PROJECT

DEVELOPING R PACKAGES

BY GENEVIEVE STEIN-O'BRIEN

DATE

MAY 22, 2019

CLIENT

R LADIES WORKSHOP

Resources for building your own R package (or a presentation on it)

- Hadley Wickham's eText: *R packages*: http://r-pkgs.had.co.nz/
- Hillary Parker's Blog: https://hilaryparker.com/2014/04/29/writing-an-r-package-from-scratch/
- John Muschelli's Blog: https://hopstat.wordpress.com/2014/05/06/smart-hackathon-day-2-writing-packages-in-rstudio/
- Stat545 tutorial: http://stat545.com/packages00_index.html
- Jeff Leek's tutorial: https://github.com/jtleek/rpackages/blob/master/README.md
- Rstudio cheatsheet: https://www.rstudio.com/wp-content/uploads/2015/03/devtools-cheatsheet.pdf
- The infinite resource that is the google results for "how to build an R package"

When and why should you build an R package

- You have more than three functions that you use repeatedly across multiple analyses
- You want to participate in reproducible research
- You want to distribute the cool code that you worked so hard on
- You want a hex sticker of your own

The minimal parts of a R package

- A R/ directory which contains your code for your package
- A DESCRIPTION file which stores important metadata about your package, i.e. version number, author, dependencies, license information, etc.
- A NAMESPACE file which contains imports, exports, and search path.
- For most people the DESCRIPTION and NAMESPACE will be automatically generated by devtools, Roxygen, or Rstudio.

A little more description about DESCRIPTION

- Example minimal DESCRIPTION file from the Rstudio cheatsheet

```
Package: mypackage
Title: Title of Package
Version: 0.1.0
Authors@R: person("Hadley", "Wickham", email =
    "hadley@me.com", role = c("aut", "cre"))
Description: What the package does (one paragraph)
Depends: R (>= 3.1.0)
License: GPL-2
LazyData: true
Imports:
    dplyr (>= 0.4.0),
    ggvis (>= 0.2)
Suggests:
    knitr (>= 0.1.0)
```

Import packages that your package *must have* to work. R will install them when it installs your package.

Suggest packages that are not very essential to yours. Users can install them manually, or not, as they like.

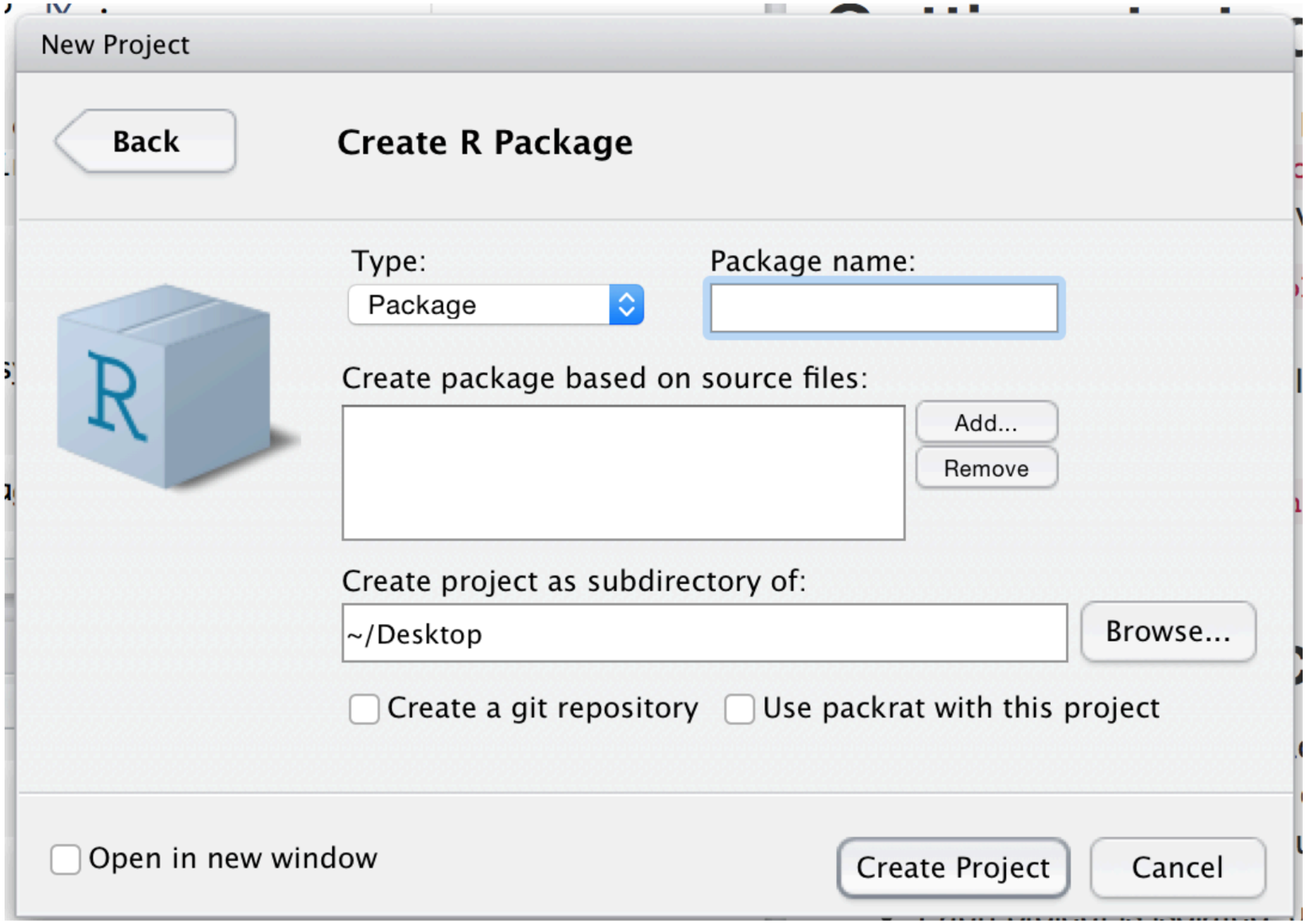
One line package generation

From within R run:

```
devtools::create("path/to/package/pkgname")
```

Using Rstudio to automate much of package building

- Click File | New project
- Choose “New Directory”:
- Then Project type: R package
- Then give your package a name and click “Create Project”:



Why you should use github for your package

- To err is human
- You want to participate in reproducible research
- You want to distribute the cool code that you worked so hard on
- You want a hex sticker of your own
- You want to collaborate on your code with others

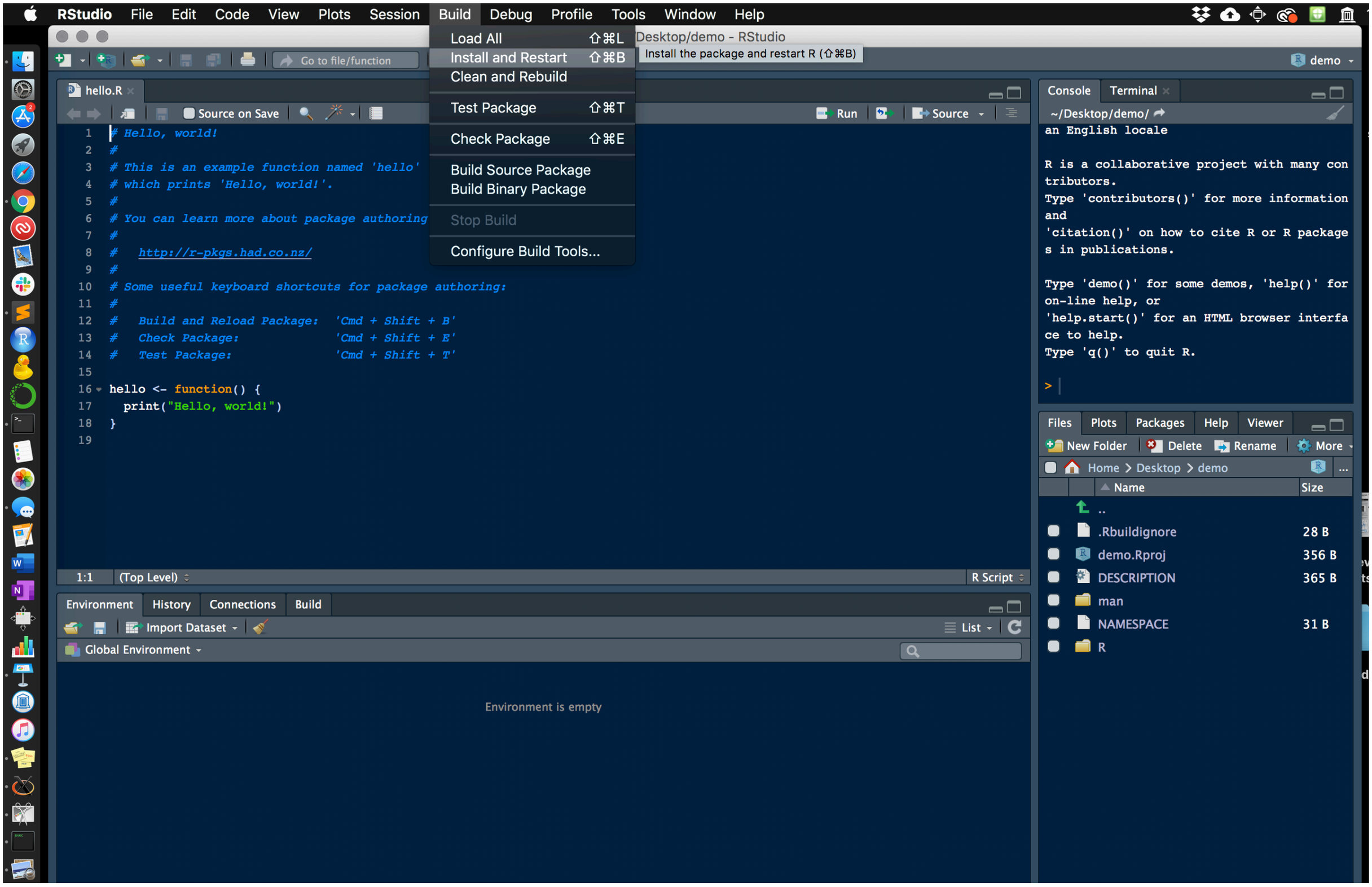
The basics of using github for version control

- Create a new Github repository with the same name (pkgname)
- In the pkgname directory on your local machine, run the commands:
  - git init
  - git remote add origin git@github.com:yourusername/packagename.git
  - Create a file in the pkgname directory called README.md. This will be displayed on the homepage of repo.
  - git add *
  - git commit -m ‘initial commit’
  - git push -u origin master
- Repeat last three steps every time you change something substantial subbing out ‘initial commit’ with ‘description of the change’

Building, checking, and installing your package

- On the command line within the pkgname directory run
  - R CMD build package name
  - R CMD check pkgname*.tar.gz
  - R CMD INSTALL packagename
- Within R using devtools (i.e. install.packages(devtools))
  - library(devtools)
  - build( )
  - check( )
  - install( )

Building and installing your package within Rstudio



Testing your package build and code

- Unit test are individual tests that help debug code and check compatibility
- The testthat package can be used to design unit tests that will run on each build or R CMD check of the package and throw an error if any of the tests fail.
- To use the testthat package you will put a file called *test-area-packagename.R* in the inst/tests directory. Then add another file called *run-all.R* to a separate directory called tests. The *run-all.R* function has this code in it:
  - library(testthat)
  - library(pkgname)
  - test_package("pkgname")

If you are only building packages for your own personal use and have excellent memory the previous slides contain pretty much everything you need.

For the rest of us, documentation is highly recommended.

Generating documentation with Roxygen

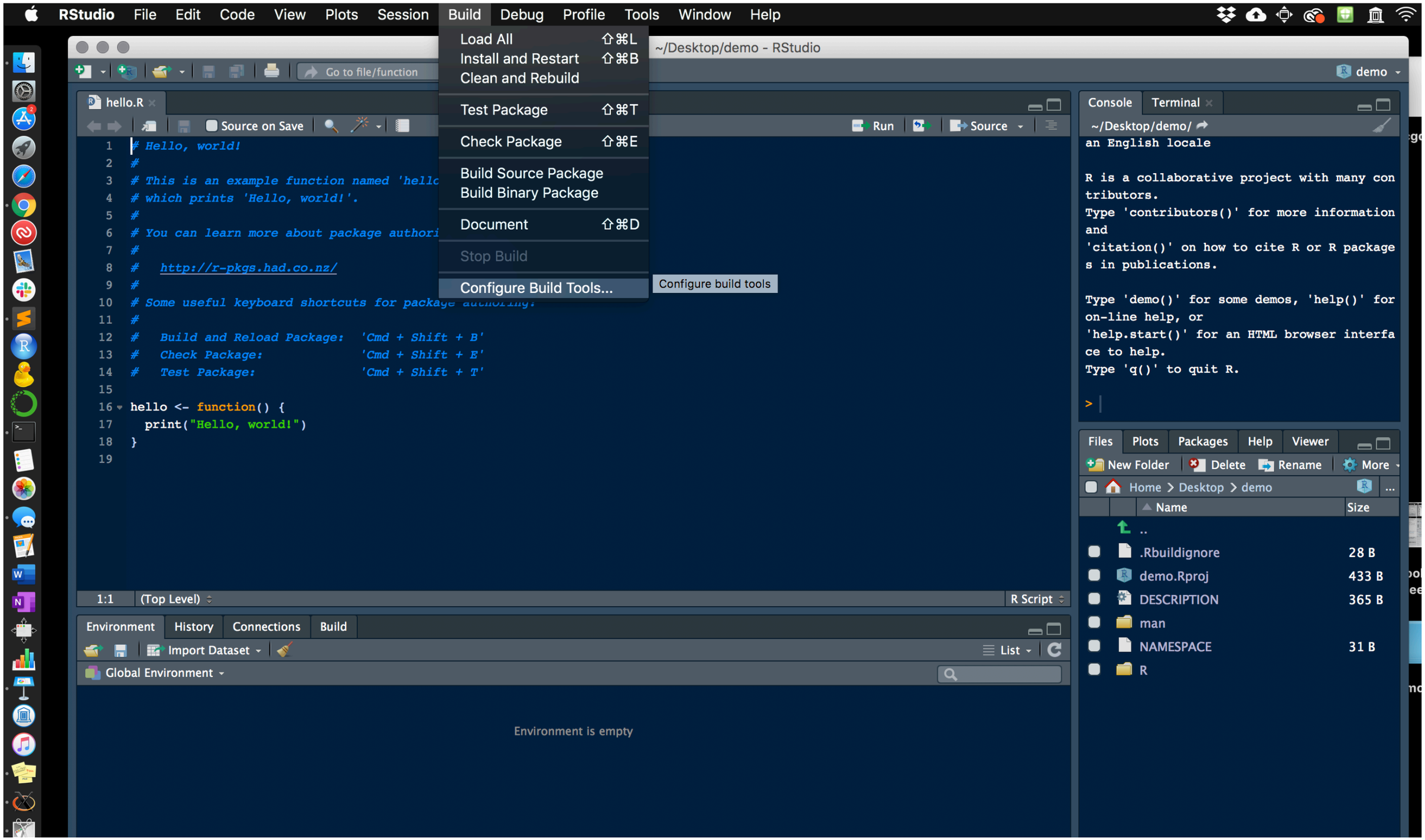
- Roxygen automatically generates man files (the info that you get when you type ? before a function name in R) from a specially formatted header above the function

- An example minimal function with Roxygen header from the Rstudio cheatsheet:

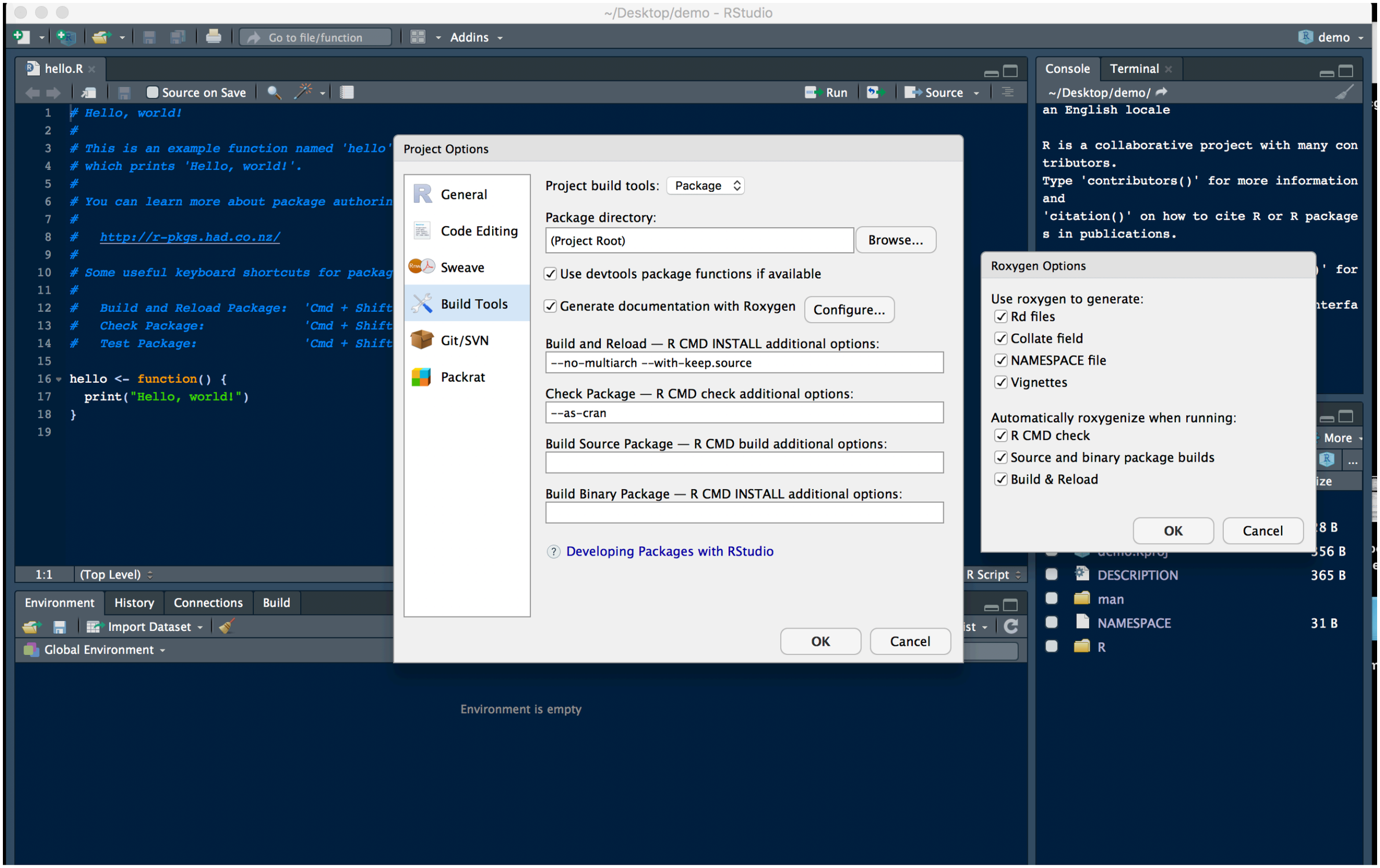
```
#' Add together two numbers.
#'
#' @param x A number.
#' @param y A number.
#' @return The sum of \code{x} and \code{y}.
#' @examples
#' add(1, 1)
#' @export
add <- function(x, y) {
  x + y
}
```

- To build Roxygen documentation from within R
  - devtools::document()

Automating documentation within Rstudio



Automating documentation within Rstudio



Vignettes - a must if you want anyone else to use your package

- The minimum that a vignette must contain according to Jeff Leek:
  - A short introduction that explain:
    - The type of data the package can be used on
    - The general purpose of the functions in the packageOne or more example analyses with
  - One or more example analyses with
    - A small, real data set
    - An explanation of the key functions
    - An application of these functions to the data
    - A description of the output and how it can be used

Building for bioconductor

- Read the requirements first!
  - There are some surprisingly specific stylistic ones, i.e. each line can be no more than 80 columns.
- Use the KISS method and keep things general.
- Don’t try an anticipate ever user need or use case.
- Don’t be afraid to ask for help. Submission to bioconducor is a dialog and they are the best people to tell you how to meet their standards.
- Get a travis to test your package: https://docs.travis-ci.com/user/tutorial/

Questions?