Making an R package

Hadley Wickham

Assistant Professor / Dobelman Family Junior Chair

Department of Statistics / Rice University

June 2011

1. Introductions and outline
2. Getting started
3. Development cycle
4. Documentation
5. What next

Introduction

HELLO

my name is

Hadley

Charlotte Wickham
Assistant Professor
Department of Statistics
Oregon State University

Barret Schloerke
Engineer
Metamarkets

http://had.co.nz/courses/11-masterclass

Day two

- Introduction to packages
- Documentation
- Testing
- Releasing your package

Caveats

- Opinionated advice
- Do as I say, not as I do
- One day is not enough: focus on practicing the basics, and pointers to details.

If you only remember one thing:

Read package sources!

Use a mac (or linux)

Otherwise, start by downloading and installing http://www.murdoch-sutherland.com/Rtools/

No matter what, you'll need to learn the command line

Windows

Start -> Control Panel -> System -> Advanced

Click on Environment Variables, under System Variables, find PATH. Make sure the following directories are in the path:

c:\Rtools\bin;c:\Rtools\perl\bin;c:\Rtools\MinGW\bin;c:\Rtools\MinGW64\bin;C:\Program Files\R\R-2.12.0\bin;

To open **command line**: Run file + cmd

Getting started

1. Decide on a name (`stringr`)
2. Create `stringr/` and `stringr/R/` directories
3. Copy in your existing code

```
├── R
|   ├── file1.r
|   ├── file2.r
```

4. Add a description file

```
├── DESCRIPTION
├── R
```

```
Package: stringr
Type: Package
Title: Make it easier to work with strings.
Version: 0.5
Author: Hadley Wickham <h.wickham@gmail.com>
Maintainer: Hadley Wickham <h.wickham@gmail.com>
Description: stringr is a set of simple wrappers that make R's string
    functions more consistent, simpler and easier to use.  It does this
    by ensuring that: function and argument names (and positions) are
    consistent, all functions deal with NA's and zero length character
    appropriately, and the output data structures from each function
    matches the input data structures of other functions.
Imports: plyr
Depends: R (>= 2.11.0)
Suggests: testthat (>= 0.3)
License: GPL-2
```

https://github.com/hadley/devtools/wiki/Package-basics

4. Document your files

5. Run `roxygenize()`

```
├── DESCRIPTION
├── man
├── R
```

```r
#' The length of a string (in characters).
#'
#' @param string input character vector
#' @return numeric vector giving number of characters in
#'   each element of the character vector.  Missing strings have
#'   missing length.
#' @keywords character
#' @seealso \code{\link{nchar}} which this function wraps
#' @export
#' @examples
#' str_length(letters)
#' str_length(c("i", "like", "programming", NA))
str_length <- function(string) {
  string <- check_string(string)

  nc <- nchar(string, allowNA = TRUE)
  is.na(nc) <- is.na(string)
  nc
}
```

https://github.com/hadley/devtools/wiki/docs-function

6. Run `R CMD check`

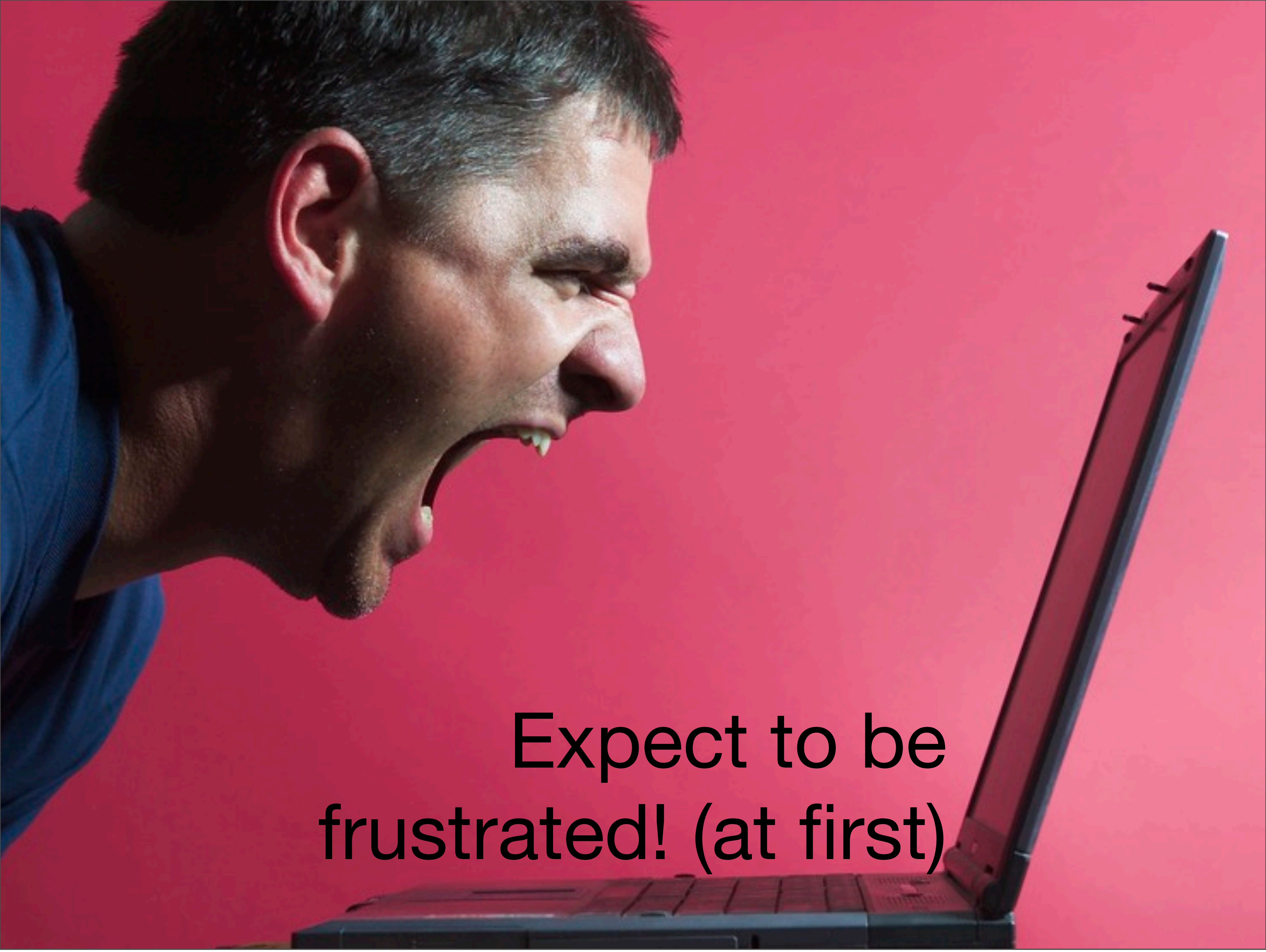
7. Use `R CMD build` to create a package file

8. Upload file to CRAN:
   ```
   ftp -u ftp://cran.R-project.org/incoming/stringr_0.5.tar.gz
   ```

```
├── DESCRIPTION
├── man
├── R
```

https://github.com/hadley/devtools/wiki/Release

You’re done!



Expect to be frustrated! (at first)

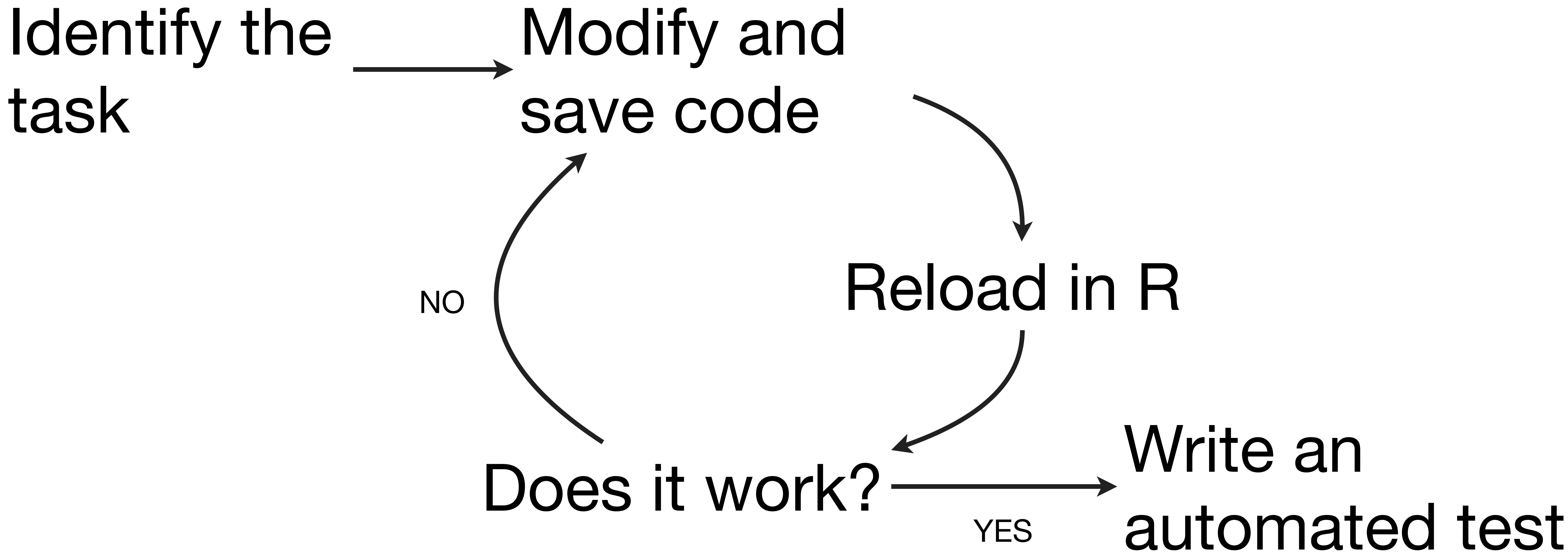
Your turn

Download the source code for the `stringr` and `plyr` from CRAN, or from github.

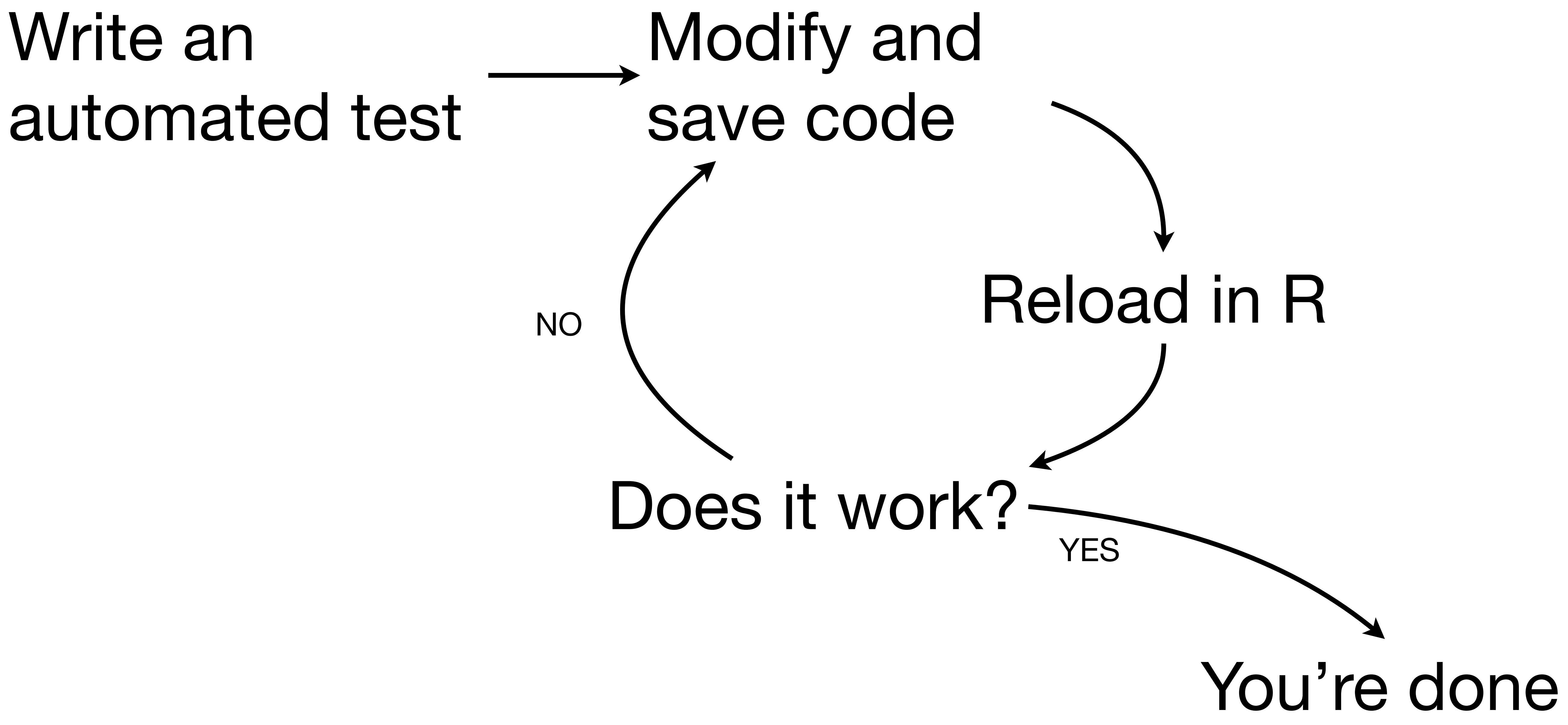
Unzip and explore. What files/directories didn't I mention?

Development cycle

Exploratory programming



Confirmatory programming



aka test driven development (**TDD**)

```
# These patterns are facilitated by the devtools
# package: https://github.com/hadley/devtools

# Once installed and set up,
library(devtools)

# You can easily:
# * Reload code and data
load_all("stringr")

# * Run automated tests
test("stringr")

# * Update documentation
document("stringr")

# My text editor automatically saves all open files when
# I leave it, so I don't even need to explicitly save
```

Your turn

Install the `devtools` package from github and follow the instructions to set it up for your preferred directory structure.

Ensure that it works by testing the `plyr` and `stringr` packages you downloaded.

Installing a package from source

Download and unzip/untar.

```
R CMD install packagename
```

Devtools

- All functions use `as.package()` to find packages - can either be path or package name
- Code useful as reference for in development package (but no tests!)
- Read the source!

Documentation

Documentation

Function-level: gives precise details about individual functions

Package-level: gives details about how to use the functions to solve real (complex) problems.

Both are essential!

Function-level

Keep code and documentation close together with roxygen.

Writing documentation is though but gets (a little) easier with time!

Package-level

Package help topic

NEWS

Vignettes + CITATION

Demos

README

`https://github.com/hadley/devtools/wiki/docs-package`

Next...

Later today

Namespaces

Testing

The release process

Package development best practices

`https://github.com/hadley/devtools/wiki/Package-basics`

Learn from others

Read the source of other packages!

```
https://github.com/hadley/plyr

https://github.com/hadley/stringr

https://github.com/hadley/lubridate

https://github.com/hadley/evaluate

https://github.com/hadley/reshape
```

Key skill

Copying and pasting! (at least when you get started)

Find another package which has done what you want and copy the basic structure.

Your turn

In the `hof-1` directory, you'll find a few functions I'm considering putting into a package for higher order functions.

Start turning these functions into a package by putting them in the appropriatedirectory structure and creating a `DESCRIPTION` file.

Load the code with `load_all(”hof”)` from `devtools`