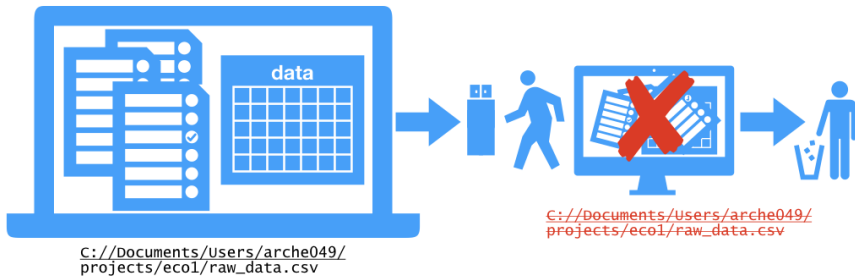
1: Introduction to RStudio



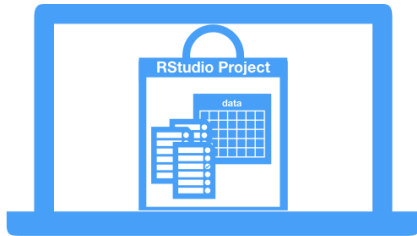
1: Introduction to RStudio



1: Introduction to RStudio

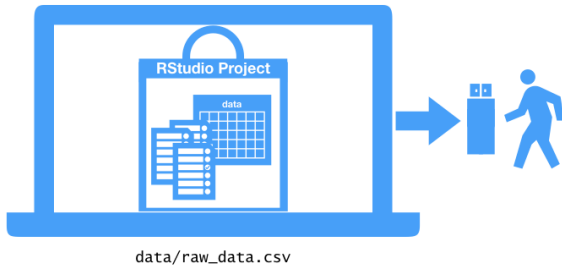


1: Introduction to RStudio

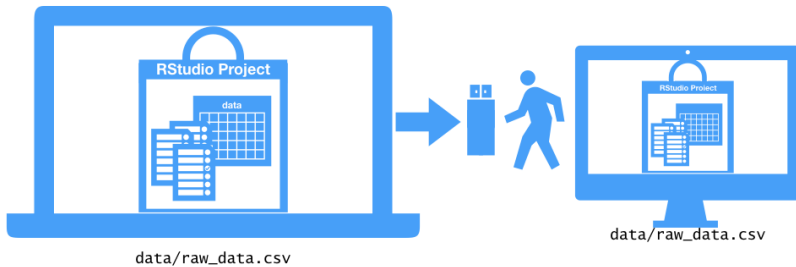


`data/raw_data.csv`

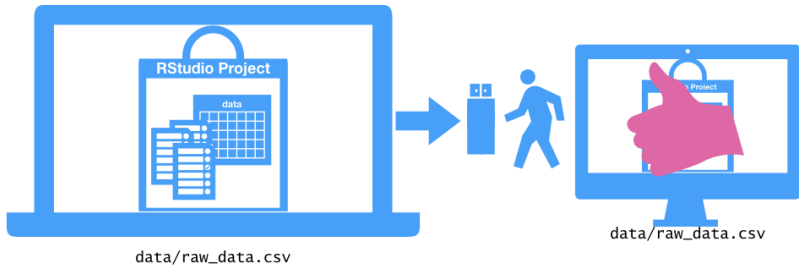
1: Introduction to RStudio



1: Introduction to RStudio



1: Introduction to RStudio



1: Introduction to RStudio

The image shows the RStudio desktop environment. The top toolbar contains icons for file operations and running code. The main window is divided into four panes. The top-left pane is the Code Editor, showing R code for loading data and calculating posterior abundance. The top-right pane is the Environment and History pane, showing the current workspace with objects like 'posteriors.sptFChI.sh' and 'sightdata'. The bottom-left pane is the R Console, showing the execution of the code from the editor. The bottom-right pane is the Files and Plots pane, showing the file explorer with the project directory structure.

1. Code Editor

```
46 load("data/output_data/posteriors_sptFChIsh_1stHalf.Rdata")
47 load("data/output_data/posteriors_sptFChIsh_2ndHalf.Rdata")
48 posteriors.sptFChI.sh <- NULL
49 for(ii in 1:75){
50   if(ii <= 40){
51     temp <- 
52   }else{
53     temp <- 
54   }
55   posteriors.sptFChI.sh <- rbind(posteriors.sptFChI.sh, temp)
56 }
57 
58 #' Spatial data for getting mixed forest for unsampled plots
59 load("data/spatial_data/plotdata.R")
60
```

2. Workspace & Git

Environment History Git

Global Environment

Data

- operdat... 29120 obs. of 12 va...
- plotdat... 482 obs. of 10 vari...
- poster... 75000 obs. of 128 ..

poster... Large list (35 elemen...
year num [1:12] 2005 2006 ...

3. R Console

```
> #' Posteriors from the models
> load("data/output_data/posteriors_sptFChIsh_1stHalf.Rdata")
> load("data/output_data/posteriors_sptFChIsh_2ndHalf.Rdata")
> posteriors.sptFChI.sh <- NULL
> for(ii in 1:75){
+   if(ii <= 40){
+     temp <- posteriors.sptFChIsh_1stHalf.Rdata
+   }else{
+     temp <- posteriors.sptFChIsh_2ndHalf.Rdata
+   }
+   posteriors.sptFChI.sh <- rbind(posteriors.sptFChI.sh, temp)
+ }
> View(sightdata)
>
```

4. Files & Plots

Files Plots Packages Help Viewer

New Folder Delete Rename More

sightability_with_spatial_augmentation

Name	Size
nitinnr	39 B
...	2.5 K
...	0 B
...	142 B
sightability_with_spatial_augme...	205 B
support docs	

1: Introduction to RStudio

The screenshot displays the RStudio environment with the following components:

- Source Editor:** Contains R code for loading data and processing posteriors. The code includes comments in Italian and a loop to process data for 125 plots.
- Console:** Shows the execution of the code, with output indicating the loading of posteriors and the creation of the `sightdata` object.
- Environment:** Lists the objects in the global environment, including `operdat...`, `plotdat...`, `posteri...`, `sightda...`, and `temp`.
- Files:** Shows the file structure of the project, including `.gitignore`, `.RData`, `.Rhistory`, `data`, `documents`, `output`, `programs`, `README.md`, `sightability_with_spatial_augme...`, and `support docs`.

```
46 load("data/output_data/posteriors_sptFChIsh_1stHalf.Rdata")
47 load("data/output_data/posteriors_sptFChIsh_2ndHalf.Rdata")
48 posteriors.sptFChI.sh <- NULL
49 for(ii in 1:75){
50   if(ii <= 40){
51     temp <- posteriors.sptFChI.sh.1of2[[ii]]
52   }else{
53     temp <- posteriors.sptFChI.sh.2of2[[ii-40]]
54   }
55   posteriors.sptFChI.sh <- rbind(posteriors.sptFChI.sh, temp)
56 }
57
58 #' Spatial data for getting mixed forest for unsampled plots
59 load("data/spatial_data/plotdata.R")
60
```

Console output:

```
> #' Posteriors from the models
> load("data/output_data/posteriors_sptFChIsh_1stHalf.Rdata")
> load("data/output_data/posteriors_sptFChIsh_2ndHalf.Rdata")
> posteriors.sptFChI.sh <- NULL
> for(ii in 1:75){
+   if(ii <= 40){
+     temp <- posteriors.sptFChI.sh.1of2[[ii]]
+   }else{
+     temp <- posteriors.sptFChI.sh.2of2[[ii-40]]
+   }
+   posteriors.sptFChI.sh <- rbind(posteriors.sptFChI.sh, temp)
+ }
> View(sightdata)
>
```

Environment Data:

Object	Class	Size
operdat...	29120 obs. of 12 va...	
plotdat...	482 obs. of 10 vari...	
posteri...	75000 obs. of 128 v...	
sightda...	124 obs. of 26 vari...	
temp	1000 obs. of 128 va...	

Values:

Variable	Value
ii	75L
landsat...	num [1:12] 2005 2005 ...
posteri...	Large list (40 elemen...
posteri...	Large list (35 elemen...
year	num [1:12] 2005 2006 ...

1: Introduction to RStudio

~/Documents/postdoc/R_projects/sightability_with_spatial_augmentation - master - RStudio

Go to file/function Addins

sightability_with_spatial_augmentation

Environment History Git

Diff Commit Staged Status Path

programs/two-step/e_calculate_abundance_sptFChI

year	juldate	plot	offplot	waypoint	observed	pilot	leftseat	rightseat	snowdepth	wind	wdir	tempf	bar	btrend	
1	2005	5	58	2	620	1	2	2	3	3	0	0	-10	3049	3
2	2005	5	23	2	80	1	1	1	1	3	0	0	10	3035	3
3	2005	11	105	2	147	0	1	1	4	3	0	0	14	2996	2
4	2005	5	60	2	634	0	2	2	3	3	0	0	-5	3033	2
5	2005	11	100	2	148	0	1	1	4	3	0	0	18	3001	2
6	2005	27	100	2	200	0	1	1	3	3	5	200	21	3037	3
7	2005	11	128	2	138	1	1	1	3	3	0	0	-14	3001	2
8	2005	5	72	2	624	1	2	2	3	3	0	0	-10	3035	2
9	2005	11	72	2	153	1	1	1	4	3	0	0	14	2994	2
10	2005	20	75	2	158	1	1	1	3	3	4	30	9	3010	3
11	2005	5	60	2	636	0	2	2	3	3	0	0	-5	3033	2

Showing 1 to 11 of 124 entries

Console ~/Documents/postdoc/R_projects/sightability_with_spatial_augmentation/

```
> #' ## Plots
> #'
> #' Abundance
> #' popnEsts
> ggplot(data = popn.data[popn.data$Method!="b: FE model"&
+   popn.data$Method!="d:spt FChI (mixed)"&
+   popn.data$Method!="e:Spt RChI (mixed)"&
+   popn.data$Method!="e:spt RChI (shrub)"&
+   popn.data$Method!="f:spt RC (shrub)",],
+   aes(x = Year, y = tauhat, colour=Method, shape=Method))+
+   geom_pointrange(aes(ymin=tau.LL90, ymax=tau.UL90), position=position_dodge(0.3))+
+   ylab("Abundance and 90% CI")
+   scale_x_continuous(breaks=seq(2004, 2016, 1))+
+   theme_minimal()
> |
```

Files Plots Packages Help Viewer

Zoom Export

Abundance and 90% CI

Method

- a: mHT
- c: TS model

Year

Activity 1: Data management and updating

Here, we will read in and process three weeks of experimental data and do some preliminary analysis. Then, we will get a final (4th) week of data, which we will merge with the original data.

The goals are to:

1. Be introduced to RStudio
2. Create a framework for keeping data organized and up-to-date
3. Automatically update our analyses based on the master dataset

Context: Abundance data from ~75 invertebrate species sampled on various beaches along the Dutch coast.

Zuur, A.F., E.N. Ieno, and G.M. Smith (2007) Analysing Ecological Data. Springer, New York.

Activity 1: Data management and updating

Introduction to RStudio

- ▶ Open RStudio

Activity 1: Data management and updating

Introduction to RStudio

- ▶ Open RStudio
- ▶ Set up global options

Remember to not save .Rdata or history! This is important for ensuring that your code is linear and reproducible (i.e., at the beginning of each script, you load all the data you will need for that entire script - no more, no less!)

Activity 1: Data management and updating

Introduction to RStudio

- ▶ Open RStudio
- ▶ Set up global options

Remember to not save .Rdata or history! This is important for ensuring that your code is linear and reproducible (i.e., at the beginning of each script, you load all the data you will need for that entire script - no more, no less!)

- ▶ Create a project

Now, your R code pathways will be relatively linked to where that .Rproj file is located. This eases programming and collaboration!

Activity 1: Data management and updating

Introduction to RStudio

- ▶ Open RStudio
- ▶ Set up global options

Remember to not save .Rdata or history! This is important for ensuring that your code is linear and reproducible (i.e., at the beginning of each script, you load all the data you will need for that entire script - no more, no less!)

- ▶ Create a project

Now, your R code pathways will be relatively linked to where that .Rproj file is located. This eases programming and collaboration!

Now we can do some analysis of invertebrates!

Activity 1: Data management and updating

1. In the File window of RStudio, copy the **student_folders/student_template** folder. Rename the folder after yourself (or an alias).
2. Open a new R Script file and save it to that new folder as **student_folders/
yourname/activity1a_data_processing.R**

Activity 1: Data management and updating

1. In the File window of RStudio, copy the **student_folders/student_template** folder. Rename the folder after yourself (or an alias).
2. Open a new R Script file and save it to that new folder as **student_folders/yourname/activity1a_data_processing.R**

Activity Overview:

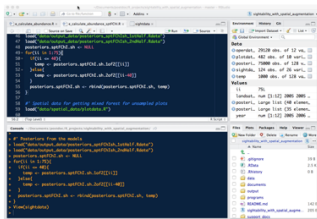
1. We will read in first three weeks of data and combine them, process the data a little bit, and save the merged/processed data for analysis.
2. We will save another new R Script file as **“activity1b_data_analysis.R”** and code/run some preliminary regression analysis.
3. We will pretend to have just gotten the final week's data in and update everything in a “reproducible” way.

2. R-Markdown and R-Oxygen with knitr for documenting your code and creating reproducible reports

Overview of knitr

R Code

written with
R-Markdown or
R-Oxygen



The screenshot shows an RStudio interface. The top pane contains R code for a function `posterior.apf202.sh` that calculates posterior probabilities for a set of models. The bottom pane shows the console output, which includes the function definition and the results of the function call, such as the log-likelihood values for each model.



Report

.html
.doc
.pdf



Why **knitr** for manuscripts?



Julia Silge
@juliasilge



 Follow

I am having to re-do some pretty onerous data cleaning work, and I am SO THANKFUL that it is all in knitr, fully reproducible, etc. #rstats

Why **knitr** for manuscripts?



Julia Silge
@juliasilge



 Follow

I am having to re-do some pretty onerous data cleaning work, and I am SO THANKFUL that it is all in knitr, fully reproducible, etc. [#rstats](#)

“I can do reproducible work in R (making me happy) and format the output report in Word (making my collaborators happy)” - Richard Layton http://rmarkdown.rstudio.com/articles_docx.html

Why **knitr** for manuscripts?

Native R Scripts (.R extensions) (or any analysis code) are generally not designed for reading, but the **knitr** library has been designed for converting R scripts into readable reports, such as Word, PDF, and/or html documents.

Not only do these types of reports help with collaborating, they provide a great framework for archiving your analyses and results.

Example:

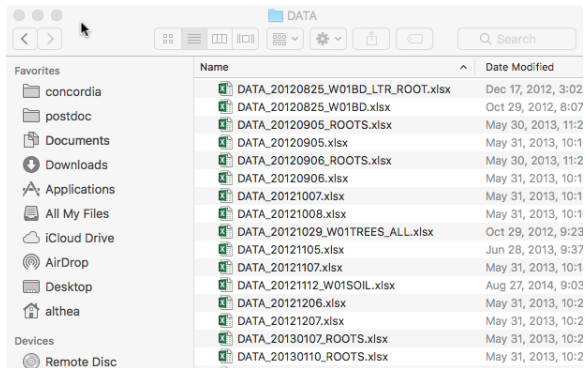
<https://conservancy.umn.edu/handle/11299/181607>

Documenting Code: General tips

- ▶ Consistent and meaningful naming conventions
 - ▶ `a = b*c`
 - ▶ `weekly.pay = hours.worked*pay.rate` (not cross-compatible)
 - ▶ `weekly_pay = hours_worked*pay_rate`
 - ▶ `weeklyPay = hoursWorked*payRate`

Documenting Code: General tips

- ▶ Consistent and meaningful naming conventions
 - ▶ $a = b * c$
 - ▶ `weekly.pay = hours.worked*pay.rate` (not cross-compatible)
 - ▶ `weekly_pay = hours_worked*pay_rate`
 - ▶ `weeklyPay = hoursWorked*payRate`
- ▶ Use YYYYMMDD or equivalent for dates



Documenting Code: R-Markdown

R-Markdown combines `markdown` language, which is “an easy-to-write plain text format” and embedded `R code chunks` that are “run so their output can be included in the final document” [1]

Documenting Code: R-Markdown

R-Markdown combines [markdown](#) language, which is “an easy-to-write plain text format” and embedded [R code chunks](#) that are “run so their output can be included in the final document” [1]

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

##	speed	dist
##	Min. : 4.0	Min. : 2.00
##	1st Qu.:12.0	1st Qu.: 26.00
##	Median :15.0	Median : 36.00
##	Mean :15.4	Mean : 42.98
##	3rd Qu.:19.0	3rd Qu.: 56.00
##	Max. :25.0	Max. :120.00

[1] www.rmarkdown.rstudio.com

Documenting code: R-Markdown

Exercise 2a: Introduction to R-Markdown

- ▶ File > New File > R Markdown...
- ▶ Choose “html” - optionally put in a title and press “OK”
- ▶ This R-Markdown template is ready to “knit” into an html as-is
 - ▶ Click the blue Knit button
 - ▶ Save as “**student_folders/yourname/activity2a_intro_rmarkdown.Rmd**”
 - ▶ View the resultant html
- ▶ Take a few minutes to modify the .Rmd and view how the changes appear in the knit html document.

<https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf>

Documenting code: ezknitr

What folder did the html end up in?

Documenting code: ezknitr

What folder did the html end up in?

Now imagine if you wanted to keep the programs/scripts in a folder separate from reports ([highly recommended!](#)). You can easily direct the output html file into a different folder using **ezknitr** package.

Documenting code: ezknitr

What folder did the html end up in?

Now imagine if you wanted to keep the programs/scripts in a folder separate from reports (**highly recommended!**). You can easily direct the output html file into a different folder using **ezknitr** package.

```
library(ezknitr)
ezknit("student_folders/yournm/activity2a_intro_rmarkdown.Rmd",
      out_dir = "student_folders/yournm/reports",
      fig_dir = "figures",
      keep_md = F)
```

Note: When using ezknit, must manually save the .Rmd file before knitting!

Documenting Code: R-Oxygen

Instead of using the R-Markdown language, you can also use **pure R scripts** plus **Roxygen comments** (`#'`) to create fully reproducible reports.

Documenting Code: R-Oxygen

Instead of using the R-Markdown language, you can also use **pure R scripts** plus **Roxygen comments** (`#'`) to create fully reproducible reports.

Benefit: The entire program can be written and run in the familiar R Script file, then “spun” into an html/Word/pdf document at any point.

- ▶ Learning to code with R-Oxygen is arguably more natural since we already use `#` for commenting

Documenting Code: R-Oxygen

Instead of using the R-Markdown language, you can also use **pure R scripts** plus **Rxygen comments** (`#'`) to create fully reproducible reports.

Benefit: The entire program can be written and run in the familiar R Script file, then “spun” into an html/Word/pdf document at any point.

- ▶ Learning to code with R-Oxygen is arguably more natural since we already use `#` for commenting

Additionally, use `#+` to define and label R chunks like we did with the `“{r ...}”` code in the R-Markdown language.

Documenting Code: R-Oxygen

Activity 2b: Introduction to R-Oxygen

Here, we'll quickly convert “activity1b_data_analysis.R” into an html document.

1. Open “activity1b_data_analysis.R”
2. Save As. . .
“student_folder/yourname/activity2b_intro_roxygen.R”

LaTeX

Another benefit of using knitr/Rmd/Roxygen for creating statistical reports is the nice interface with LaTeX equation syntax.

Activity 3. LaTeX Equations

1. Create a new .Rmd html document
2. Save it as “student_folders/yourname/activity3_latex.Rmd”
3. Create the following in the output html
 - ▶ $\alpha + \beta = 2\theta$
 - ▶ $\pi^2 = 9.86$
 - ▶ $\sum_{i=1}^n \sqrt{i} = 42$ (advanced)
4. When you “knit” remember to use ezknitr:

```
library(ezknitr)
ezknit("student_folders/yourname/activity3_latex.Rmd",
      out_dir = "student_folders/yourname/reports",
      fig_dir = "figures",
      keep_md = F)
```

<https://tobi.oetiker.ch/lshort/lshort.pdf> (hint: tables on p75)

Putting RStudio & knitr together: Project Workflow

Project Workflow w/ RStudio & knitr

Example project directory

- ▶ data/
 - ▶ raw_data/
 - ▶ processed_data/
 - ▶ output_data/
- ▶ manuscript/
 - ▶ ms_figures/
 - ▶ transmissions/
 - ▶ submission/
- ▶ output/
 - ▶ figures/
- ▶ programs/
- ▶ project_file.Rproj

Project Workflow w/ RStudio & knitr

- ▶ **data/**
 - ▶ **raw_data/**
 - ▶ survey_data20161227.csv
 - ▶ survey_data20161230.csv
 - ▶ survey_data20170103.csv
 - ▶ **processed_data/**
 - ▶ survey_data_all.Rdata
 - ▶ **output_data/**
 - ▶ model_out.Rdata

Project Workflow w/ RStudio & knitr

- ▶ **programs/**

- ▶ a_data_processing.R
- ▶ b_data_analysis.R
- ▶ c_plots.R

Project Workflow w/ RStudio & knitr

- ▶ **output/**
 - ▶ a_data_processing.html
 - ▶ b_data_analysis.html
 - ▶ c_plots.html
 - ▶ **figures/**
 - ▶ eda1.jpg
 - ▶ scatter1.jpg

Project Workflow w/ RStudio & knitr

- ▶ **manuscript/**
 - ▶ ms.Rmd
 - ▶ ms.pdf
 - ▶ ms.docx
 - ▶ **ms_figures/**
 - ▶ fig1.jpg
 - ▶ fig2.jpg

Project Workflow w/ RStudio & knitr

- ▶ **manuscript/**

- ▶ ms.Rmd
- ▶ ms.pdf
- ▶ ms.docx
- ▶ **ms_figures/**
 - ▶ fig1.jpg
 - ▶ fig2.jpg

- ▶ **transmittals/**

- ▶ **from_john/**
- ▶ ms20170523.docx
- ▶ ms20170625.docx
- ▶ **from_bob/**
- ▶ ms20170626.docx

Project Workflow w/ RStudio & knitr

- ▶ **manuscript/**

- ▶ ms.Rmd
- ▶ ms.pdf
- ▶ ms.docx
- ▶ **ms_figures/**
 - ▶ fig1.jpg
 - ▶ fig2.jpg

- ▶ **transmittals/**

- ▶ **from_john/**
- ▶ ms20170523.docx
- ▶ ms20170625.docx
- ▶ **from_bob/**
- ▶ ms20170626.docx

- ▶ **submission/**

- ▶ ms.pdf
- ▶ fig1.pdf
- ▶ fig2.pdf
- ▶ coverletter.docx

Project Workflow w/ RStudio & knitr

Example of an RStudio project that Althea & John used from conceptualization through publication.

Project Workflow w/ RStudio & knitr

Example of an RStudio project that Althea & John used from conceptualization through publication.

Activity 3: Creating a reproducible report using R-Markdown

Tasks:

1. Knit the rmd as is into a .doc file
2. Add a third week of data and update report
3. Change the formatting using the word-styles-reference-01.docx

3. GitHub for version-control, collaborating and archiving

Introduction to GitHub

GitHub provides a place for you to back-up and version-control your R projects.

- ▶ Interfaces directly with RStudio
- ▶ Free (public repositories or with university email) or cheap (private repositories)
- ▶ Facilitates collaboration with other co-authors (or the public)
- ▶ Can go back to previous code versions (version-control aspect)

Once you're up and running, it's (usually) simple to use!

Introduction to GitHub



Jonathan Tonkin

@jdtonkin



 Follow

Finally made the jump to using **#git** to version control my **#rstats** code in **#rstudio**. No more dozens of versions and cluttered folders!

Introduction to GitHub



Jonathan Tonkin

@jdtonkin



 Follow

Finally made the jump to using **#git** to version control my **#rstats** code in **#rstudio**. No more dozens of versions and cluttered folders!



Callum Macgregor

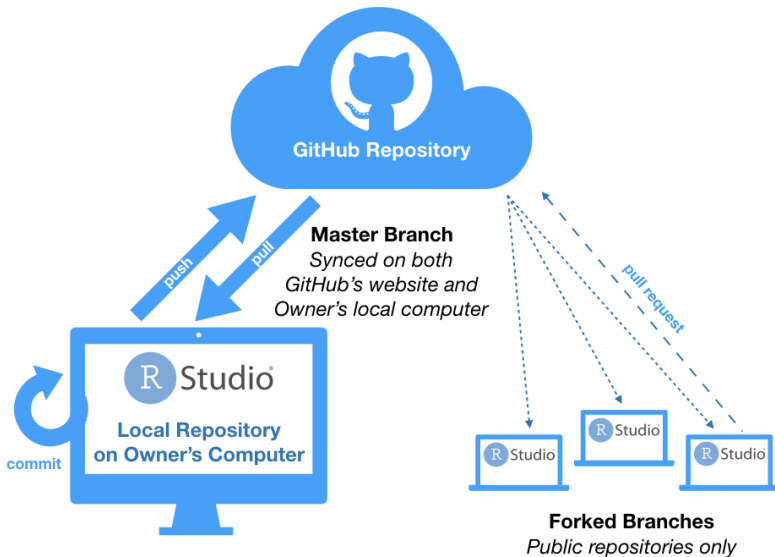
@Macgregor_Cal



 Follow

Managed to delete important R script when tidying up for Github commit. Managed to rescue it, thanks to previous Github commit!
#rstats

GitHub Overview



GitHub Terminology

► **repository**

- A project (here, an RStudio Project) that contains code, analysis, output, etc that is held by GitHub
- Repositories may be public or private
- Most repositories will have one “owner” with direct writing privileges; however,
- All repositories can be made to include more than one “owner” or “collaborator” with direct write privileges

GitHub Terminology

► **repository**

- A project (here, an RStudio Project) that contains code, analysis, output, etc that is held by GitHub
- Repositories may be public or private
- Most repositories will have one “owner” with direct writing privileges; however,
- All repositories can be made to include more than one “owner” or “collaborator” with direct write privileges

► **branch**

- A particular version of a repository
- The “master” branch is the original version of the repository, which can be forked (see below) or edited directly by branch owners/collaborators.
- “Forked” branches are any “clones” that are made by anyone other than the repository owner

GitHub Terminology

- ▶ **commit**

- ▶ Verb: to save a change or series of changes to the repository, with an associated “commit message”
- ▶ Noun: the saved changes as an entity, which can be “pushed” back to the associated branch

GitHub Terminology

▶ **commit**

- ▶ Verb: to save a change or series of changes to the repository, with an associated “commit message”
- ▶ Noun: the saved changes as an entity, which can be “pushed” back to the associated branch

▶ **push**

- ▶ To merge 1 or more commits back to the associated branch
- ▶ You can only push directly to repositories that you have ownership or collaborator privileges for
- ▶ If you don't have direct writing privileges, you have to create a “pull request”
- ▶ The direction of a push is from your local version (RStudio) to GitHub server

GitHub Terminology

- ▶ **pull**

- ▶ To take any changes that have been made to the associated branch and sync them with your local version
- ▶ The direction of a pull is from the GitHub server to your local version (RStudio)

GitHub Terminology

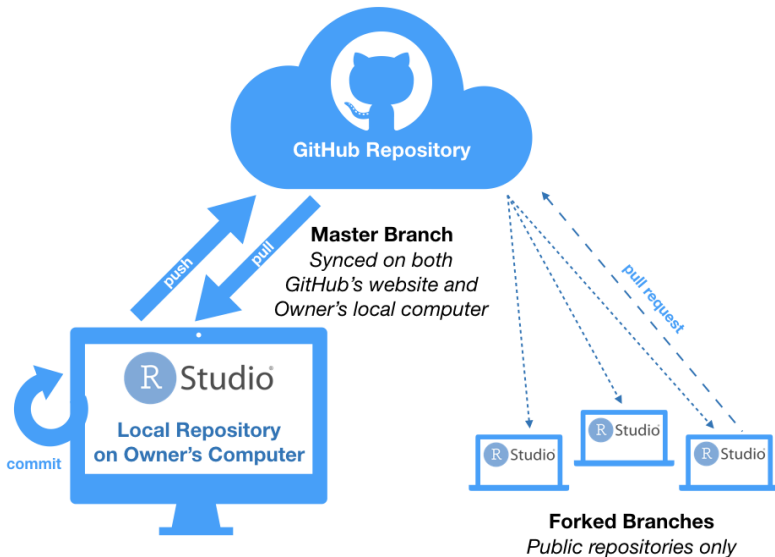
► pull

- To take any changes that have been made to the associated branch and sync them with your local version
- The direction of a pull is from the GitHub server to your local version (RStudio)

► pull request

- A series of commits that have been made on a forked branch that are being asked to merge with the next upstream branch (i.e., usually the master branch)
- You will view and accept pull requests through GitHub directly
- *Side note: Called “pull requests” because you are requesting the branch owner to pull in your changes to their branch.*

GitHub Overview



Activity 4: GitHub in RStudio

First, we will use the built-in interface for GitHub within RStudio.

Secondly, we will use the most common git commands in shell.

Activity 4: GitHub in RStudio

First, we will use the built-in interface for GitHub within RStudio.

Secondly, we will use the most common git commands in shell.

```
git pull origin master # To pull
git add data # add files to a commit
git commit -m "commit message here" #commit
git push origin master # To push commit(s)
```

Additional GitHub Tips

- ▶ Don't use github with large files (push limit of 100MB, warning >50MB)
 - ▶ If necessary, use git-large-file add on (advanced!)
- ▶ Create new projects in GitHub first, then sync them with RStudio (File > New Project > Version Control > Git)

Thanks!

We can use your feedback for the workshop at TWS. Please take a moment or two to fill out our survey.

https://docs.google.com/forms/d/e/1FAIpQLSdmvePcbb3wztx2JpJUhgTqpjvbr0z3etpnjGrRSRfTBnLpcQ/viewform?usp=sf_link

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Additional resources

- ▶ <https://swcarpentry.github.io/r-novice-gapminder/02-project-intro/>
- ▶ knitr documentation and help <https://yihui.name/knitr/>
- ▶ Markdown basics <https://daringfireball.net/projects/markdown/basics>
- ▶ R-Oxygen formatting/tips <https://rpubs.com/alobo/spintutorial>
- ▶ Online Reproducible Research Course <http://eriqande.github.io/rep-res-web/>