Reproducible Research:

Writing an R Package

Prof James P Curley

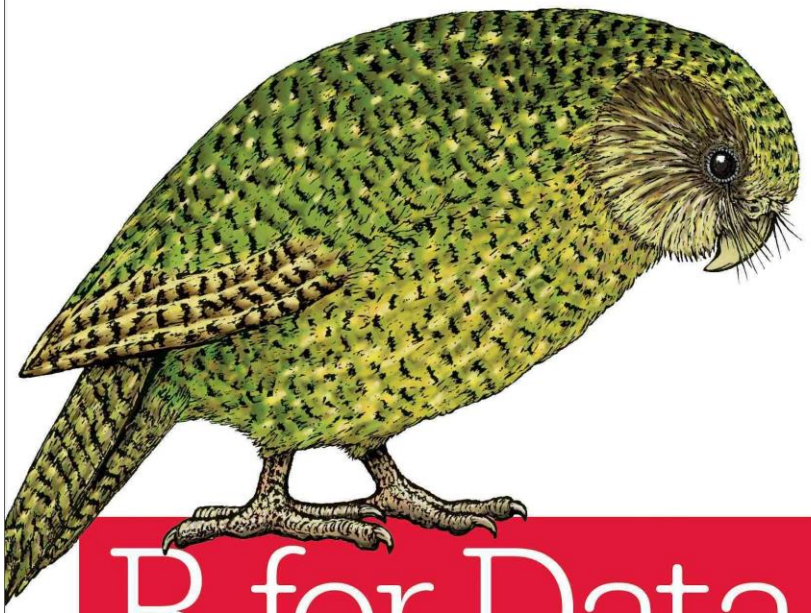
jc3181@columbia.edu

@jalapic

Materials will be available here:

- twitter.com/jalapic
- GitHub.com/jalapic/RPackage

O'REILLY®

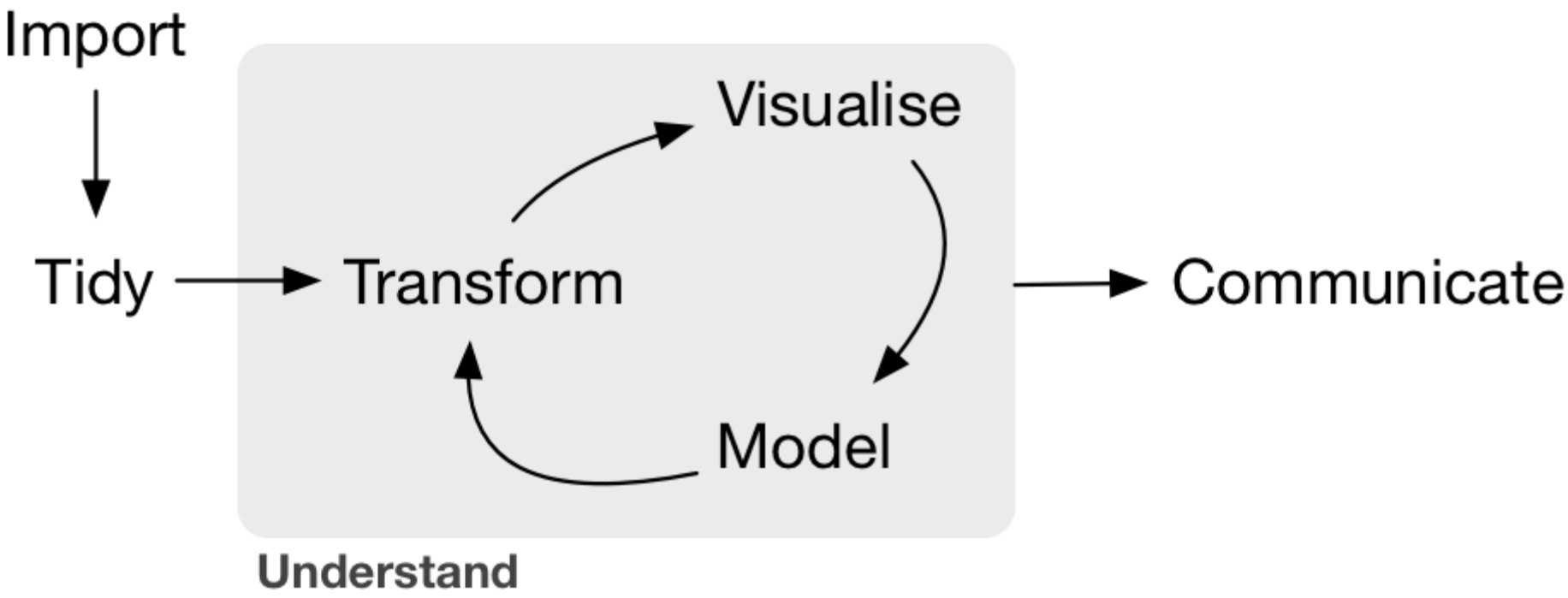


R for Data Science

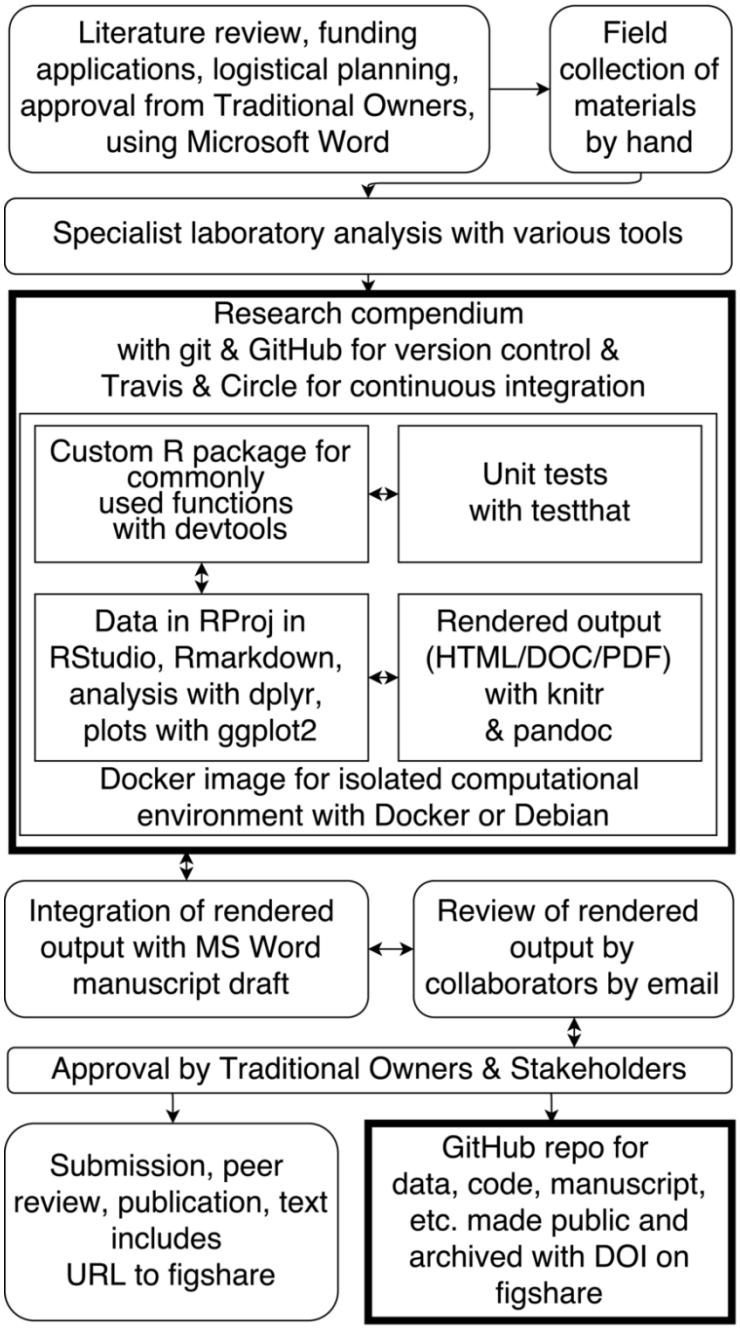
IMPORT, TIDY, TRANSFORM, VISUALIZE, AND MODEL DATA

Hadley Wickham &
Garrett Grolemund

Research Workflow – ideally automated and reproducible



Wickham & Grolemund / RStudio



Stage I: Data Input

Stage II: Data Processing

Stage III: Data Analysis

Using R & Rstudio for Reproducible Research

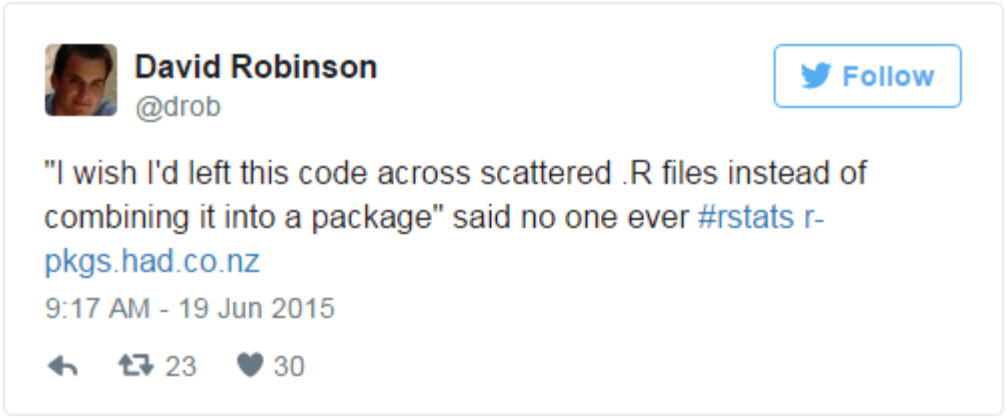
It can be overwhelming !

Let's start with one method –

Writing an R package

https://www.practicereproducibleresearch.org/case-studies/benmarwick.html

Inspiration for writing a R package



Organization: Keep all .R files and data related to a project in the same place

Documentation: Helpful documentation enables you to remember how to use functions you've written – as well as your colleagues

Collaboration: Others will be able to follow your work and reproduce it – they may even collaborate to help improve the work

Credit: Producing an R package is added productivity and can is evidence of contribution to research. Especially if the package is available on CRAN it shows that you have worked up to a high standard in making the package.

References

Very helpful tutorials/blogs:

https://www.analyticsvidhya.com/blog/2017/03/create-packages-r-cran-github/

http://web.mit.edu/insong/www/pdf/rpackage_instructions.pdf

https://hilaryparker.com/2014/04/29/writing-an-r-package-from-scratch/

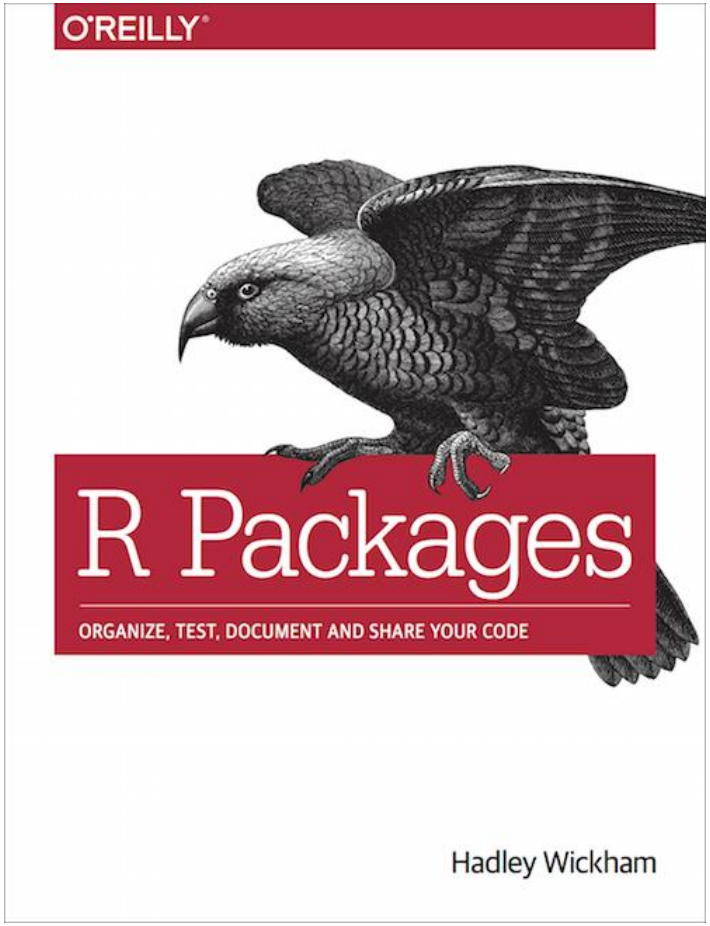
http://kbroman.org/pkg_primer/

http://stat545.com/packages00_index.html

http://tinyheero.github.io/jekyll/update/2015/07/26/making-your-first-R-package.html

If you're bold:

https://cran.r-project.org/doc/manuals/r-release/R-exts.html



The BEST resource

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

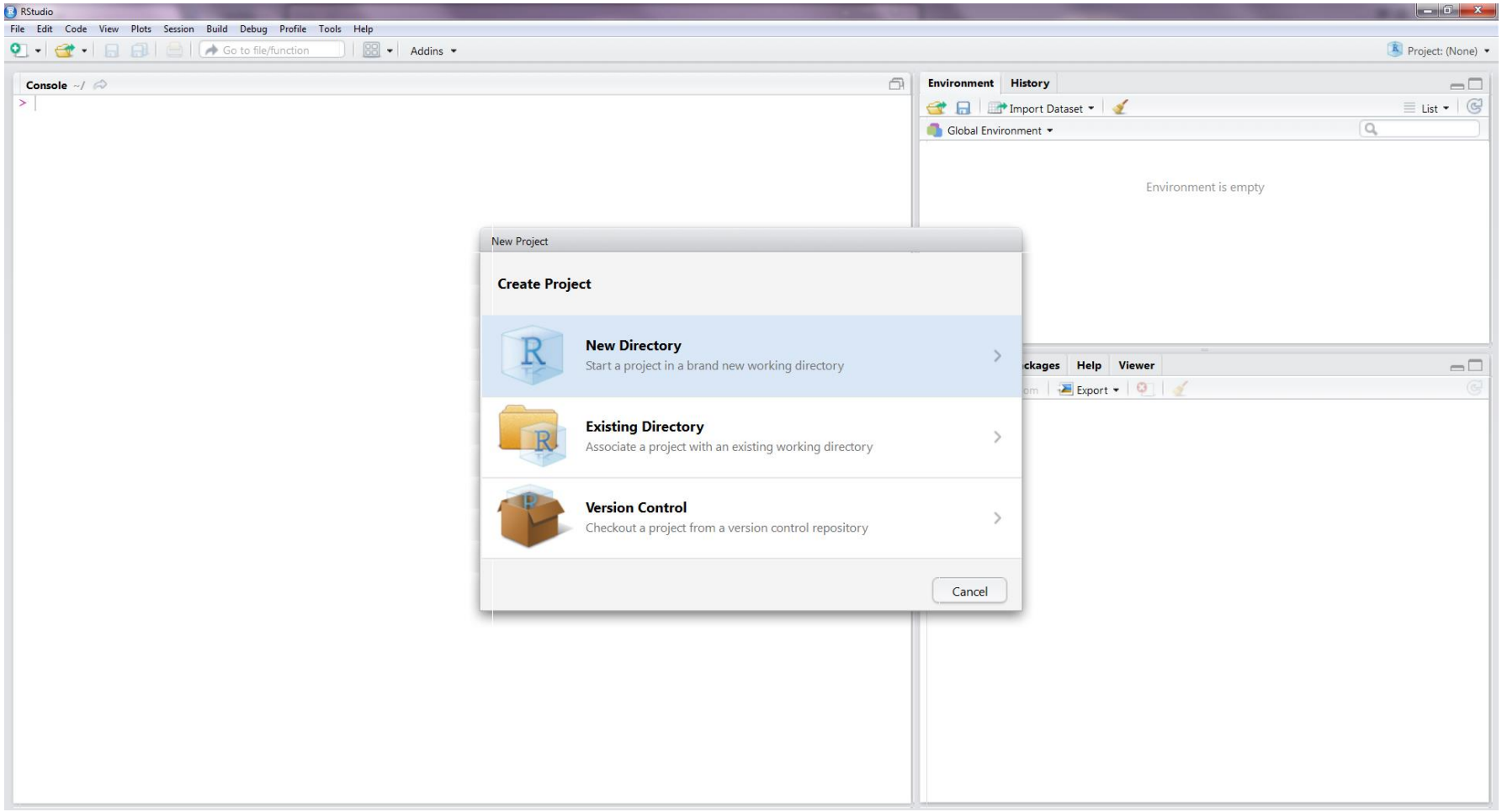
Writing our first basic R package

We will use Rstudio to build our package utilizing two packages:

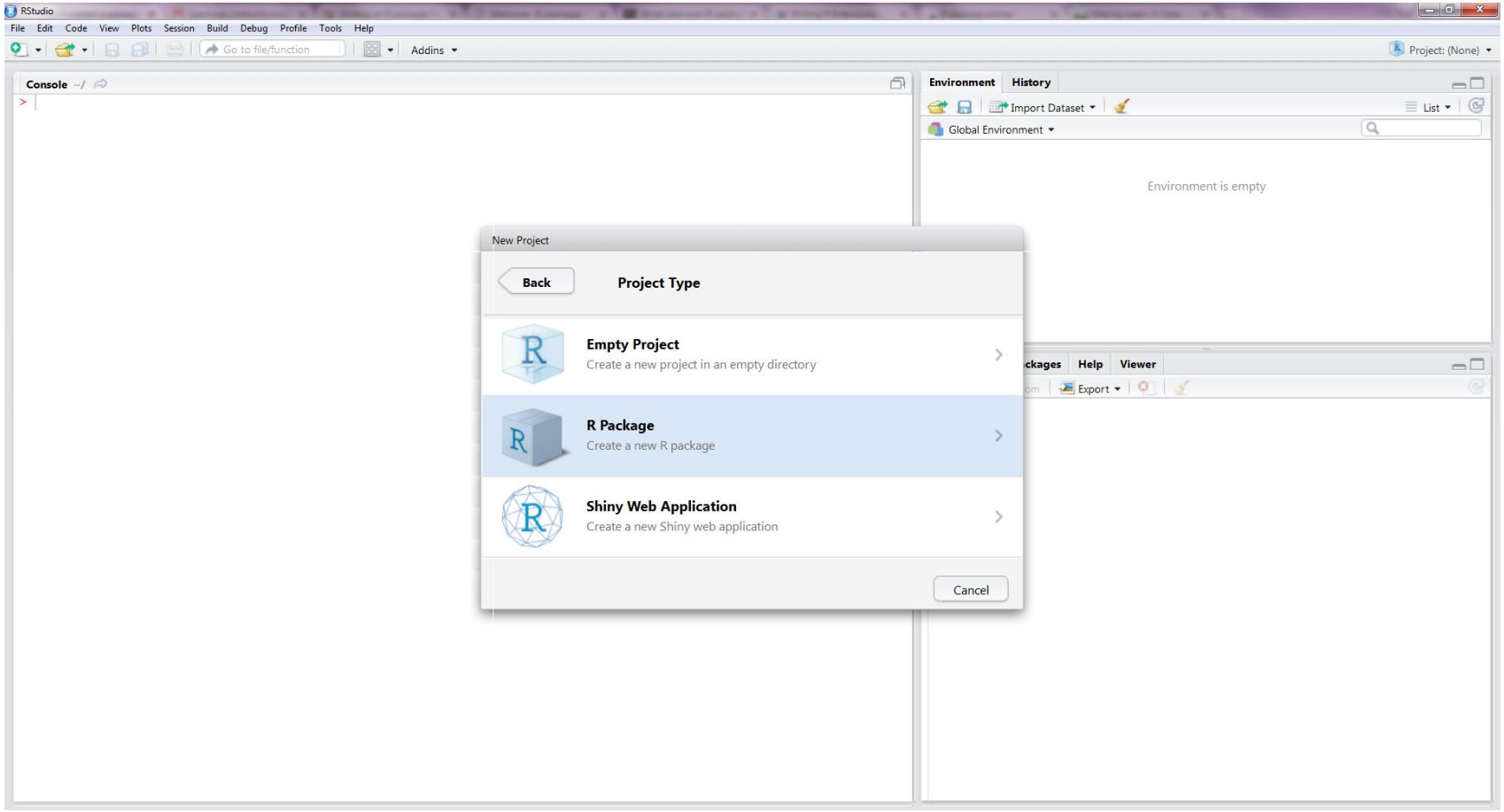
devtools

roxygen2

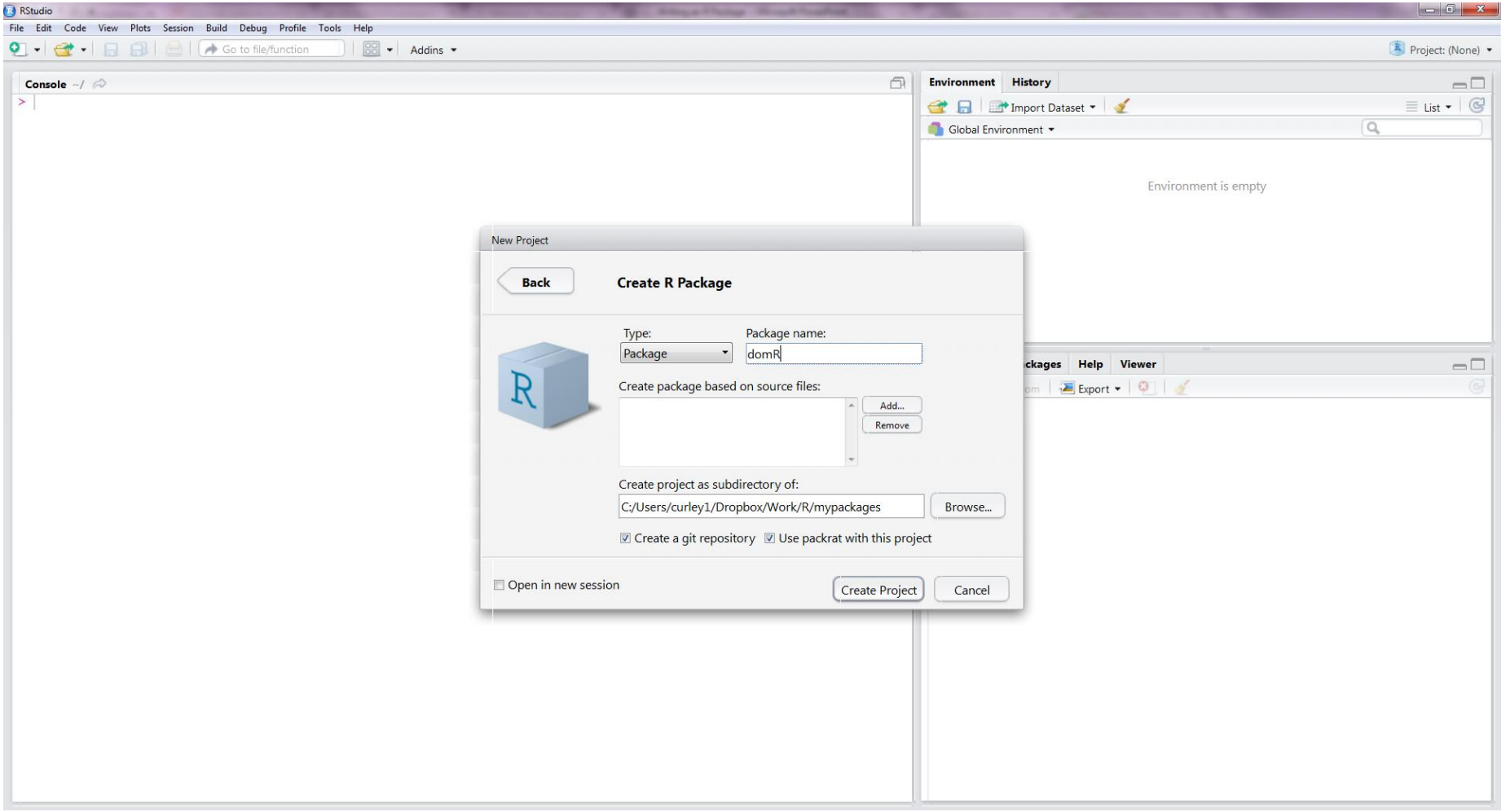
Project > New Project or File > New Project



New Directory > R Package

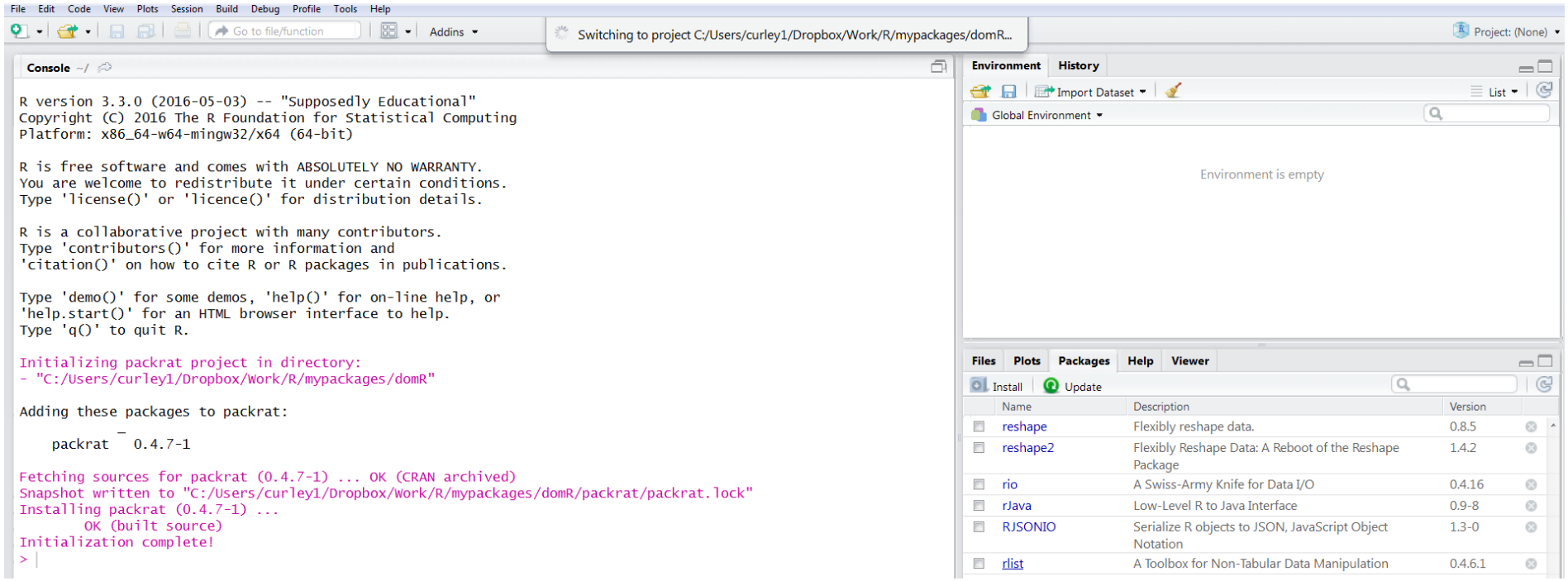


Choose a package name and a folder location

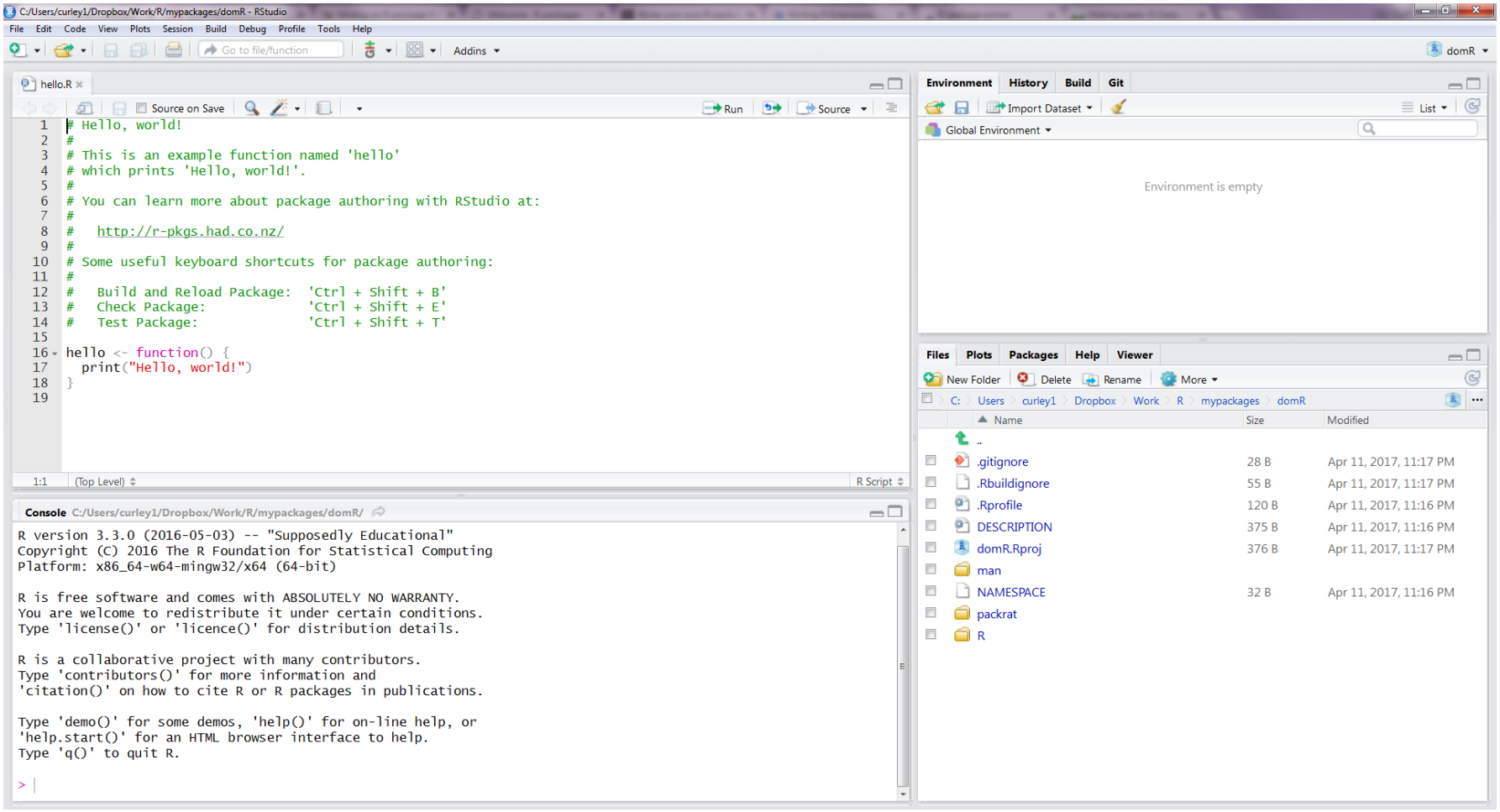


Why “packrat” ?

You should see this happen….



Several Files will be auto-generated in your folder like this: (I'll explain what these are)



Populate the DESCRIPTION file:

http://r-pkgs.had.co.nz/description.html

From this

```
Package: domR
Type: Package
Title: What the Package Does (Title Case)
Version: 0.1.0
Author: Who wrote it
Maintainer: The package maintainer <yourself@somewhere.net>
Description: More about what it does (maybe more than one line)
    Use four spaces when indenting paragraphs within the Description.
License: What license is it under?
Encoding: UTF-8
LazyData: true
```

To something like this

```
Package: compete
Title: Analyzing Social Hierarchies
Type: Package
Version: 0.1
Author: c(person("James P.", "Curley", role = c("aut", "cre"),
    email = "jc3181@columbia.edu")
Maintainer: James P. Curley <jc3181@columbia.edu>
Description: Organizing and Analyzing Social Dominance
    Hierarchy Data.
Depends:
    R (>= 3.1.0)
License: GPL-3
LazyData: true
Imports:
    graphics,
    igraph,
    sna,
    stats,
    utils
URL: https://github.com/jalapic/compete
BugReports: https://github.com/jalapic/compete
RoxygenNote: 5.0.0

```

If package uses functions from other packages, include them in the "Imports" section.

It's better in most cases to put them in "Imports" rather than "Depends"

Add .R functions to R folder

R code

The first principle of using a package is that all R code goes in `R/`. In this chapter, you'll learn about the `R/` directory, my recommendations for organising your functions into files, and some general tips on good style. You'll also learn about some important differences between functions in scripts and functions in packages.

R code workflow

The first practical advantage to using a package is that it's easy to re-load your code. You can either run `devtools::load_all()`, or in RStudio press **Ctrl/Cmd + Shift + L**, which also saves all open files, saving you a keystroke.

This keyboard shortcut leads to a fluid development workflow:

1. Edit an R file.
2. Press Ctrl/Cmd + Shift + L.
3. Explore the code in the console.
4. Rinse and repeat.

Congratulations! You've learned your first package development workflow. Even if you learn nothing else from this book, you'll have gained a useful workflow for editing and reloading R code.

See http://r-pkgs.had.co.nz/r.html for advice on code style and best practice

Add roxygen comments to top of .R functions script

The documentation workflow

In this section, we'll first go over a rough outline of the complete documentation workflow. Then, we'll dive into each step individually. There are four basic steps:

1. Add roxygen comments to your `.R` files.
2. Run `devtools::document()` (or press Ctrl/Cmd + Shift + D in RStudio) to convert roxygen comments to `.Rd` files. (`devtools::document()` calls `roxygen2::roxygenise()` to do the hard work.)
3. Preview documentation with `?`.
4. Rinse and repeat until the documentation looks the way you want.

The process starts when you add roxygen comments to your source file: roxygen comments start with `#'` to distinguish them from regular comments. Here's documentation for a simple function:

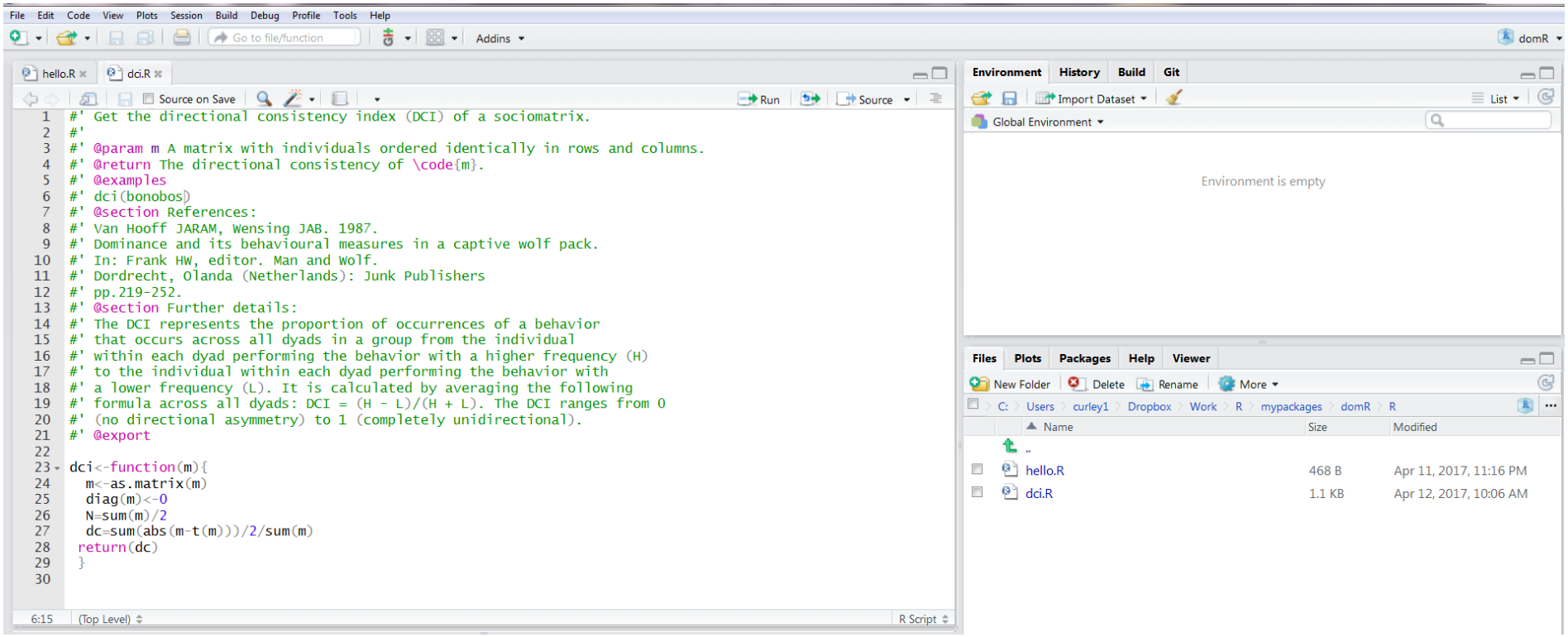
```
#' Add together two numbers.
#'
#' @param x A number.
#' @param y A number.
#' @return The sum of \code{x} and \code{y}.
#' @examples
#' add(1, 1)
#' add(10, 1)
add <- function(x, y) {
  x + y
}
```

See http://r-pkgs.had.co.nz/man.html for style guide and best practice

roxygen comments

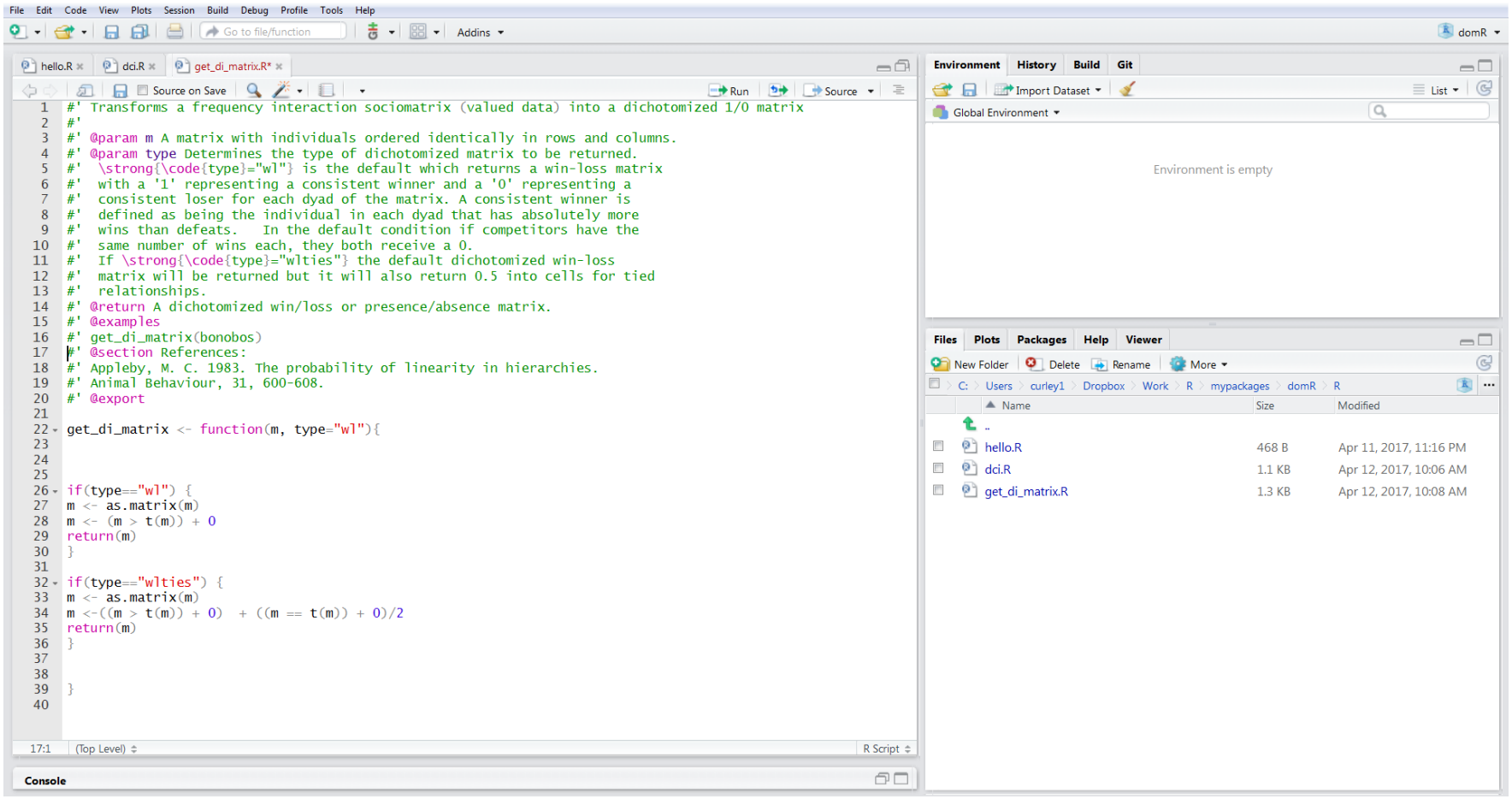
title	= title of the function.
description	= detailed explanation of what the package does.
param	= parameters/arguments in function. You can use multiple 'param' return = what object the function returns
importFrom	= functions you need from other packages
examples	= sample code (keep it small)
section	= anything else relevant that you might want to add
export	= function to export

Add .R functions to R folder and add roxygen comments

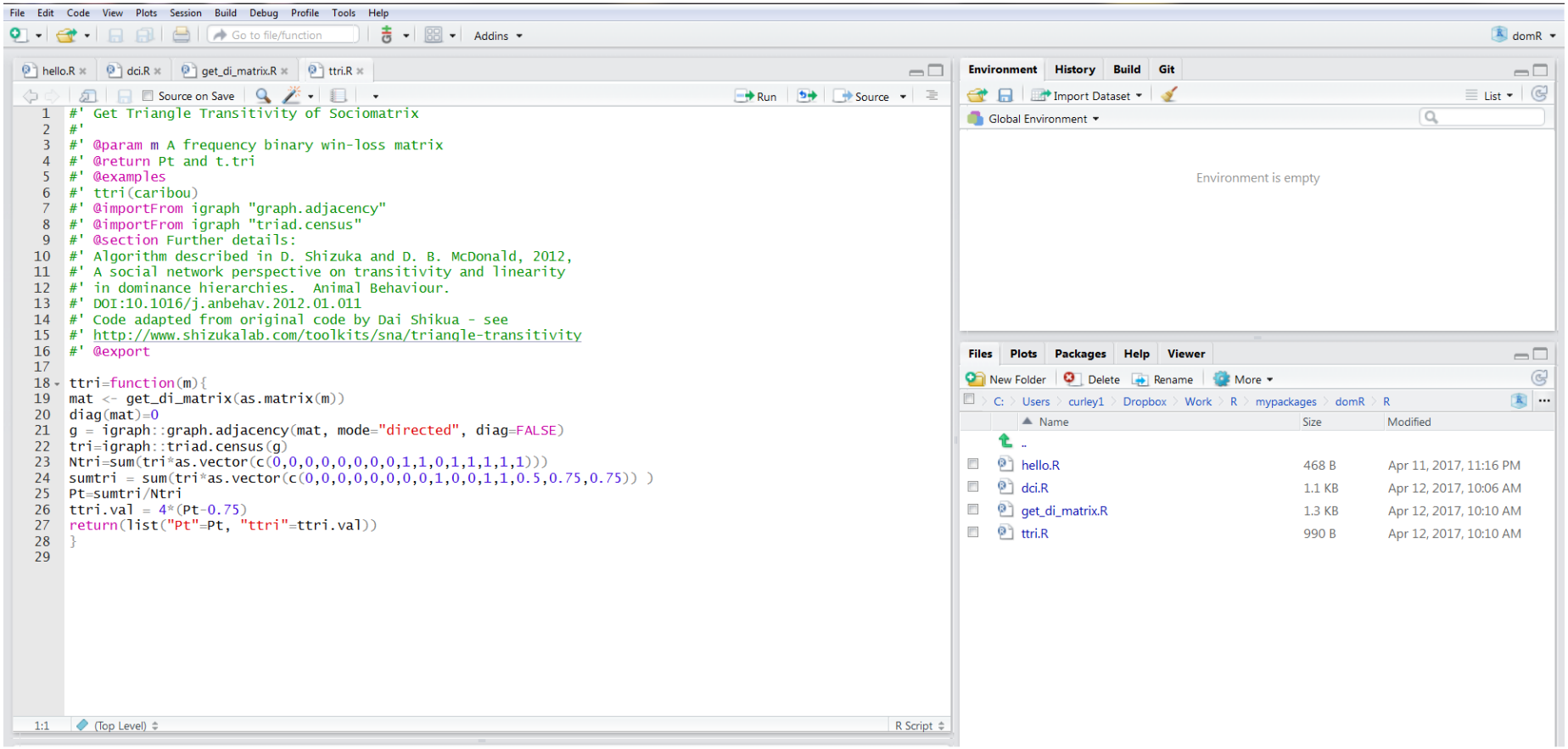


Save functions with name of function

Do for next function …



Keep going…


File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
domR
hello.R
dci.R
get_di_matrix.R
ttri.R
Source on Save
Run
Source
#' Get Triangle Transitivity of Sociomatrix
#'
#' @param m A frequency binary win-loss matrix
#' @return Pt and t.tri
#' @examples
#' ttri(caribou)
#' @importFrom igraph "graph.adjacency"
#' @importFrom igraph "triad.census"
#' @section Further details:
#' Algorithm described in D. Shizuka and D. B. McDonald, 2012,
#' A social network perspective on transitivity and linearity
#' in dominance hierarchies. Animal Behaviour.
#' DOI:10.1016/j.anbehav.2012.01.011
#' Code adapted from original code by Dai Shikua - see
#' http://www.shizukalab.com/toolkits/sna/triangle-transitivity
#' @export
ttri=function(m){
mat <- get_di_matrix(as.matrix(m))
diag(mat)=0
g = igraph::graph.adjacency(mat, mode="directed", diag=FALSE)
tri=igraph::triad.census(g)
Ntri=sum(tri*as.vector(c(0,0,0,0,0,0,0,0,1,1,0,1,1,1,1,1)))
sumtri = sum(tri*as.vector(c(0,0,0,0,0,0,0,0,0,1,0,0,1,1,0.5,0.75,0.75)))
Pt=sumtri/Ntri
ttri.val = 4*(Pt-0.75)
return(list("Pt"=Pt, "ttri"=ttri.val))
}
1:1
(Top Level)
R Script
Environment
History
Build
Git
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
C: Users curley1 Dropbox Work R mypackages domR R
Name
Size
Modified
..
hello.R 468 B Apr 11, 2017, 11:16 PM
dci.R 1.1 KB Apr 12, 2017, 10:06 AM
get_di_matrix.R 1.3 KB Apr 12, 2017, 10:10 AM
ttri.R 990 B Apr 12, 2017, 10:10 AM

Load .R files …

```
> devtools::load_all()
Loading domR
Error in (function (dep_name, dep_ver = NA, dep_compare = NA)  :
  Dependency package igraph not available.
> |
```

Fix error !

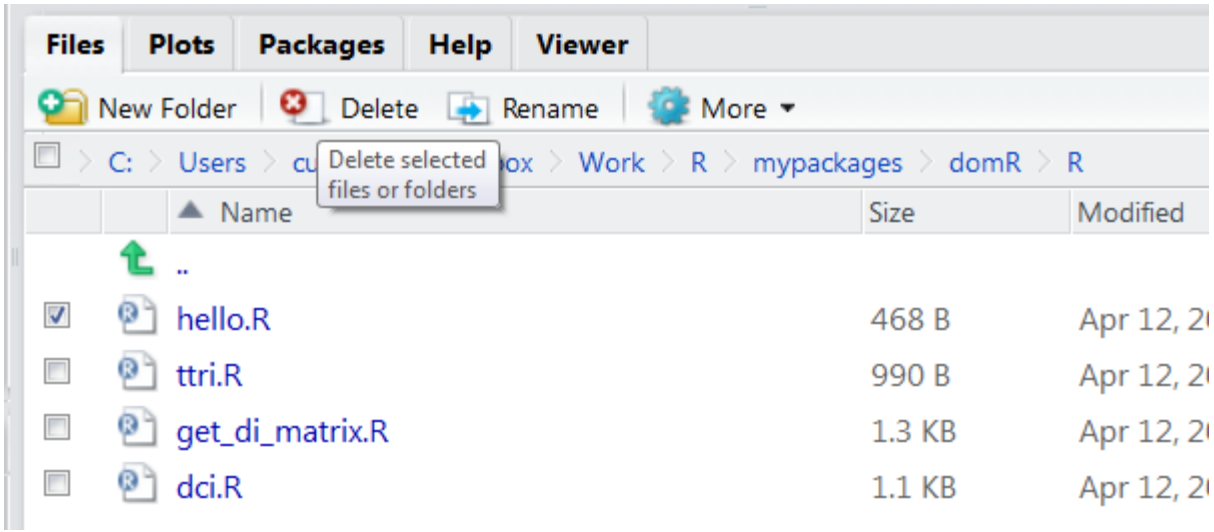
```
> install.packages("igraph")
Installing package into 'C:/Users/curley1/Dropbox/Work/R/mypackages/domR/packrat/lib/x86_64-w64-mingw32/3.3.0'
(as 'lib' is unspecified)
also installing the dependencies 'dichromat', 'munsell', 'labeling', 'xtable', 'iterators', 'gtable', 'plyr', 's
cales', 'tibble', 'lazyeval', 'pkgmaker', 'registry', 'rngtools', 'gridBase', 'colorspace', 'RColorBrewer', 'for
each', 'doParallel', 'ggplot2', 'reshape2', 'NMF', 'irlba'

trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.3/dichromat_2.0-0.zip'
Content type 'application/zip' length 147728 bytes (144 KB)
|
```

```
The downloaded binary packages are in
        C:\Users\curley1\AppData\Local\Temp\RtmpqUPtze\downloaded_packages
> devtools::load_all()
Loading domR
> |
```

We're good

Deleting files

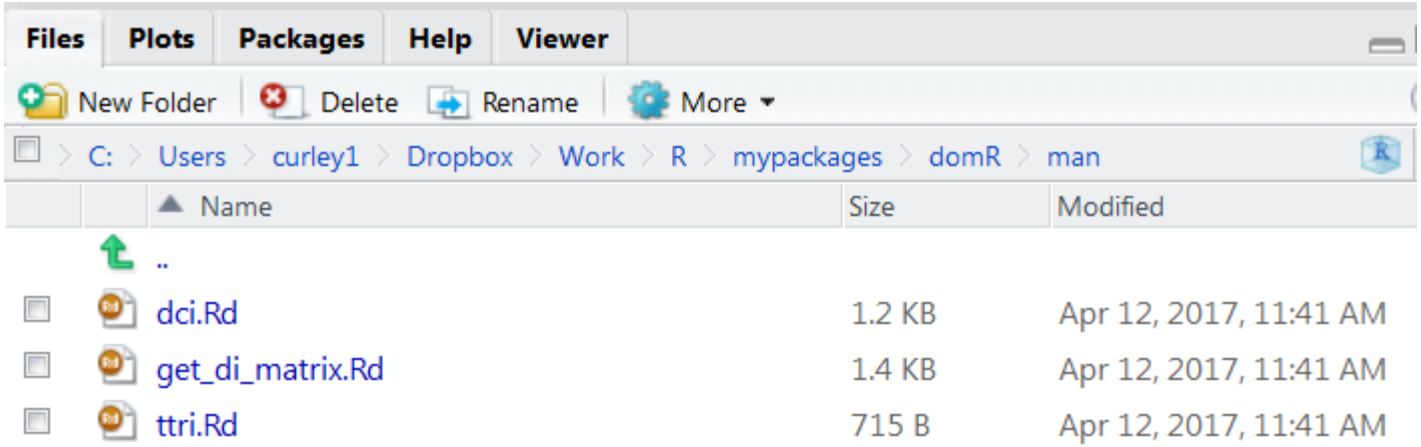
If we don't want a file – remove it's .R file and its .Rd file (if it exists).
Easiest way is with the Delete icon in Files window



Add documentation
Man

devtools::document()

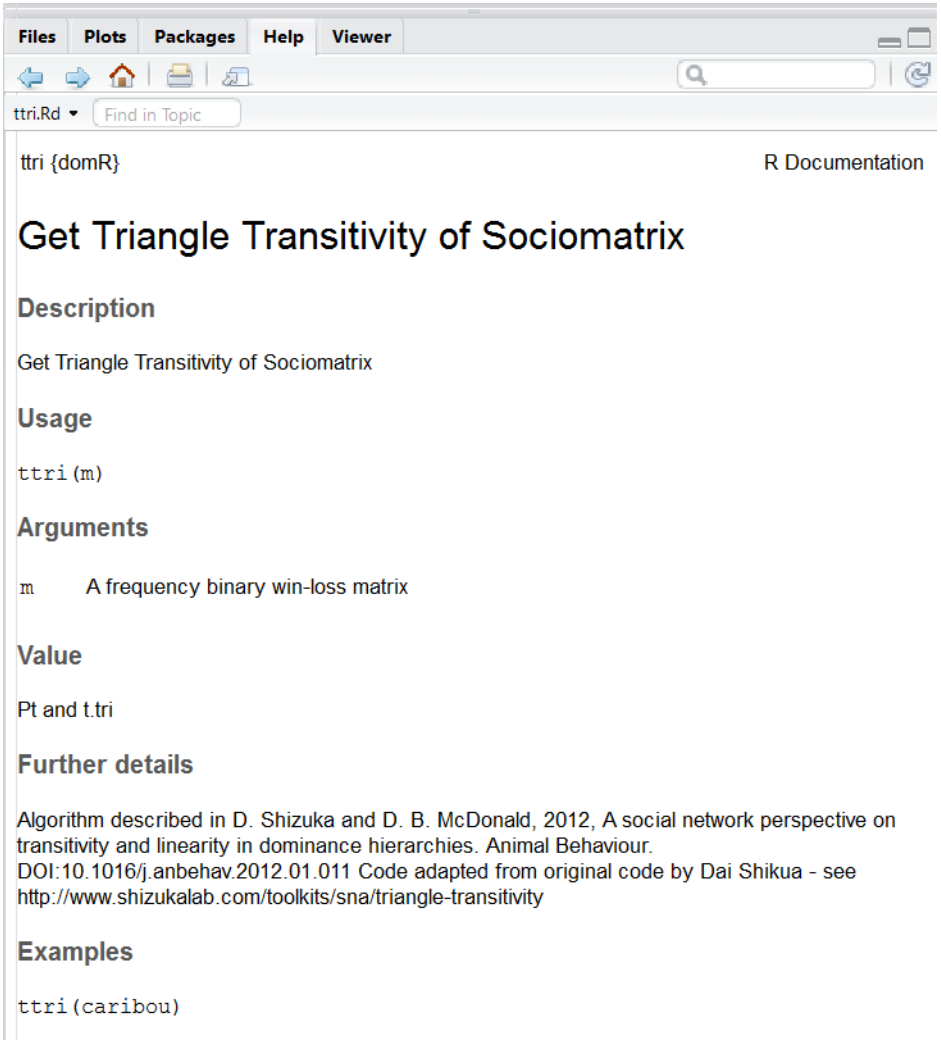
```
> devtools::document()
Updating domR documentation
Loading domR
First time using roxygen2. Upgrading automatically...
Warning: The existing 'NAMESPACE' file was not generated by roxygen2, and will not be overwritten.
Writing dci.Rd
Writing get_di_matrix.Rd
Writing ttri.Rd
>
```



We now have docs !

Check them out with “?”

```
> ?ttri
Using development documentation for ttri
> 
```



Files | Plots | Packages | Help | Viewer

ttri.Rd | Find in Topic

ttri {domR} R Documentation

Get Triangle Transitivity of Sociomatrix

Description

Get Triangle Transitivity of Sociomatrix

Usage

```
ttri(m)
```

Arguments

m	A frequency binary win-loss matrix

Value

Pt and t.tri

Further details

Algorithm described in D. Shizuka and D. B. McDonald, 2012, A social network perspective on transitivity and linearity in dominance hierarchies. Animal Behaviour. DOI:10.1016/j.anbehav.2012.01.011 Code adapted from original code by Dai Shikua - see http://www.shizukalab.com/toolkits/sna/triangle-transitivity

Examples

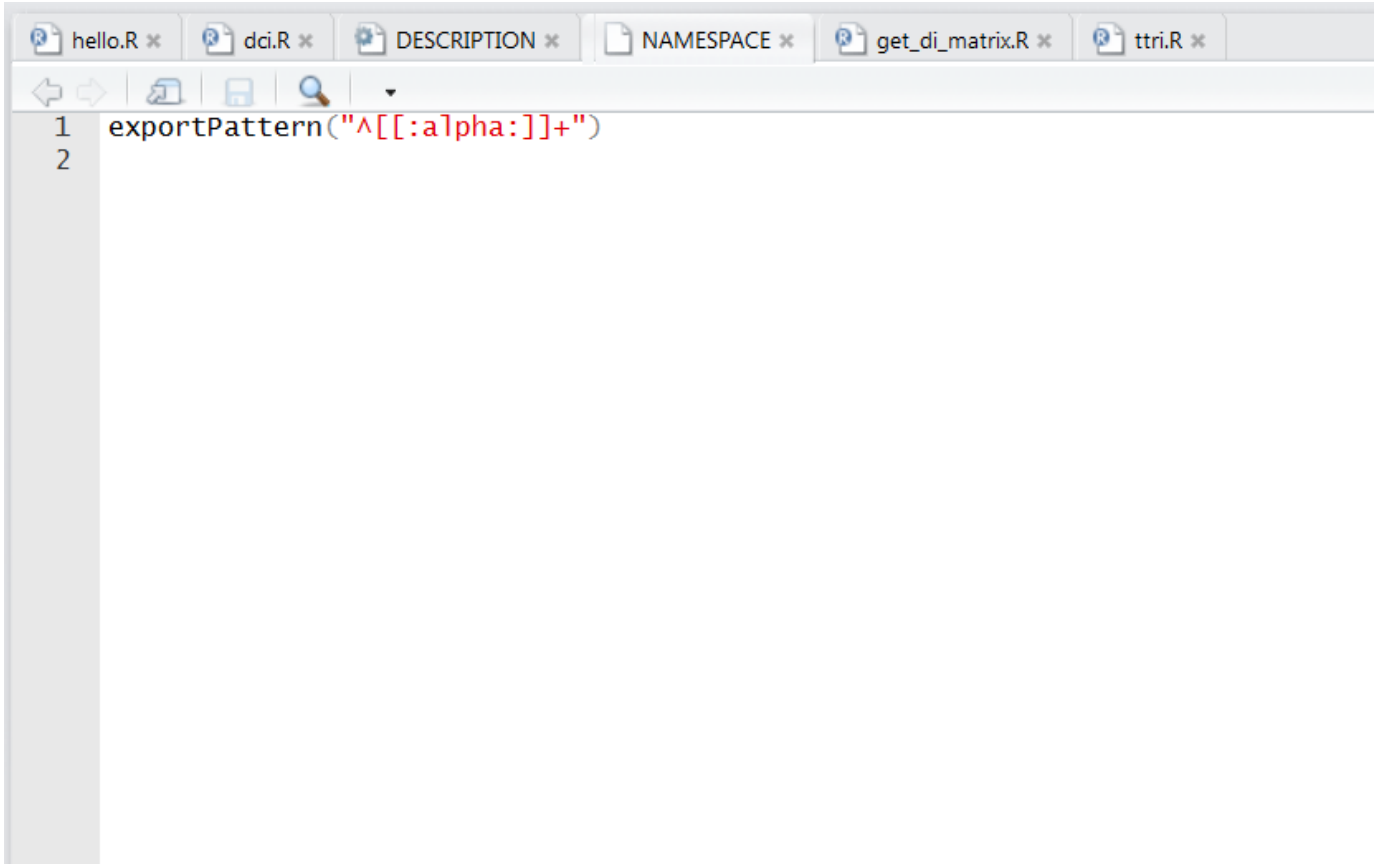
```
ttri(caribou)
```

```
Warning: The existing 'NAMESPACE' file was not generated by roxygen2, and will not be overwritten.
```

What does this mean ???

NAMESPACE

This is what the NAMESPACE looked like on initial package set up:



NAMESPACE

Google it:

```
Warning: The existing 'NAMESPACE' file was not generated by roxygen2, and will not be overwritten.
```

Yihui Xie-2

378 posts

Feb 06, 2017; 11:13am **Re: roxygen2 v6.0.0**

If your package source is version controlled (meaning you are free to regret any time), I'd recommend you to delete the three files NAMESPACE, chr.Rd, and essai-package.Rd. Then try to roxygenize again. Basically the warnings you saw indicates that roxygen2 failed to find the line

% Generated by roxygen2: do not edit by hand

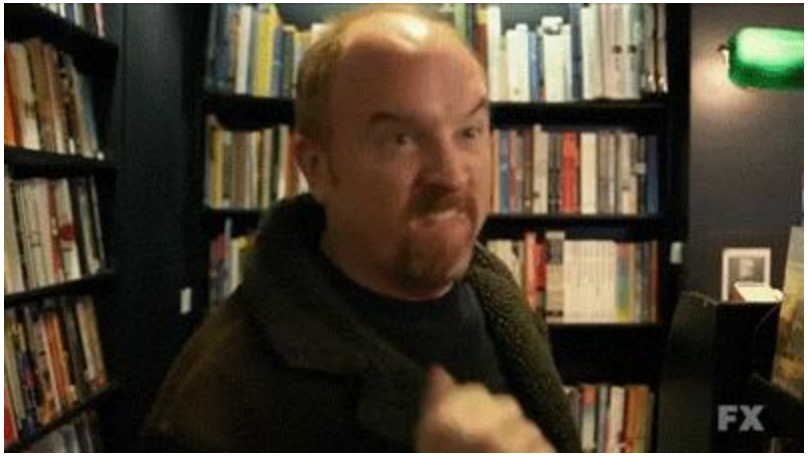
in your NAMESPACE and .Rd files, so it thinks these files were probably not previously generated by roxygen2. I think the cause is package.skeleton(), which generated the Rd files. Seriously, friends don't let friends use package.skeleton()... (it is 2017 now)

Regards,
Yihui
--
Yihui Xie <[hidden email]>
Web: http://yihui.name

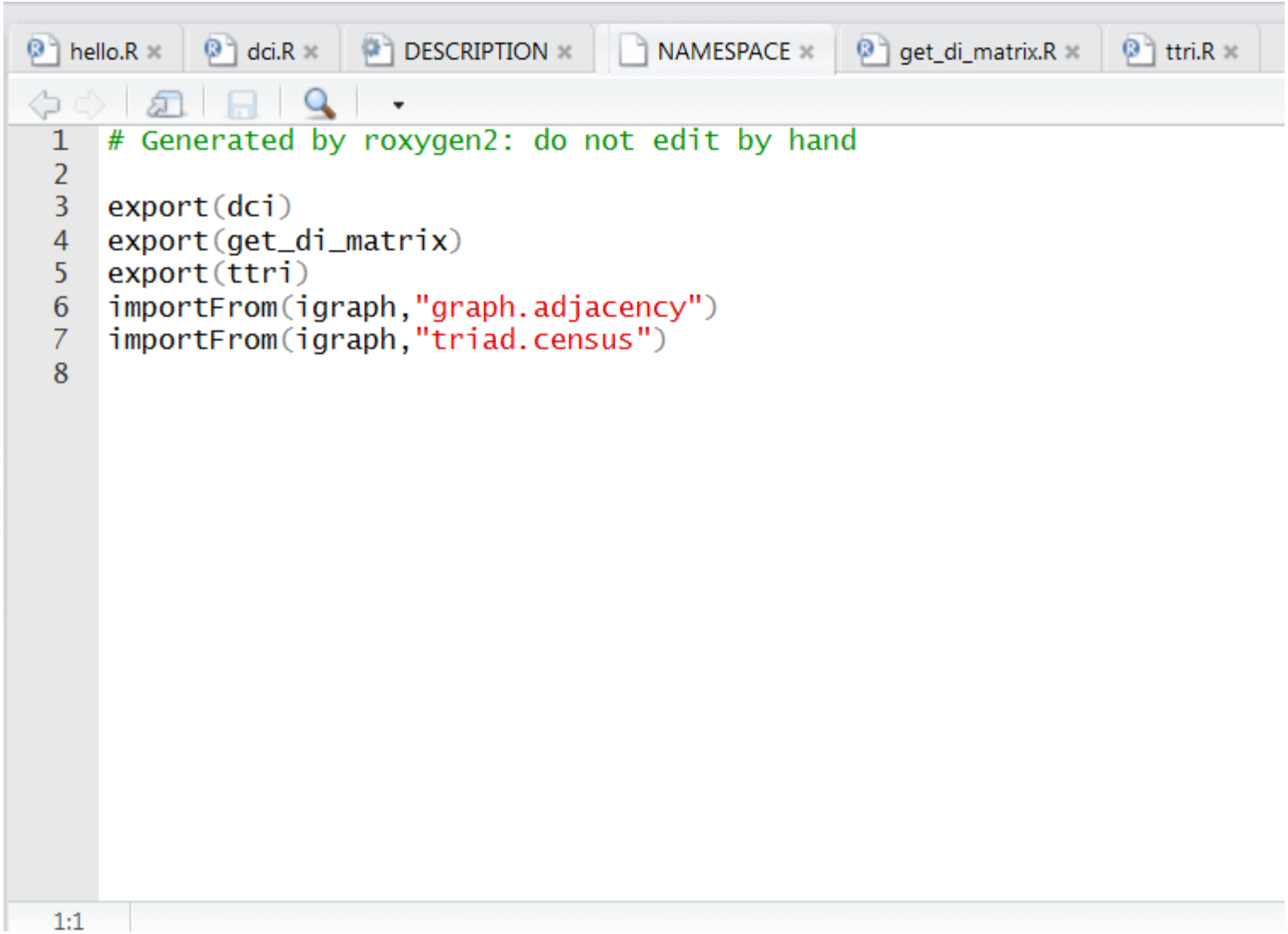
Do as Yihui says …

Delete .Rd files we just made, Delete the NAMESPACE, problem fixed !!!

```
> devtools::document()
Updating domR documentation
Loading domR
First time using roxygen2. Upgrading automatically...
Writing NAMESPACE
Writing dci.Rd
Writing get_di_matrix.Rd
Writing ttri.Rd
>
```



That's better – it should have the "do not edit by hand"



For every new .R file you will need to run:

devtools::document()

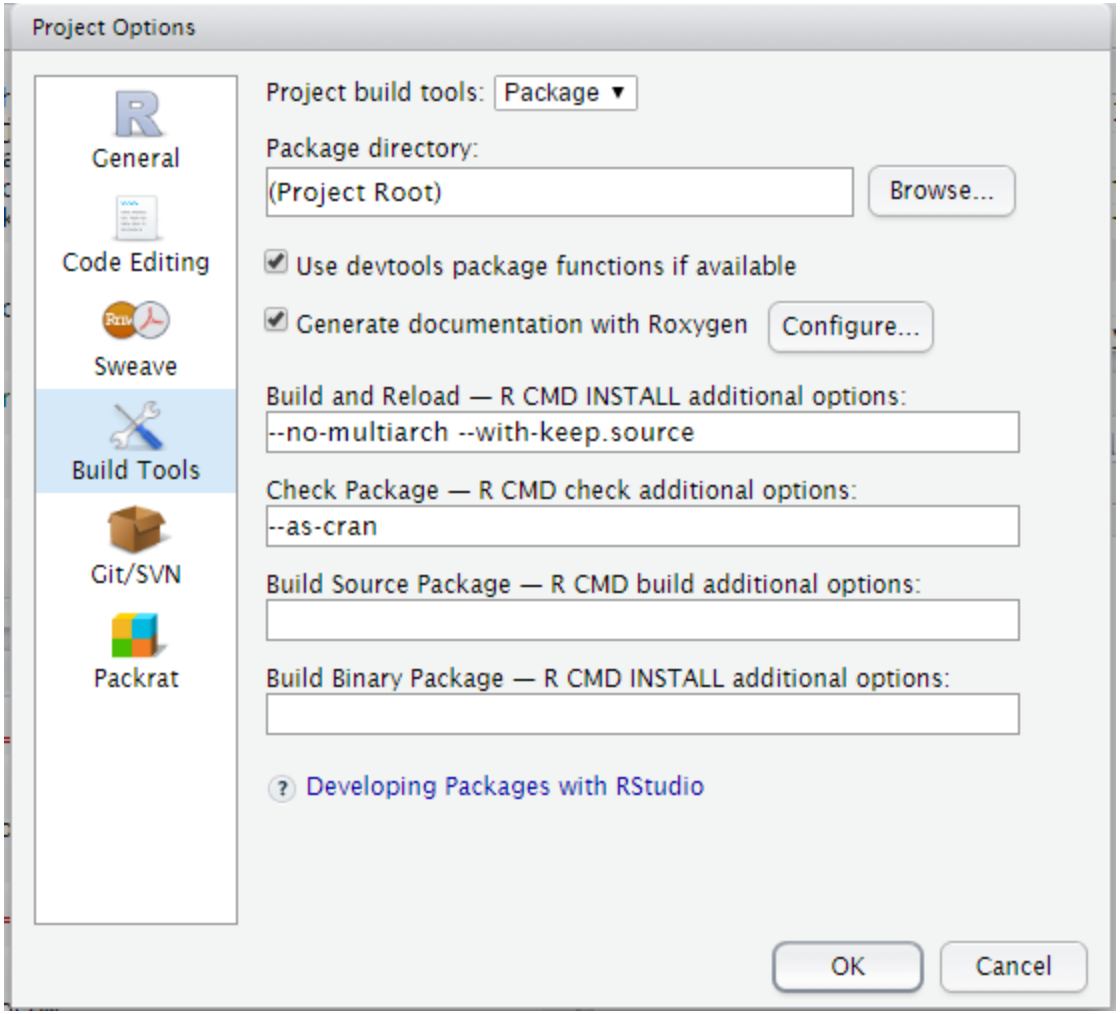
To generate your documentation.

You will also need to run it if you change the information in any .R file
- This often happens after you run CHECKS and realize you made some errors in the text.

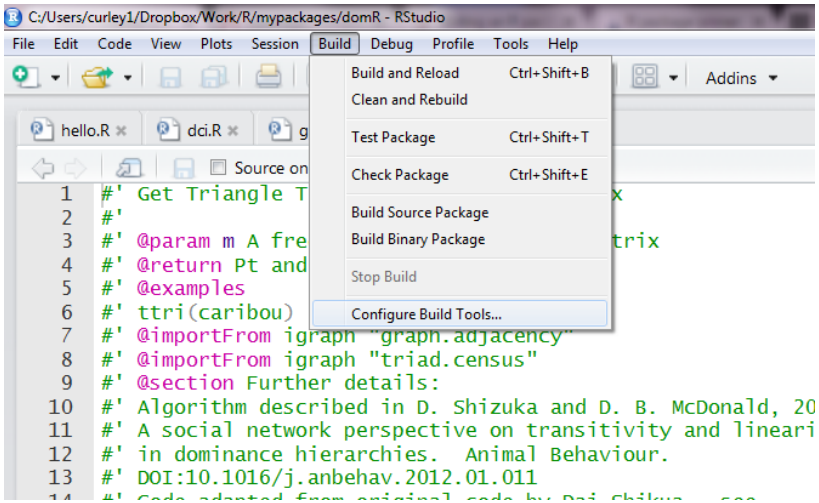
BUILD!

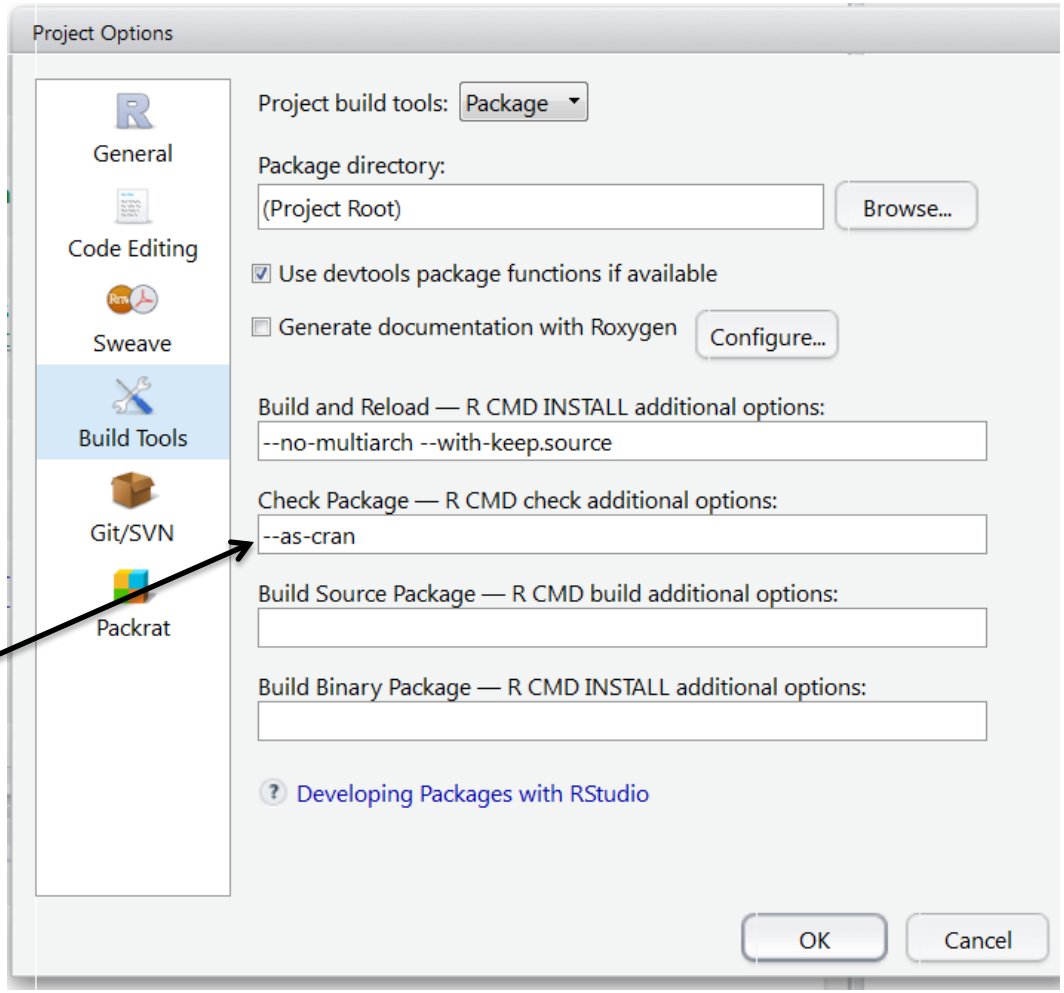
Build > Configure Build Tools.

Now check the “Generate documentation with Roxygen” option and put “–as-cran” under Check Package space to simulate the CRAN package checking and testing.



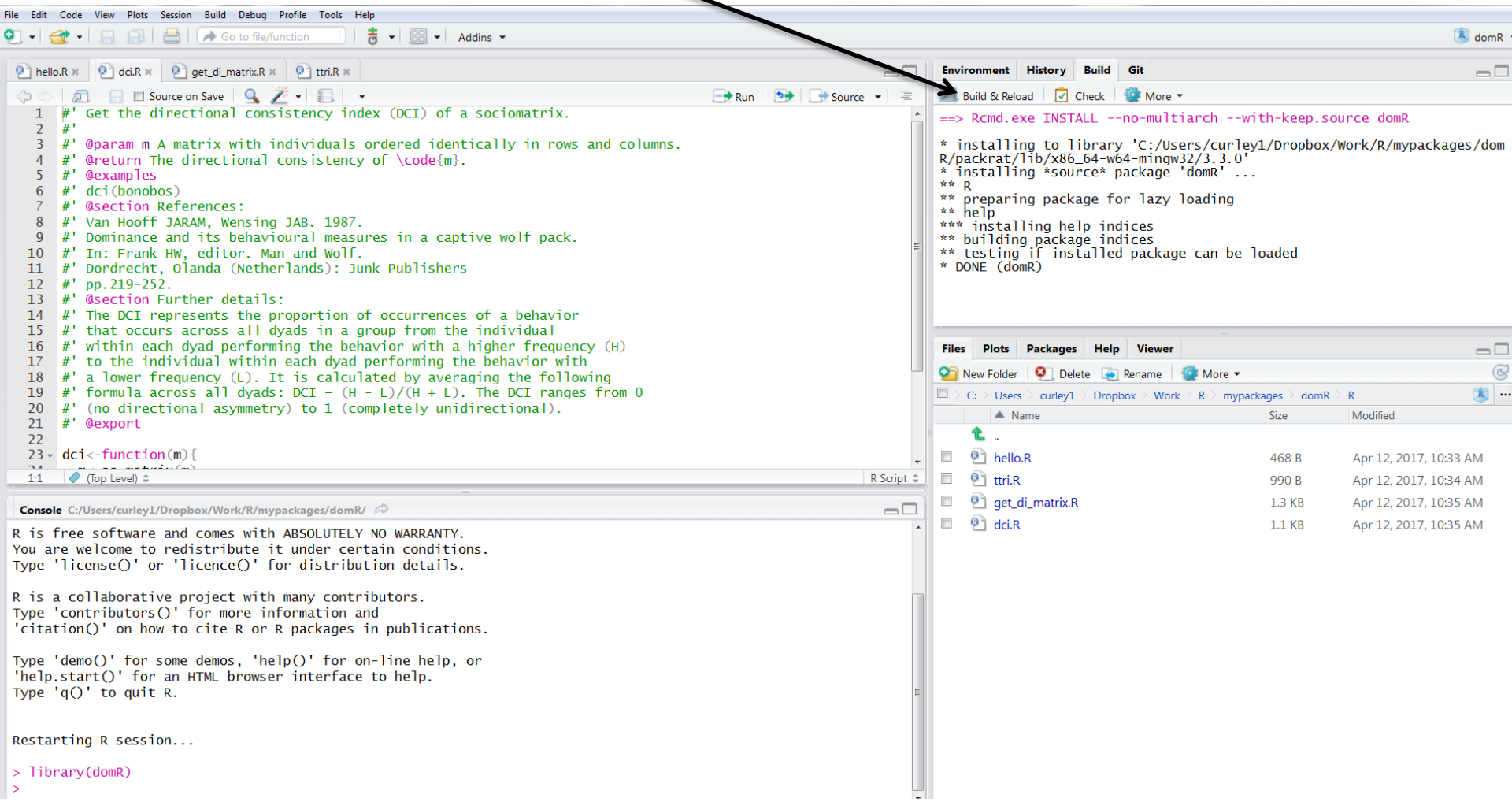
Configure Build Tools … (this is for CRAN checks so not entirely necessary for all)





Add: --as-cran

Build & Reload



You should be able to start using your functions now like a regular package:

```
Restarting R session...

> library(domR)
> m <- matrix(c(NA,2,30,6,19,122,0,NA,18,
+               0,19,85,0,1,NA,3,8,84,0,0,0,NA,267,50,0,
+               0,0,5,NA,10,1,0,4,4,1,NA), ncol=6)
>
> m
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   NA    0    0    0    0    1
[2,]    2   NA    1    0    0    0
[3,]   30   18   NA    0    0    4
[4,]    6    0    3   NA    5    4
[5,]   19   19    8  267   NA    1
[6,]  122   85   84   50   10   NA
> ttri(m)
$Pt
[1] 1

$ttri
[1] 1

> |
```

Often you will want to add data to the package:

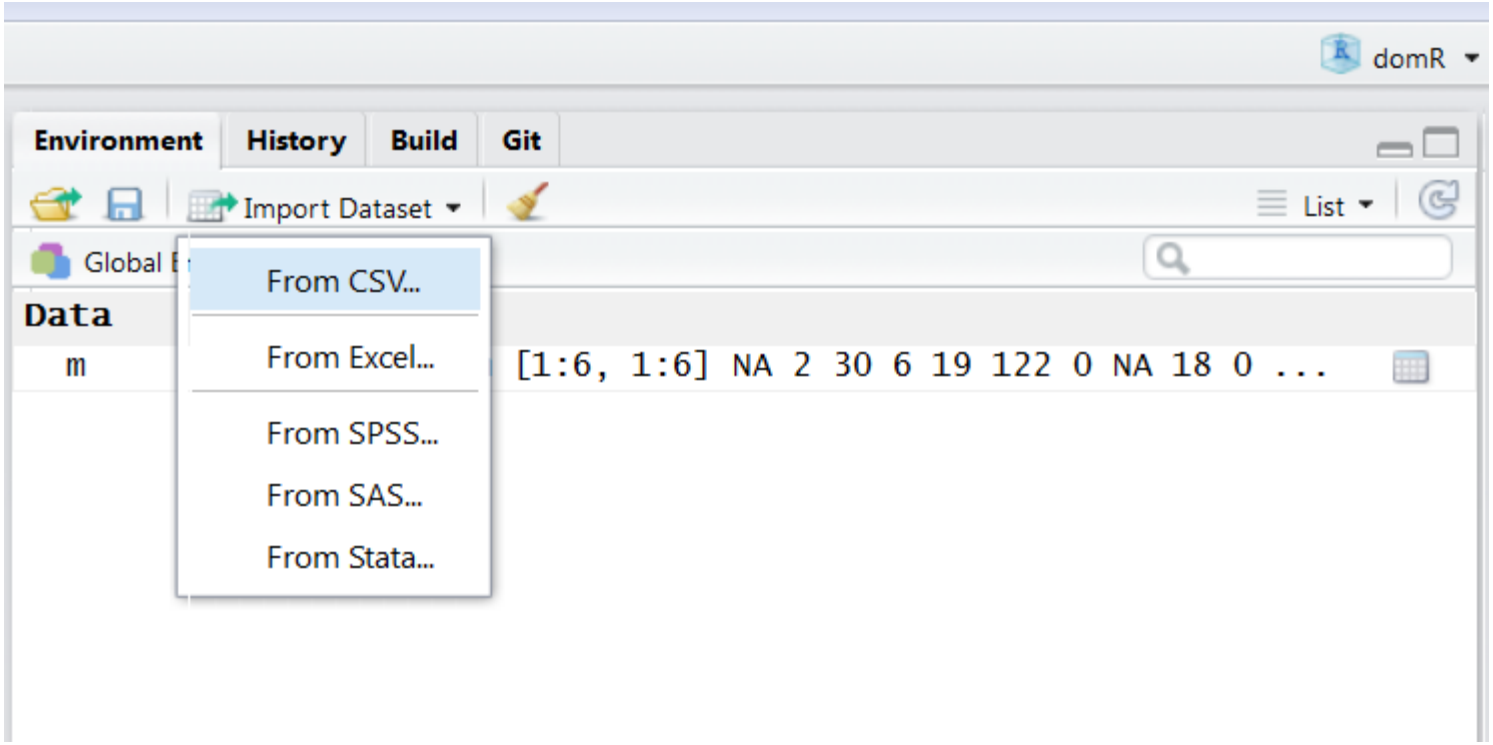
- for examples
- for users
- for functions

It's worth checking that the working directory is your project at this point:

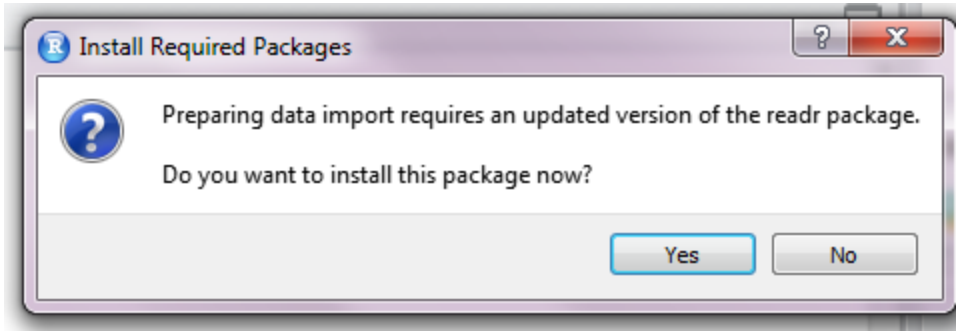
```
> setwd("C:/Users/curley1/Dropbox/Work/R/mypackages/domR")
> getwd()
[1] "C:/Users/curley1/Dropbox/Work/R/mypackages/domR"
>
```

Here's how to make a data file available to users of your package

Example: Load "caribou.csv" into Global Environment

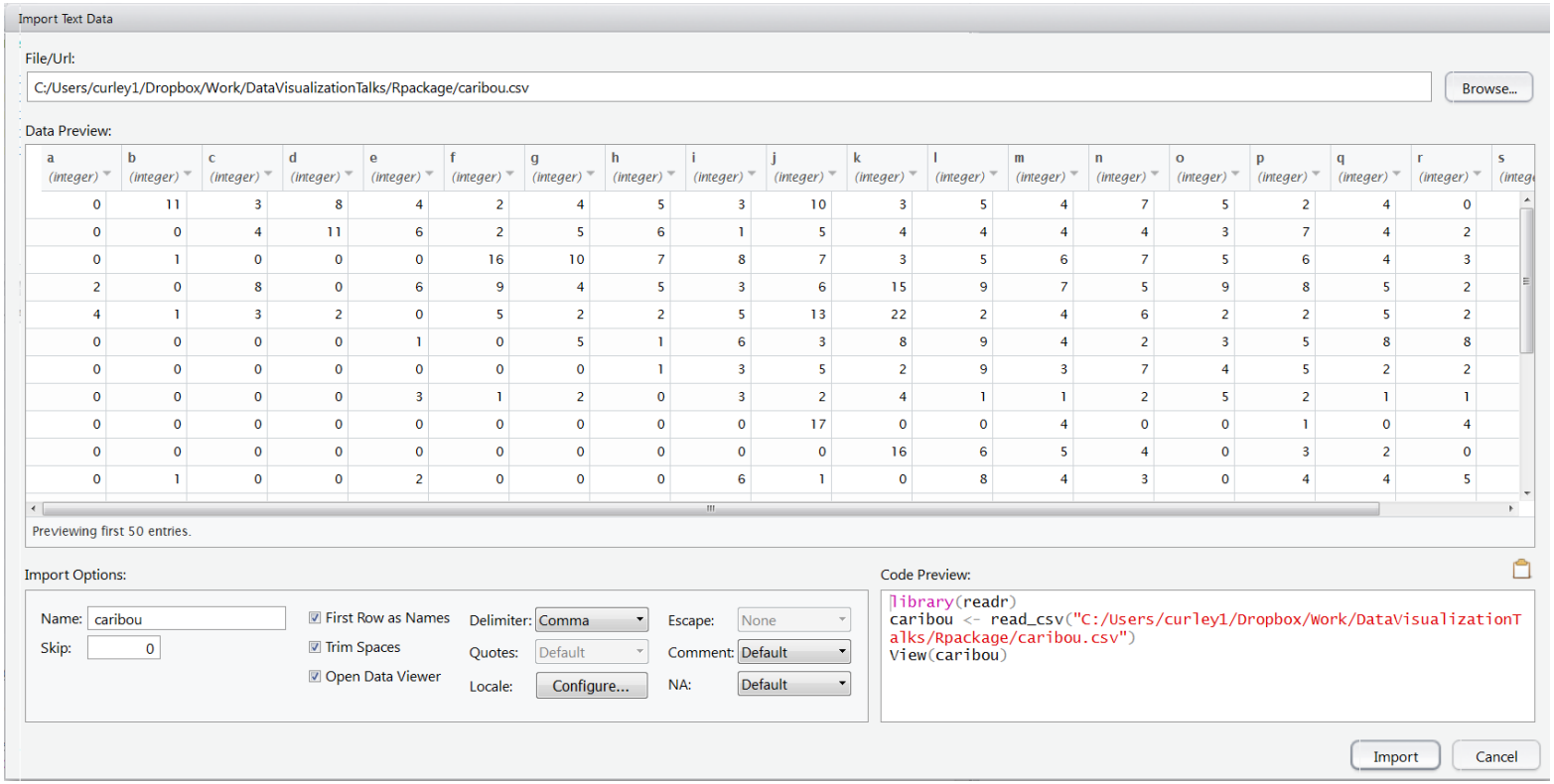


If you're following this tutorial and using "packrat", and you import using the Rstudio dropdown menu you will get this prompt: Hit "YES"

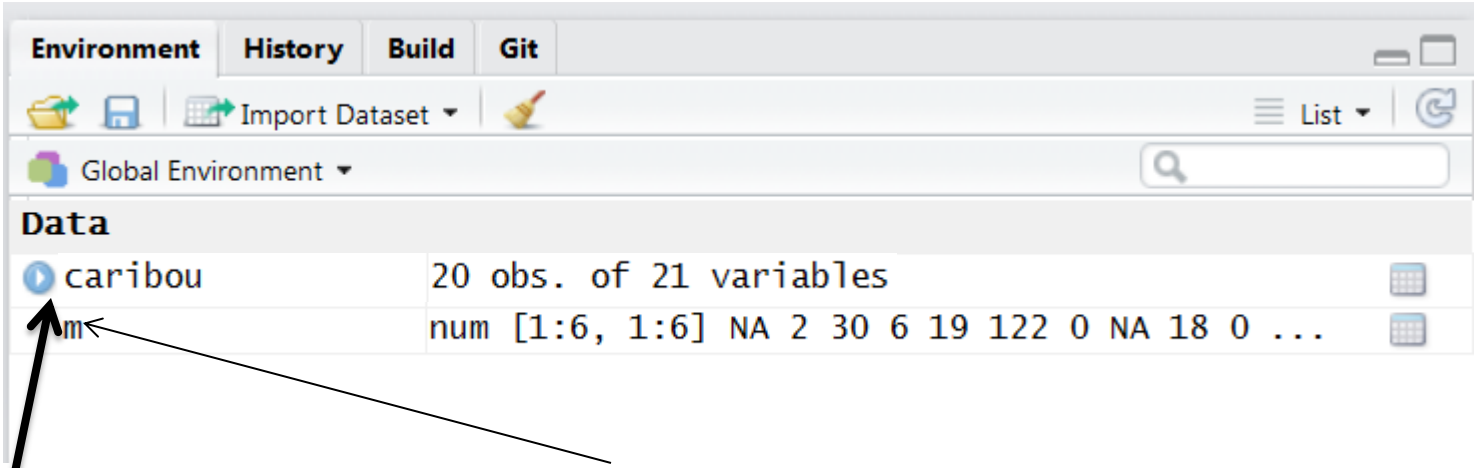


You could also import your data file in any other way to your working directory that you are comfortable with.

Importing …


Import Text Data
File/Url:
C:/Users/curley1/Dropbox/Work/DataVisualizationTalks/Rpackage/caribou.csv
Browse...
Data Preview:
a (integer) b (integer) c (integer) d (integer) e (integer) f (integer) g (integer) h (integer) i (integer) j (integer) k (integer) l (integer) m (integer) n (integer) o (integer) p (integer) q (integer) r (integer) s (integ
0 11 3 8 4 2 4 5 3 10 3 5 4 7 5 2 4 0
0 0 4 11 6 2 5 6 1 5 4 4 4 4 3 7 4 2
0 1 0 0 0 16 10 7 8 7 3 5 6 7 5 6 4 3
2 0 8 0 6 9 4 5 3 6 15 9 7 5 9 8 5 2
4 1 3 2 0 5 2 2 5 13 22 2 4 6 2 2 5 2
0 0 0 0 1 0 5 1 6 3 8 9 4 2 3 5 8 8
0 0 0 0 0 0 0 1 3 5 2 9 3 7 4 5 2 2
0 0 0 0 3 1 2 0 3 2 4 1 1 2 5 2 1 1
0 0 0 0 0 0 0 0 0 17 0 0 4 0 0 1 0 4
0 0 0 0 0 0 0 0 0 0 16 6 5 4 0 3 2 0
0 1 0 0 2 0 0 0 6 1 0 8 4 3 0 4 4 5
Previewing first 50 entries.
Import Options:
Name: caribou
Skip: 0
First Row as Names
Trim Spaces
Open Data Viewer
Delimiter: Comma
Quotes: Default
Locale: Configure...
Escape: None
Comment: Default
NA: Default
Code Preview:
library(readr)
caribou <- read_csv("C:/Users/curley1/Dropbox/Work/DataVisualizationTalks/Rpackage/caribou.csv")
View(caribou)
Import
Cancel

It should be in your Global Environment …



The "m" is the object we used to test our function earlier

The data object should have the name that you want it to be called in your package.

hello.R | dci.R | data_processing.R | DESCRIPTION | NAMESPACE | caribou.R | bonobos.R | get_di_matrix.R

Source on Save | Run | Source

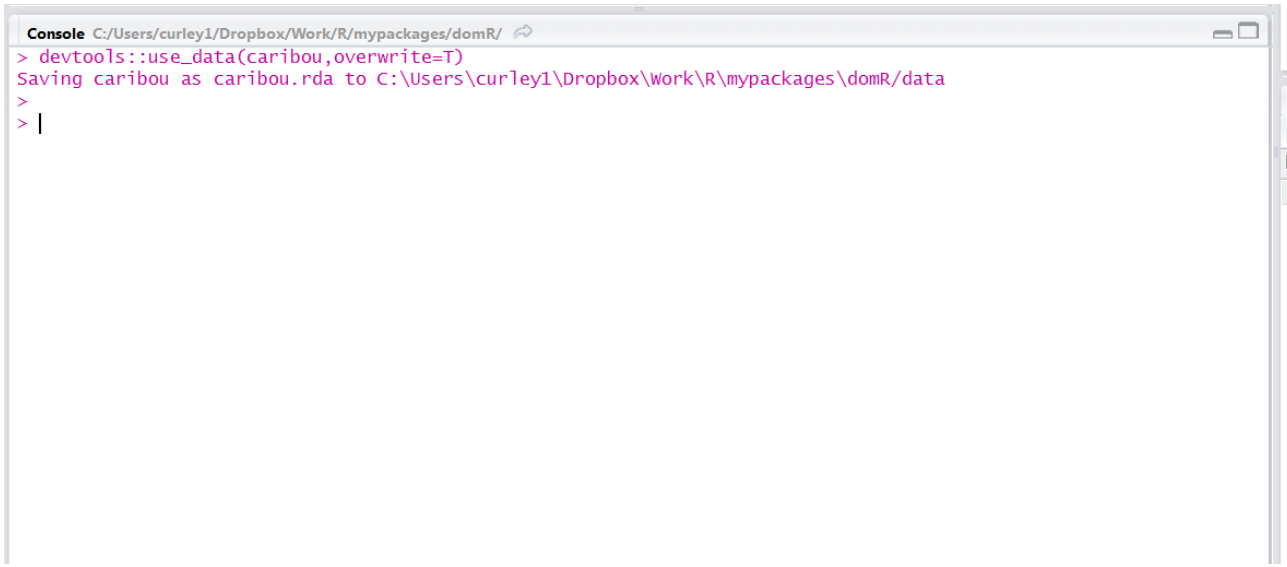
```
1
2  #Script to turn raw csv into a object for Package
3
4  bonobos <- readr::read_csv("C:/Users/curley1/Dropbox/Work/DataVisualizationTalks/Rpackage/bonobos.csv")
5  caribou <- readr::read_csv("C:/Users/curley1/Dropbox/Work/DataVisualizationTalks/Rpackage/caribou.csv")
6
7
8  bonobos <- bonobos[,-1]
9  caribou <- caribou[,-1]
10
11 rownames(bonobos)<-colnames(bonobos)
12 rownames(caribou)<-colnames(caribou)
13
14 write.csv(bonobos, "data-raw/bonobos.csv", row.names=F)
15 write.csv(caribou, "data-raw/caribou.csv", row.names=F)
16
```

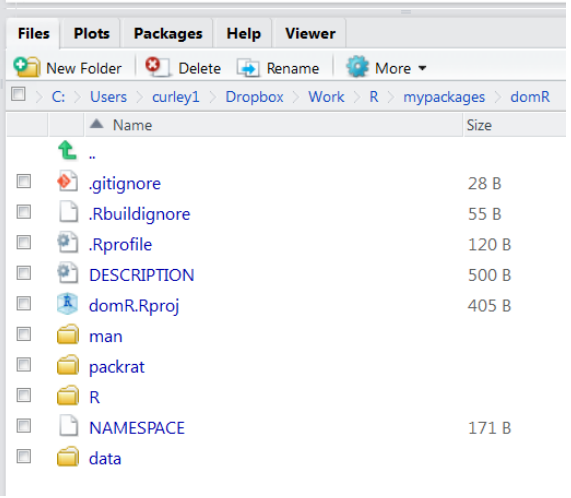
Creating our first data object

devtools::use_data()

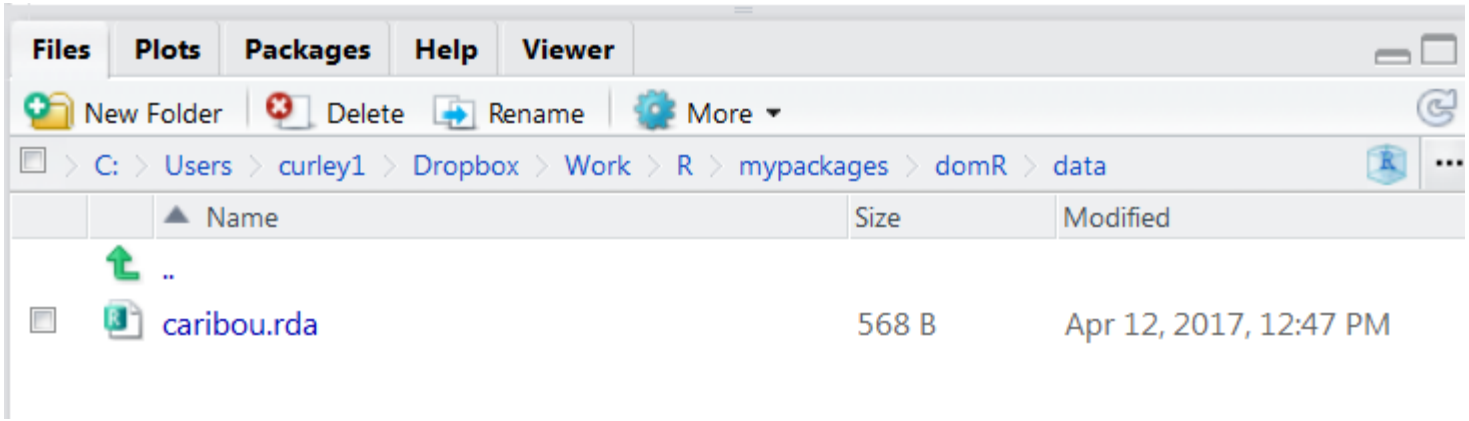
1. Will create a “DATA” folder
2. Will save the object as a “.rda” file in the DATA folder
3. Use “overwrite=T” to replace an existing file of same name (or just add file if no file of that name)

devtools::use_data(caribou,overwrite=T)





The file should now be in the “DATA” folder:



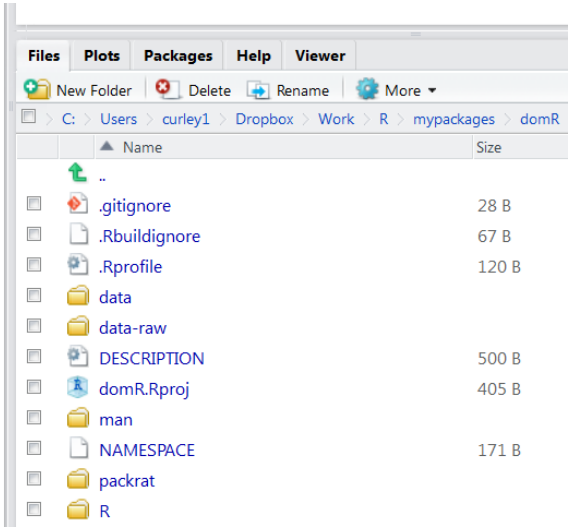
What about raw data ?

devtools::use_data-raw()

- best to keep this and any .R files used in processing data

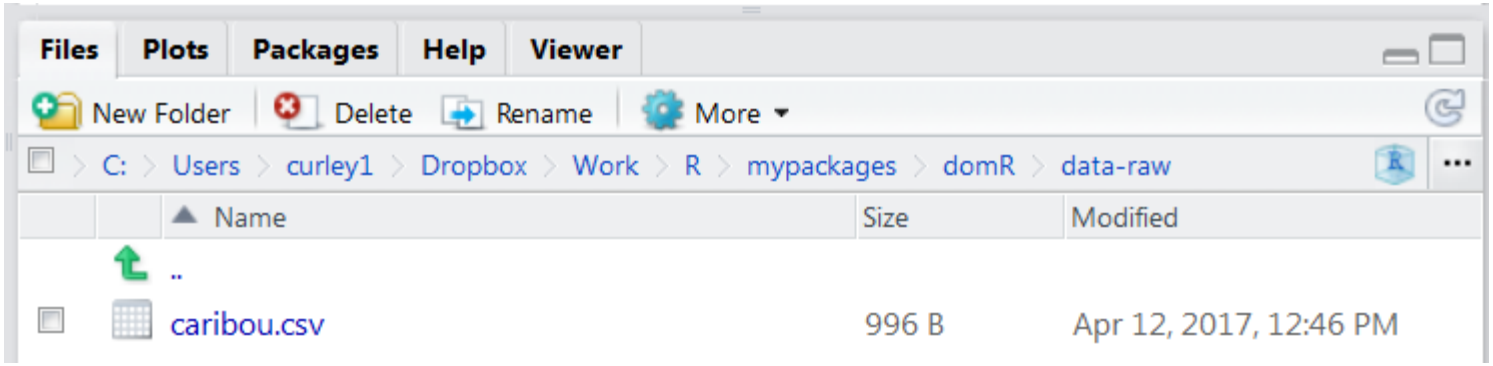
- to enable anybody to go from raw data to final outcome, should be able to determine how data was processed

```
> devtools::use_data(caribou,overwrite=T)
Saving caribou as caribou.rda to C:\Users\curley1\Dropbox\Work\R\mypackages\domR/data
>
> devtools::use_data_raw()
* Creating `data-raw`.
* Adding `data-raw` to `.Rbuildignore`.
Next:
* Add data creation scripts in data-raw
* Use devtools::use_data() to add data to package
>
>
```

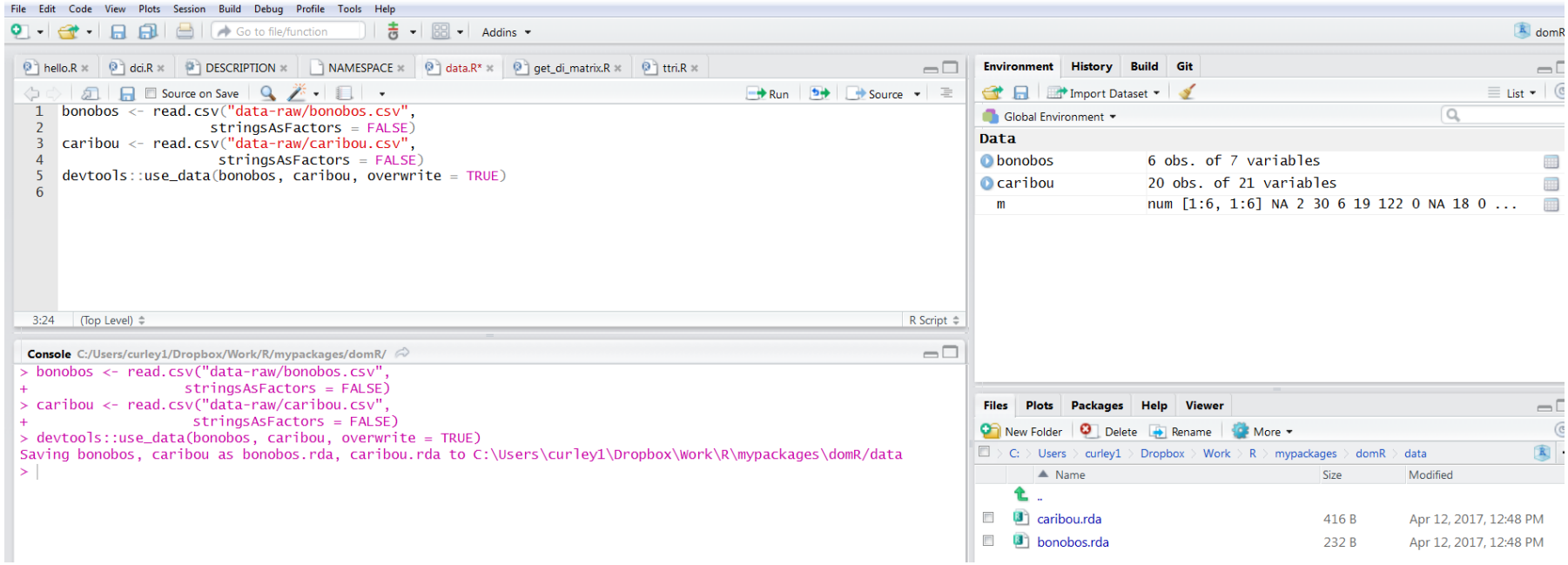


Files that you want to save into “data-raw” can be saved in any format:

```
>
> write.csv(caribou, "data-raw/caribou.csv", row.names=F)
>
```



Can do for multiple datasets



Here, using a file called “data.R” that I will save in “data-raw” that contains the code to load files and save as “.rda” files in “data” folder.

I also added “data_processing.R” into “data-raw” folder

Add .R files to R folder for datasets

hello.R | dci.R | DESCRIPTION | NAMESPACE | data.R | caribou.R | bonobos.R | get_di_matrix.R | ttri.R

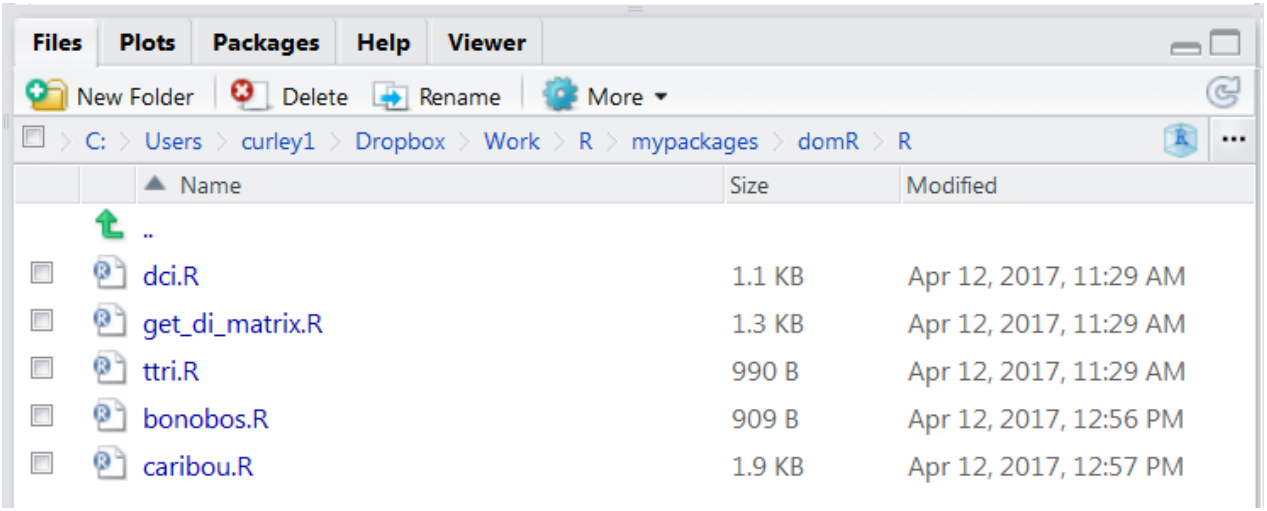
Source on Save | Run | Source

```
#' Bonobo sociomatrix
#'
#' A win-loss sociomatrix with each row being the number of wins
#' against individual bonobos in each column.
#'
#' Reference: Vervaecke, H., de Vries, H. & van Elsacker, L. 2000.
#' Dominance and its behavioral measures in a captive group of
#' bonobos (Pan paniscus). International Journal of Primatology,
#' 21, 47-68.
#'
#' @format A data frame with 6 rows and 6 variables:
#' \describe{
#'   \item{Dz}{The number of losses by bonobo Dz against all other bonobos}
#'   \item{He}{The number of losses by bonobo He against all other bonobos}
#'   \item{De}{The number of losses by bonobo De against all other bonobos}
#'   \item{Ho}{The number of losses by bonobo Ho against all other bonobos}
#'   \item{Lu}{The number of losses by bonobo Lu against all other bonobos}
#'   \item{Ki}{The number of losses by bonobo Ki against all other bonobos}
#' }
"bonobos"

```

Remember:

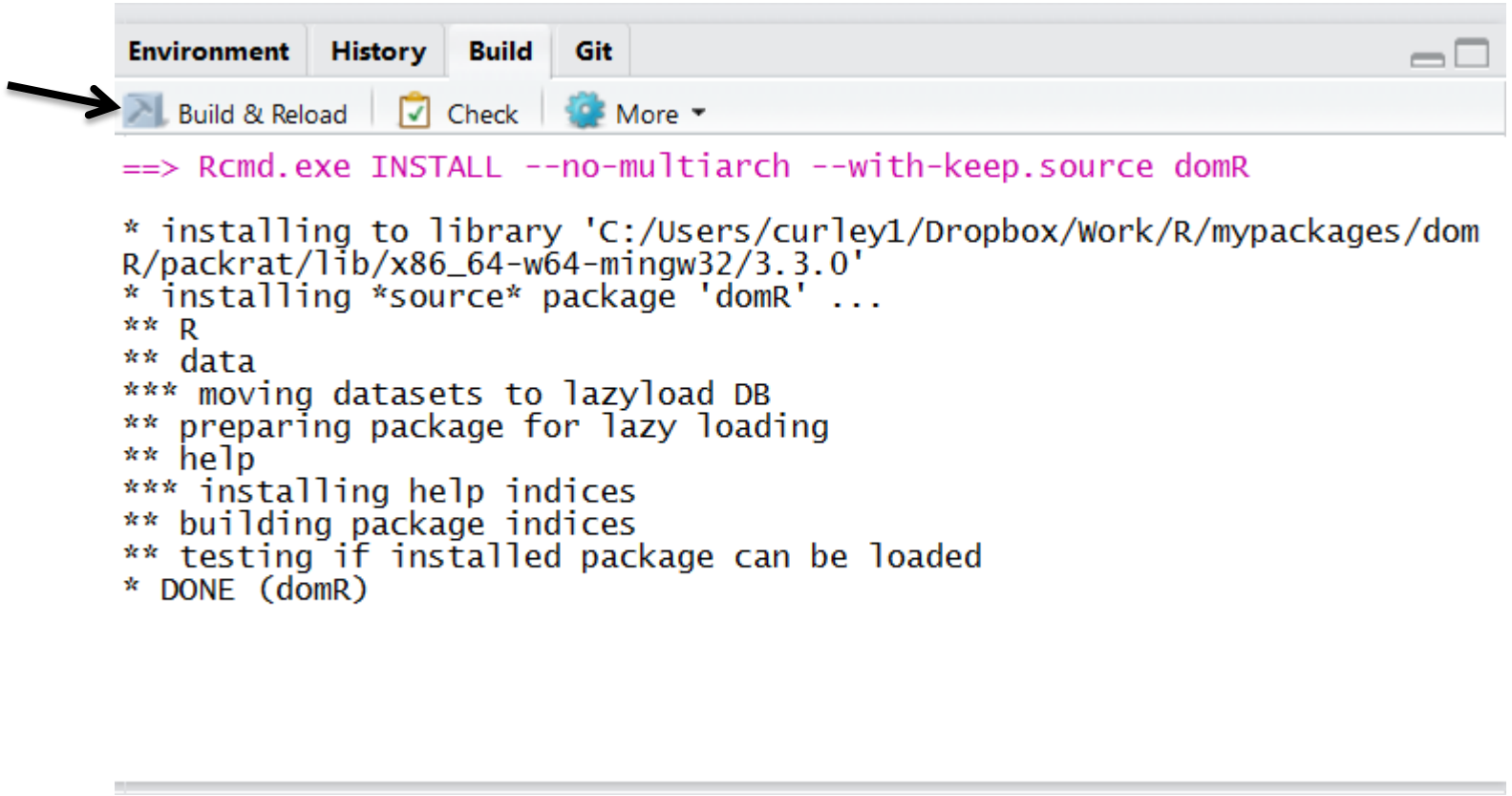
```
devtools::document()
```

Add .R files to R folder for datasets



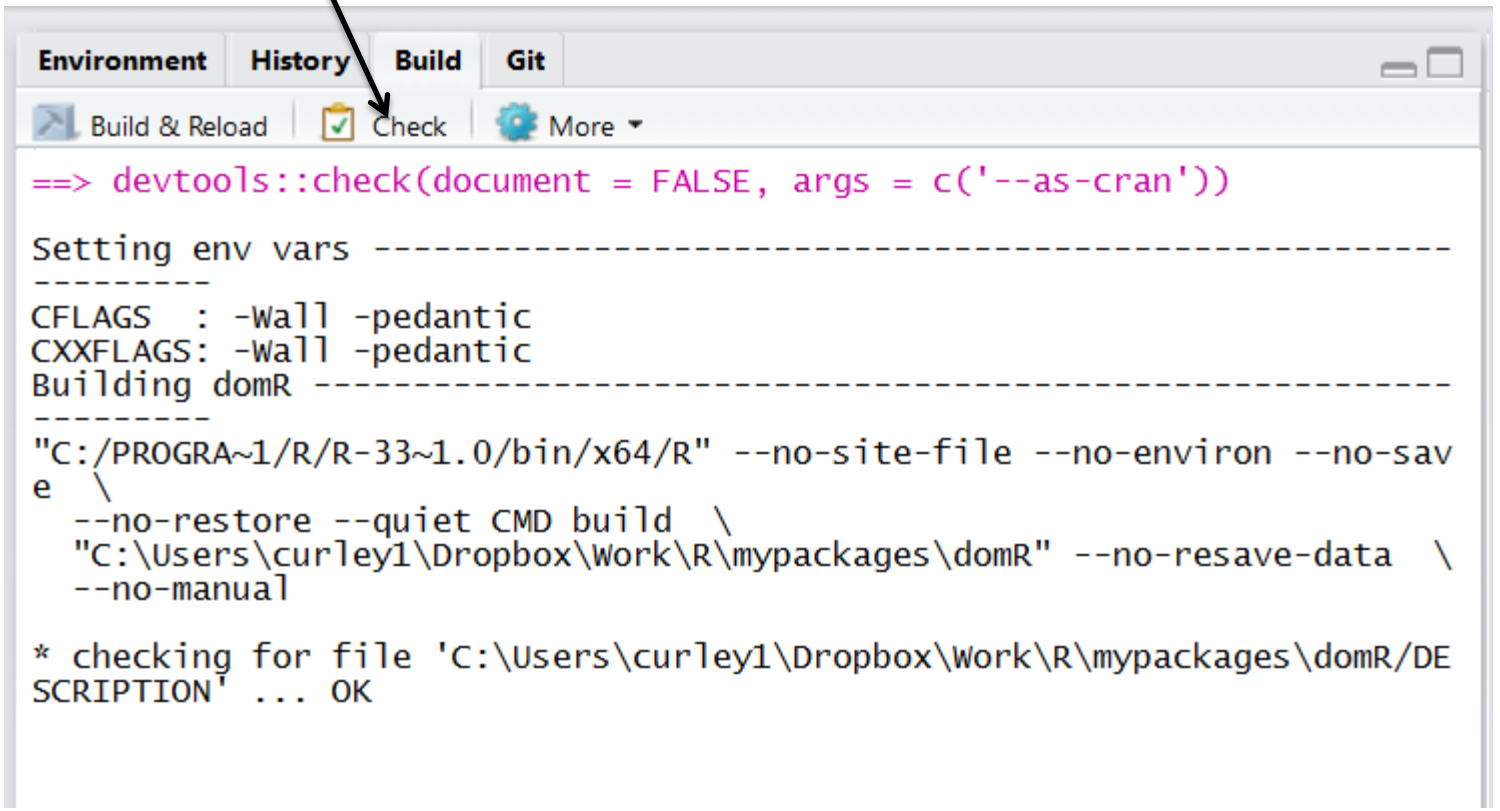
These .R files are needed when you run Checks.

It’s always ok to rebuild



Running a CRAN check

It starts by hitting this button



Running a CRAN check

It then goes like this:

Running a CRAN check

The results will have at the bottom:

ERRORS - you have to fix these immediately !

WARNINGS - you really should fix these immediately !

NOTES - you could not bother – but you should.

The point of this check is to ensure that your package will work consistently.

It can be hard to work out what the ERRORS/WARNINGS/NOTES are referring to, but it's usually some error in the code or documentation that you've written. Persevere with it – Google it !!!

It can help to run checks after every new thing you add to the package – that way you know what went wrong immediately.

```
* DONE
Status: OK


R CMD check results
0 errors | 0 warnings | 0 notes

R CMD check succeeded
```

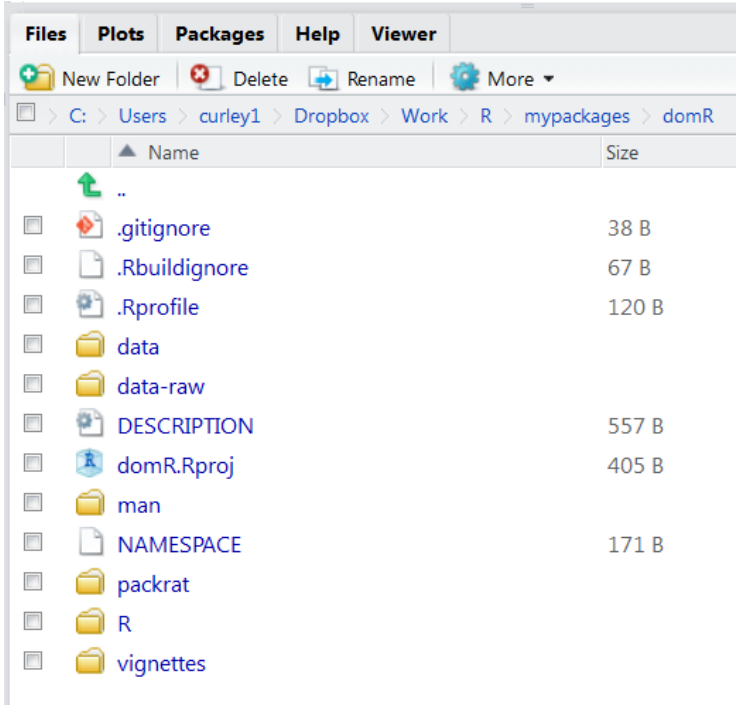
Adding vignettes

devtools::use_vignette()

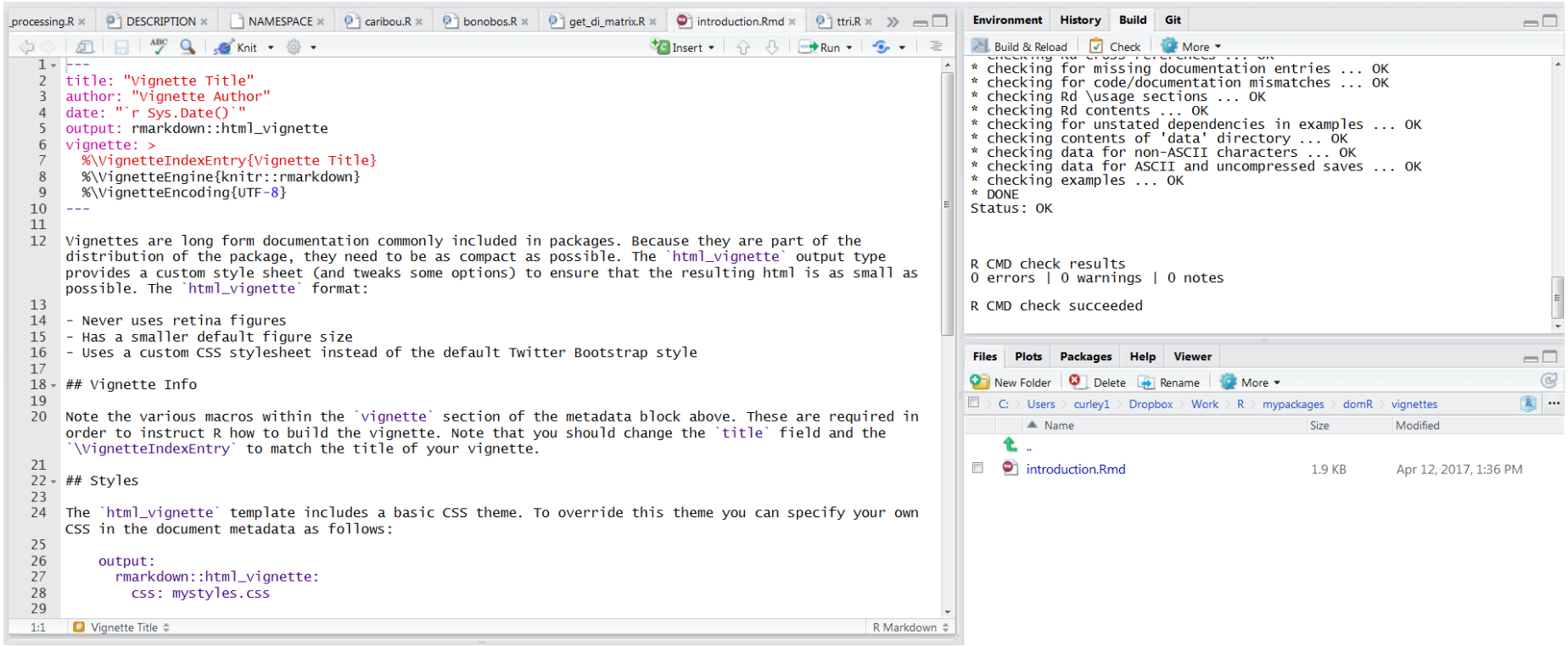
For other colleagues, for peer-reviewers, for other users, for yourself !!!

```
Console C:/Users/curley1/Dropbox/Work/R/mypackages/domR/
> devtools::use_vignette("introduction")
'rmarkdown' must be installed for this functionality.
Would you like to install it?

1: Yes
2: No

Selection:
Enter an item from the menu, or 0 to exit
Selection: 1
```

```
The downloaded binary packages are in
        C:\Users\curley1\AppData\Local\Temp\RtmpUBykFX\downloaded_packages
* Creating `vignettes`.
* Adding `inst/doc` to ./.gitignore
>
```



You can now edit the vignette….



Install your package into your R system library

```
devtools::install()
```

Your package should now be available on your machine whenever you need it.

Access it like you would any other package:

```
library("domR")
```

Publishing your package - GitHub

1. Create your repository on GitHub

2. Add the contents of your entire folder

3. Install/share from GitHub directly
   using the following:

devtools::install_github("jalapic/domR")

Publishing your package - CRAN

1. devtools::release()
2. Follow ALL of the instructions as they appear in the console
3. Add a file called “cran-comments.md” to the main folder.
4. This file should contain information based on the CRAN checks (see example).
5. Add this file to .Rbuildignore

See here: http://r-pkgs.had.co.nz/release.html

Things to cover in version 2 of this tutorial:

- Writing tests
- Writing a package website using “pkgdown”
- Further ‘how-to’ on CRAN submission
- Adding other data types e.g. internal only data into a package.