Packages a**r**e easy!

http://bit.ly/pkgsrez

Hadley Wickham
@hadleywickham
Chief Scientist, RStudio

June 2013

http://bit.ly/pkgsrez

A package is a set of conventions that (with the right tools) makes your life easier

http://bit.ly/pkgsrez

A package is a set of conventions that (with the right tools) makes your life easier

http://bit.ly/pkgsrez

```
# The right tools
#
# The latest version of R (3.0.0)
# RStudio
#
# Code development tools:
#
# * Windows: Rtools, download installer from
#   http://cran.r-project.org/bin/windows/Rtools/
# * OS X: xcode, free from the app store
# * Linux: apt-get install r-base-dev (or similar)
#
# Packages that make your life easier:

install.packages(c("devtools", "knitr", "Rcpp",
  "roxygen2", "testthat"))
```

http://bit.ly/pkgsrez

A package is a set of conventions that (with the right tools) makes your life easier

```
# Live demo!
```

R/

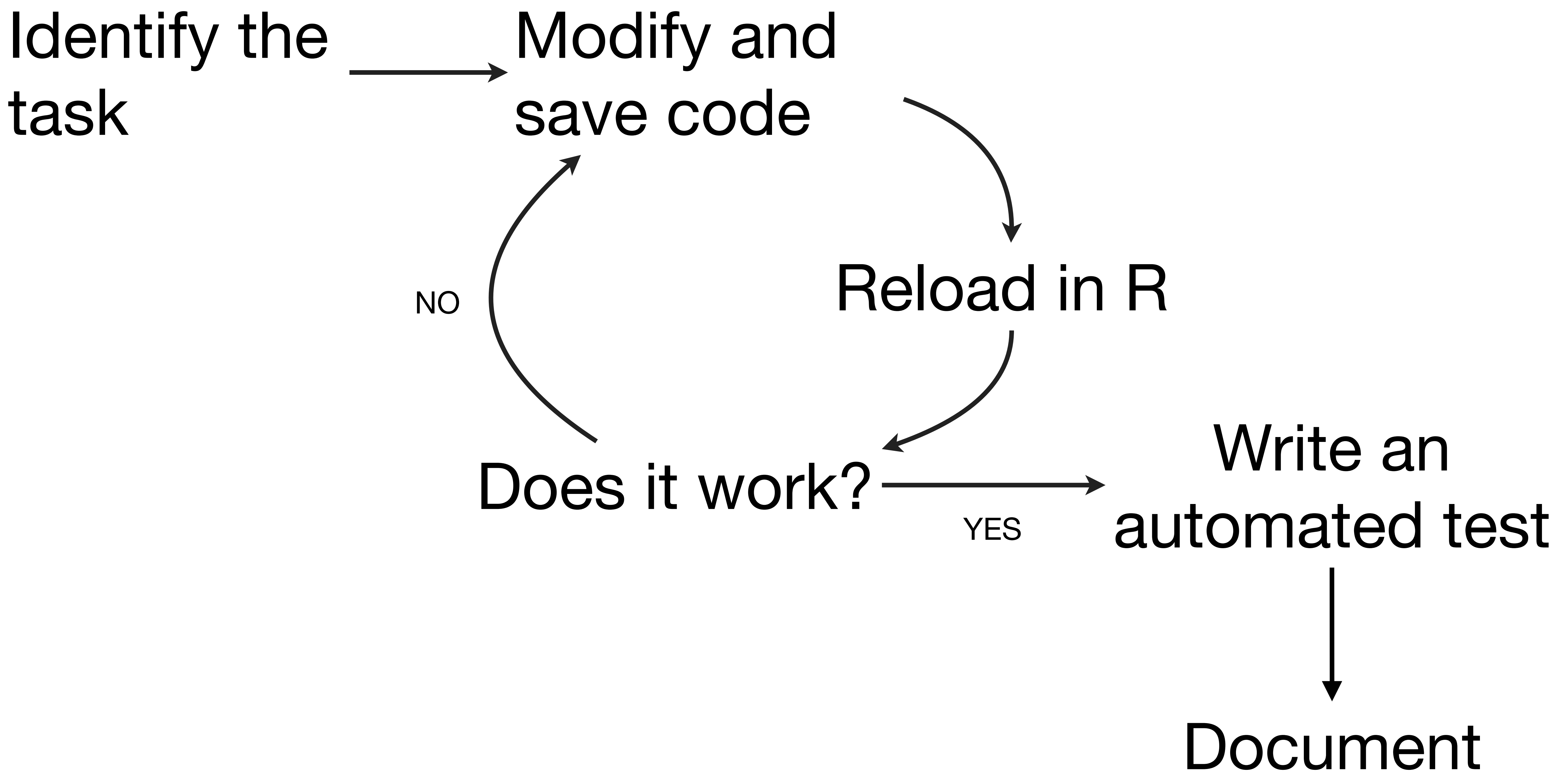
R code

```
library(devtools)
load_all()
# Creates DESCRIPTION (up next)
# Reloads all R code

# NB: All devtools functions take a path to a
# package as the first argument. If not supplied,
# uses the current directory.
```

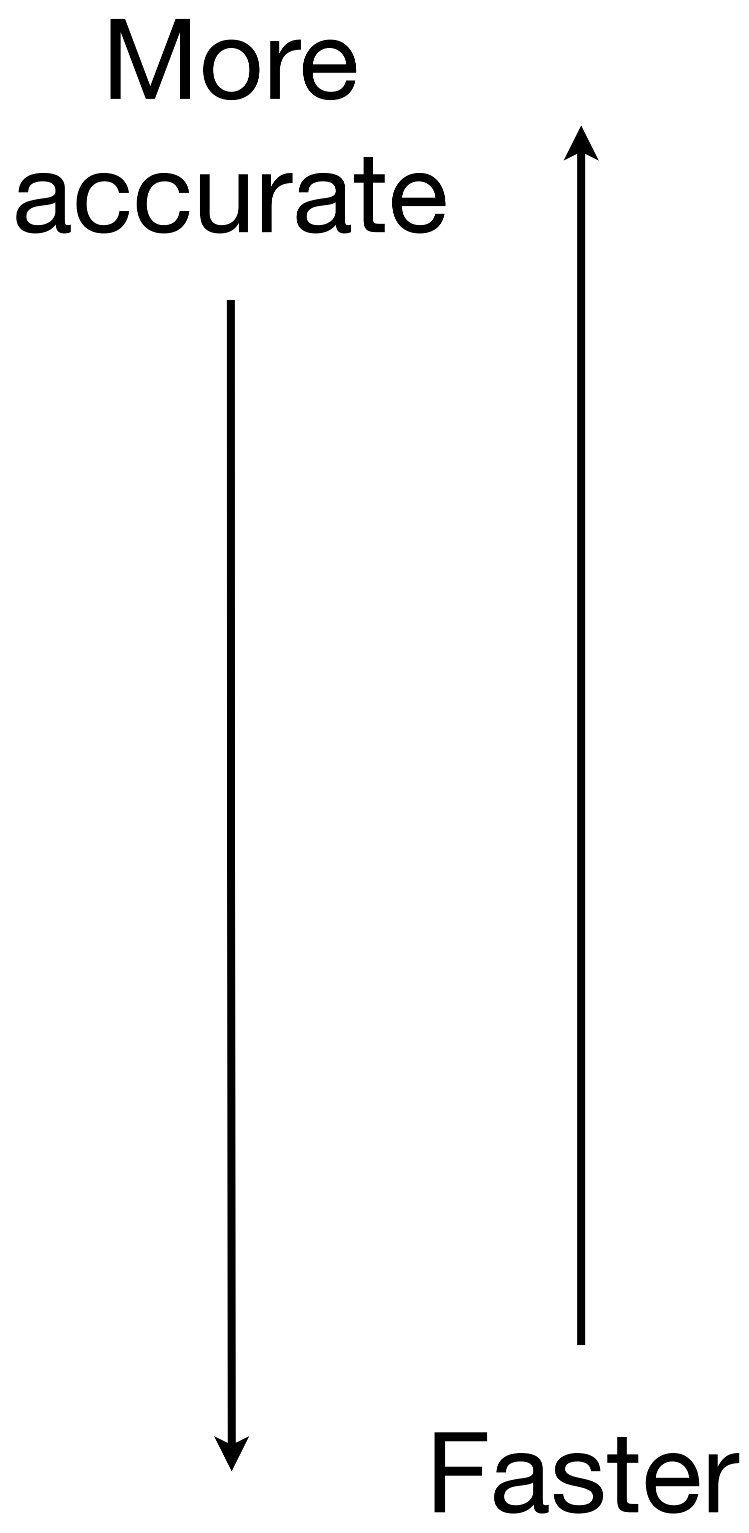
Never use `package.skeleton()`!

Why? It only works once, it does something v. easy to do by hand, and automates a job that needs to be manual

Programming cycle



```
# Reloads changed code, data, ...
load_all(”path/to/package”)

# Reload from scratch
load_all(”path/to/package”, T)

# Installs package and then reloads
install(”path/to/package”)

# Rstudio: Build & Reload
# Installs package, restarts R,
# then reloads
```



DESCRIPTION

Who can use it, what it needs, and who wrote it

```
Package: easy
Version: 0.1
Title:
Description:
Authors@R: 'Hadley Wickham <h.wickham@gmail.com> [aut,cre]'
Depends:
    R (>= 3.0.0)
License: GPL-3
Suggests:
LazyData: true
```

```
Package: easy
Version: 0.1
Title:
Description:
Authors@R: getOption("devtools.desc.author")
Depends:
    R (>= 3.0.0)
License: getOption("devtools.desc.license")
Suggests:
LazyData: true
```

We're done!

That's all you *need* to know about packages

But you can also add data, *documentation*, unit tests, *vignettes* and *C++* code

+

man/

Compiled documentation

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

Roxygen2

- Essential for function level documentation. Huge time saver
- R comments → Rd files → human readable documentation
- `Rd2roxygen` package converts Rd to roxygen if you have legacy packages

Raw R source

```
#' Order a data frame by its columns.
#'
#' This function completes the subsetting, transforming and ordering triad
#' with a function that works in a similar way to \code{\link{subset}} and
#' \code{\link{transform}} but for reordering a data frame by its columns.
#' This saves a lot of typing!
#'
#' @param df data frame to reorder
#' @param ... expressions evaluated in the context of \code{df} and
#'   then fed to \code{\link{order}}
#' @keywords manip
#' @export
#' @examples
#' mtcars[with(mtcars, order(cyl, disp)), ]
#' arrange(mtcars, cyl, disp)
#' arrange(mtcars, cyl, desc(disp))
arrange <- function(df, ...) {
  ord <- eval(substitute(order(...)), df, parent.frame())
  unrowname(df[ord, ])
}
```

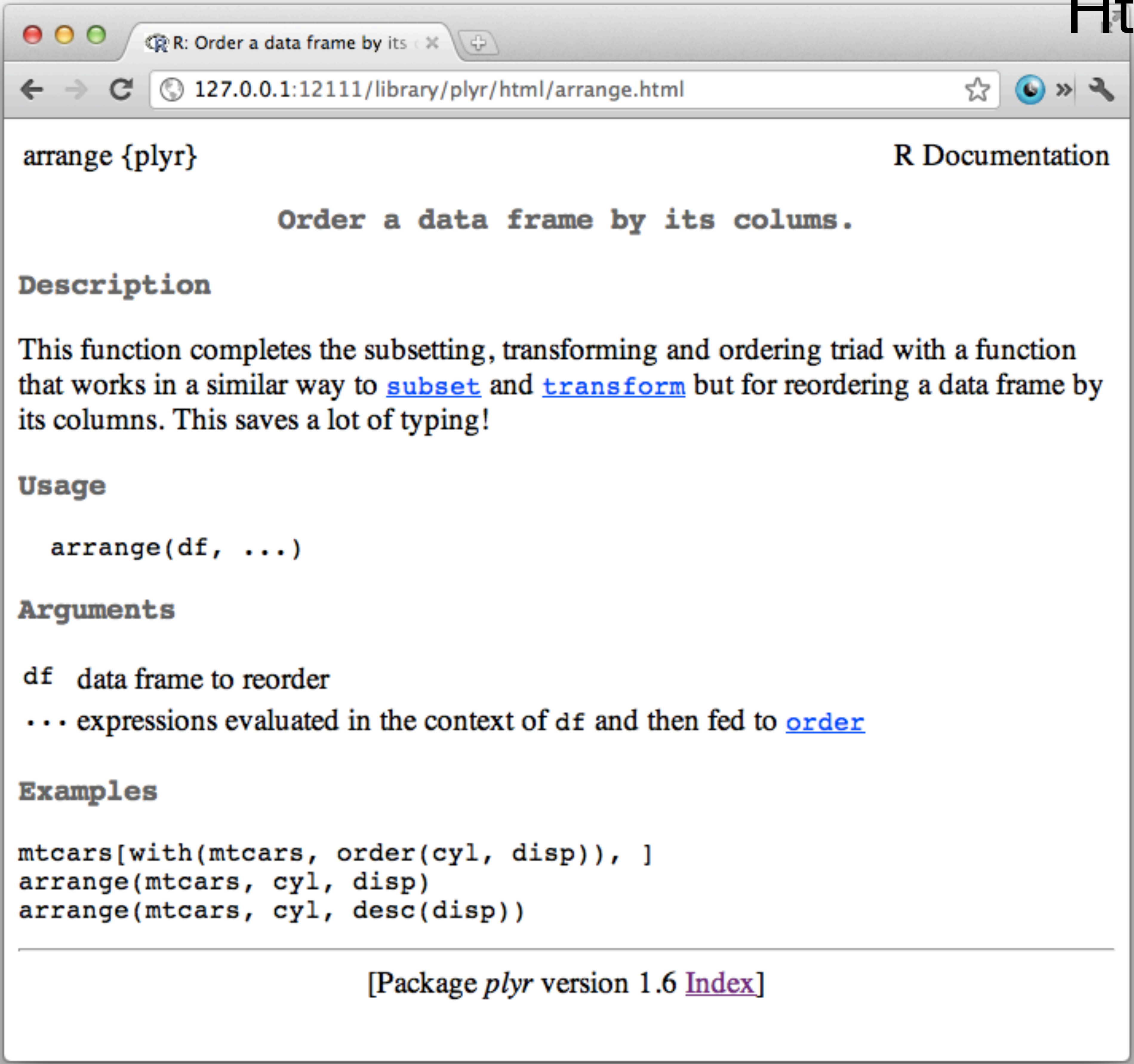
Raw R source

```r
#' Order a data frame by its columns.
#'
#' This function completes the subsetting, transforming and ordering triad
#' with a function that works in a similar way to \code{\link{subset}} and
#' \code{\link{transform}} but for reordering a data frame by its columns.
#' This saves a lot of typing!
#'
#' @param df data frame to reorder
#' @param ... expressions evaluated in the context of \code{df} and
#'   then fed to \code{\link{order}}
#' @keywords manip
#' @export
#' @examples
#' mtcars[with(mtcars, order(cyl, disp)), ]
#' arrange(mtcars, cyl, disp)
#' arrange(mtcars, cyl, desc(disp))
arrange <- function(df, ...) {
  ord <- eval(substitute(order(...)), df, parent.frame())
  unrowname(df[ord, ])
}
```

Generated Rd file

```
\name{arrange}
\alias{arrange}
\title{Order a data frame by its columns.}
\usage{arrange(df, ...)}

\description{
  Order a data frame by its columns.
}

\details{
  This function completes the subsetting, transforming and
  ordering triad with a function that works in a similar
  way to \code{\link{subset}} and \code{\link{transform}}
  but for reordering a data frame by its columns. This
  saves a lot of typing!
}
\keyword{manip}
\arguments{
  \item{df}{data frame to reorder}
  \item{...}{expressions evaluated in the context of \code{df} and then fed
to \code{\link{order}}}
}
\examples{mtcars[with(mtcars, order(cyl, disp)), ]
arrange(mtcars, cyl, disp)
arrange(mtcars, cyl, desc(disp))}
```

Html view in R


R: Order a data frame by its
127.0.0.1:12111/library/plyr/html/arrange.html

arrange {plyr} R Documentation

Order a data frame by its colums.

Description

This function completes the subsetting, transforming and ordering triad with a function that works in a similar way to `subset` and `transform` but for reordering a data frame by its columns. This saves a lot of typing!

Usage

```
  arrange(df, ...)
```

Arguments

`df`	data frame to reorder
`...`	expressions evaluated in the context of `df` and then fed to `order`

Examples

```
mtcars[with(mtcars, order(cyl, disp)), ]
arrange(mtcars, cyl, disp)
arrange(mtcars, cyl, desc(disp))
```

[Package *plyr* version 1.6 Index]

Documentation cycle

1. Update roxygen comments.
2. `document()`
3. `check_doc()`
4. `dev_help(”rdname”)`

+
vignettes/

Long-form documentation

Rmarkdown

Easy to write

Doesn’t need latex toolchain

Only available in 3.0.0

```
# Add to DESCRIPTION
```

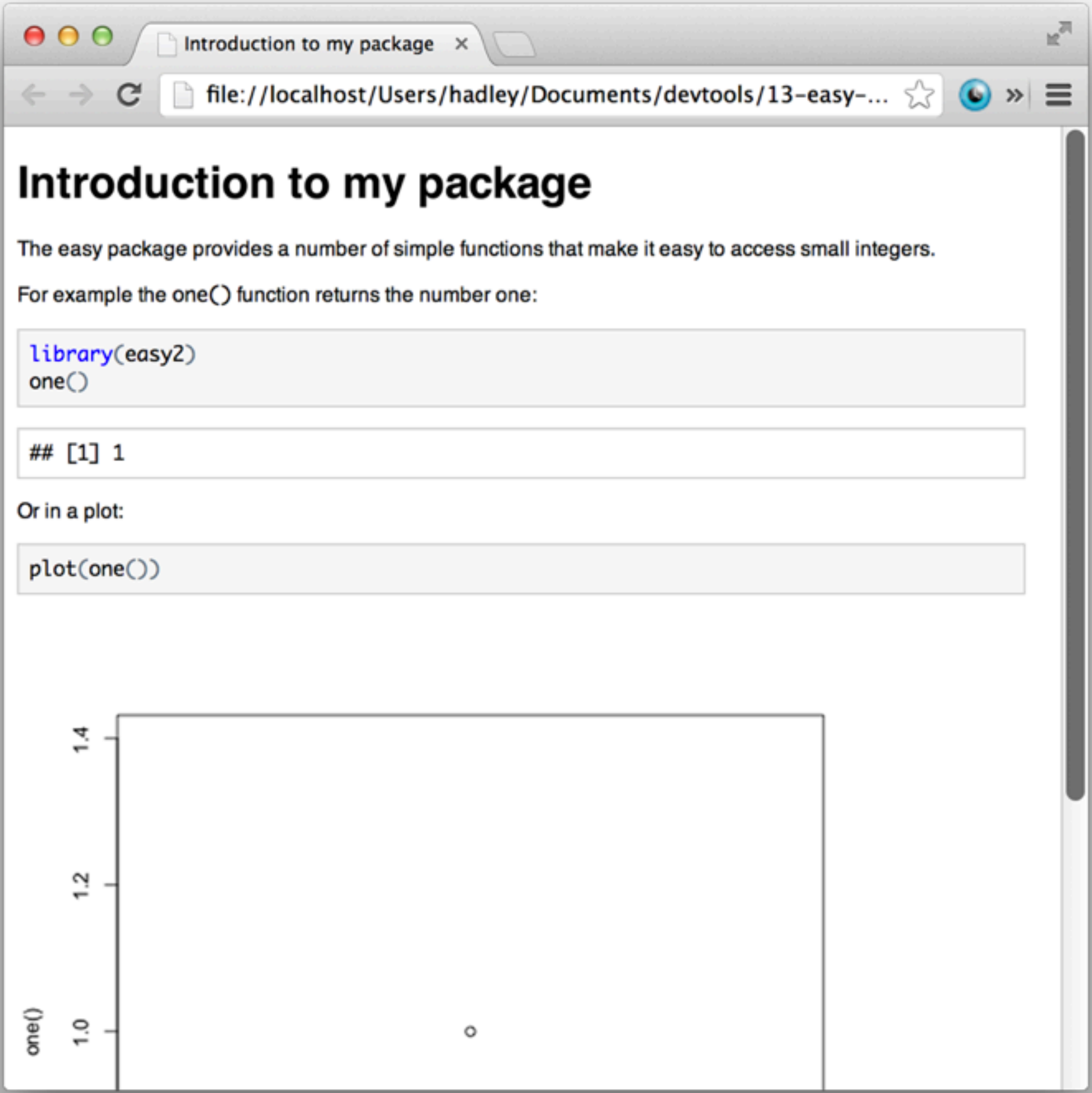
```
VignetteBuilder: knitr
Suggests: knitr
```

```
# In each .Rmd file in vignettes/
```

```
<!--
%\VignetteEngine{knitr}
%\VignetteIndexEntry{Vignette title}
-->
```

~~~
<!--
%\VignetteEngine{knitr}
%\VignetteIndexEntry{Vignette title}
-->

# Introduction to my package

The easy package provides a number of simple functions that make it easy to
access small integers.

For example the `one()` function returns the number one:

```{r}
library(easy2)
one()
```

Or in a plot:

```{r}
plot(one())
```
~~~

```
# Since vignette uses our easy package,
# we need to install it first
install()

build_vignettes()
```


Introduction to my package
file://localhost/Users/hadley/Documents/devtools/13-easy-...

Introduction to my package

The easy package provides a number of simple functions that make it easy to access small integers.

For example the one() function returns the number one:

```
library(easy2)
one()
```

```
## [1] 1
```

Or in a plot:

```
plot(one())
```

+

src/

C++ code

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

# Add to DESCRIPTION

```
Depends: Rcpp (>= 0.10.1)
LinkingTo: Rcpp
```

# Add somewhere in R code:

```
#' @useDynLib mypackage
```

# Add src/Makevars:

```
PKG_LIBS = `$(R_HOME)/bin/Rscript -e
"Rcpp:::LdFlags()"`
```

# And src/Makevars.win:

```
PKG_LIBS = $(shell "${R_HOME}/bin${R_ARCH_BIN}/
Rscript.exe" -e "Rcpp:::LdFlags()")
```

```cpp
#include <Rcpp.h>
using namespace Rcpp;

//' The number two
//'
//' @useDynLib easy
//' @examples
//' one() + two()
// [[Rcpp::export]]
int two() {
  return 2;
}
```

```
document()
load_all()
two()
```

+

● ● ●

More to learn

Other conventions

- `inst/test`: unit tests
- `data`: data files

Learn from others

If you only remember one thing:

Read the source code of other packages.

These are the packages I'm most proud of:

```
https://github.com/hadley/plyr
https://github.com/hadley/stringr
https://github.com/hadley/devtools
https://github.com/hadley/lubridate
https://github.com/hadley/evaluate
https://github.com/hadley/reshape
```

Distribution

- Easiest way: put on github and use `devtools::install_github()` to install
- Most rigorous (and painful): put on CRAN. See `check()` and `release()` for more details

`devtools`

- devtools is constantly improving as I figure out where the pain points are
- Use `install_github("devtools")` to get the latest version
- If something doesn't work for you, please file a bug at `github.com/hadley/devtools/issues`