

Computational Skills for Biostatistics I: Lecture 5

Amy Willis, Biostatistics, UW

November 8, 2017

R packages

Why write an R package?

- ▶ The best way to contribute to science
- ▶ Allow others to build on your advances
- ▶ Promote yourself and your work
- ▶ Pass this class

package = code + other stuff

This class will focus on the other stuff

Install versus load

Brief refresher: `install.packages()` downloads the package from CRAN, and `library()` loads the library

- ▶ Loads the library? `attach()`-es it
 - ▶ Adds it to search path

Attaching

Recall from homework 1 that you can attach() data frames

```
attach(women)
height
```

```
## [1] 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
```

```
height <- height*2.54
height
```

```
## [1] 147.32 149.86 152.40 154.94 157.48 160.02 162.56 16
```



```
## [11] 172.72 175.26 177.80 180.34 182.88
```

```
women$height
```

```
## [1] 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
```

Attaching

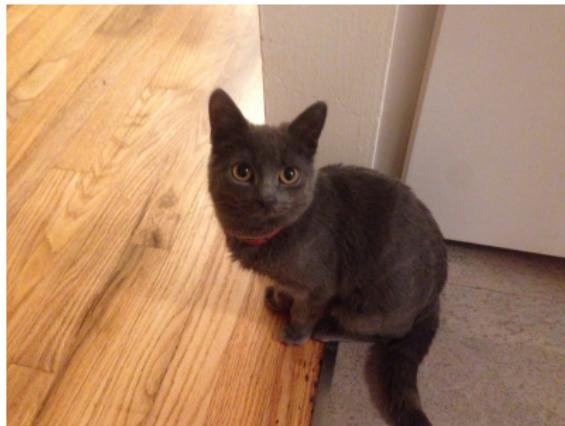
Loading a library places its functions in the search path, but the global environment will always be in the first position

```
ggplot <- function(...) "Ha! Overwritten!"  
library(ggplot2)  
ggplot(iris, aes(x = Sepal.Length, y = Petal.Width))
```

```
## [1] "Ha! Overwritten!"
```

Attach

Moral of the story: Don't attach data frames, but it's fine to attach packages

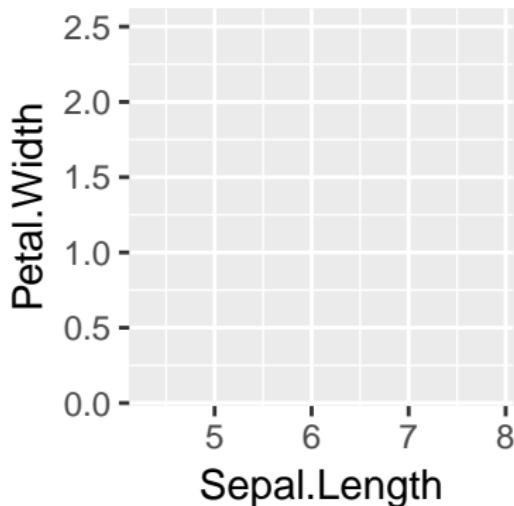


If I ever used 'require()' please forget it and use 'library()' instead

Accessing functions in packages

The operator `x::y` means “the function `y` in the namespace of package `x`”

```
ggplot <- function(...) "Ha! Overwritten!"  
ggplot2::ggplot(iris, aes(x = Sepal.Length,  
                           y = Petal.Width))
```



Namespaces

All packages have a `NAMESPACE` file: a collection of objects to be exported and imported

- ▶ To avoid overwriting users' variables
- ▶ To avoid ambiguity in function calls
- ▶ To ensure the package has everything it needs to run
- ▶ To encourage modular code

Your `NAMESPACE` is auto generated using *tags*; never directly modify your `NAMESPACE` file

Starting an R package

Ideally, create packages from scratch as soon as you begin on a project

- ▶ RStudio->File->New project->New Directory->R Package

The screenshot shows the RStudio interface with the following details:

- Console:** Displays the R session output:

```
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line
help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

Restarting R session...
> library(firstpackage)
> |
```
- Code Editor:** Shows the content of the `hello.R` script:

```
1 # Hello, world!
2 #
3 # This is an example function named 'hello'
4 # which prints 'Hello, world!'.
5 #
6 # You can learn more about package authoring with RStudio at:
7 #
8 # http://r-pkgs.had.co.nz/
9 #
10 # Some useful keyboard shortcuts for package authoring:
11 #
12 # Build and Reload Package: 'Cmd + Shift + B'
13 # Check Package: 'Cmd + Shift + E'
14 # Test Package: 'Cmd + Shift + T'
15
16 hello <- function() {
17   print("Hello, world!")
18 }
19
```
- Files View:** Shows the directory structure:

Name	Size	Md5
..	-	-
.Rbuildignore	28 B	No
DESCRIPTION	373 B	No
firstpackage.Rproj	356 B	No
man	-	-
NAMESPACE	31 B	No
R	-	-
- Build View:** Shows the command being run:

```
R CMD INSTALL --no-multiarch --with-keep.source firstpackage
```

Output of the command:

```
* installing to library '/Library/Frameworks/R.framework/Versions/3.4/Resources/library'
* installing *source* package 'firstpackage' ...
** R
** preparing package for lazy loading
** help
*** installing help indices
*** building package indices
*** testing if installed package can be loaded
* DONE (firstpackage)
```

Starting an R package: DESCRIPTION

Short-term: Keeps track of imports (dependencies)

Long-term: Help others find your package

The screenshot shows a code editor window with four tabs at the top: 'hello.R', 'DESCRIPTION', 'NAMESPACE', and 'hello.Rd'. The 'DESCRIPTION' tab is active. Below the tabs is a toolbar with icons for back, forward, search, and file operations. The main area contains the following R code:

```
1 Package: firstpackage
2 Type: Package
3 Title: What the Package Does (Title Case)
4 Version: 0.1.0
5 Author: Who wrote it
6 Maintainer: The package maintainer <yourself@somewhere.net>
7 Description: More about what it does (maybe more than one line)
8     Use four spaces when indenting paragraphs within the Description.
9 License: What license is it under?
10 Encoding: UTF-8
11 LazyData: true
12 Imports:
13     dplyr,
14     ggplot
15 Suggests:
16     breakaway
17 |
```

Writing R packages: documentation

roxygen documentation makes everything incredibly easy

```
install.packages("devtools")
devtools::install_github("hadley/devtools")
install.packages("roxygen2")
```

Writing R packages: documentation

Under the Build tab, under Build tools, check Generate documentation with Roxygen

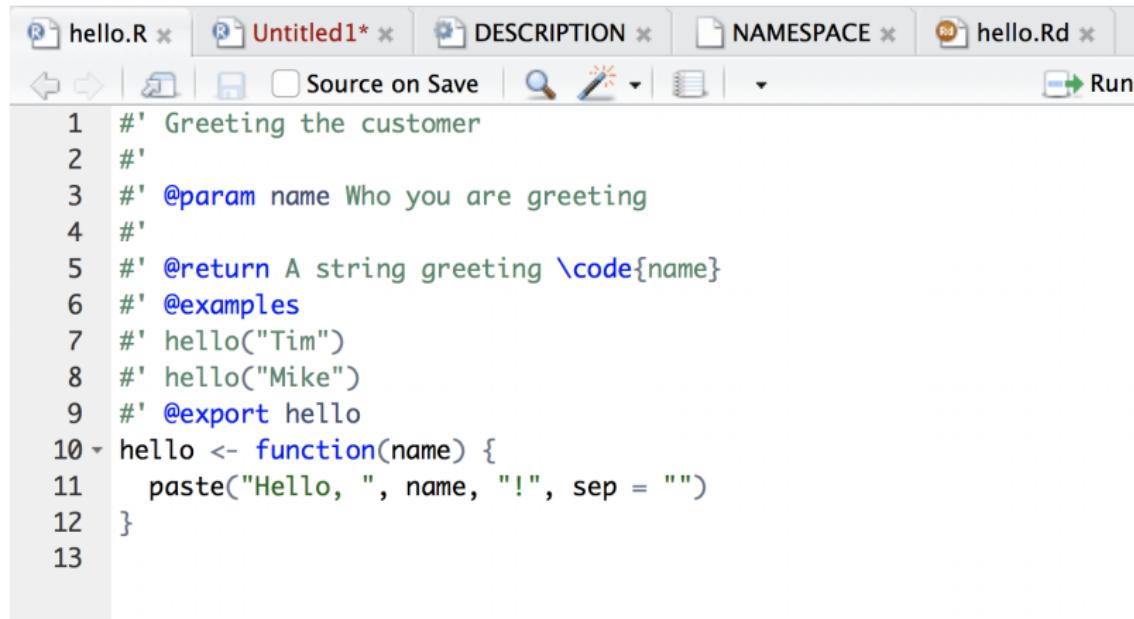
The screenshot shows the RStudio interface with the following details:

- Console:** Displays R code and its output. The output includes `library(firstpackage)`, `hello("Amy")` with result [1] "Hello, Amy!", and a warning about installing roxygen2.
- Project Options dialog:** Opened from the Build tab.
 - Project build tools:** Set to "Package".
 - Package directory:** Set to "(Project Root)".
 - Checklist:** Includes "Use devtools package functions if available" (unchecked) and "Generate documentation with Roxygen" (checked). A "Configure..." button is next to the checked option.
 - Build and Reload — R CMD INSTALL additional options:** Contains `--no-multiarch --with-keep.source`.
 - Check Package — R CMD check additional options:** An empty field.
 - Build Source Package — R CMD build additional options:** An empty field.
 - Build Binary Package — R CMD INSTALL additional options:** An empty field.
 - Help:** "Developing Packages with RStudio" link.
- Project Explorer:** Shows files like hello.R, DESCRIPTION, NAMESPACE, and hello.Rd.
- Code Editor:** Shows the content of hello.R, which contains a Roxygen2 docblock and a function definition for hello.
- Build Tab:** Shows the progress of the build process:
 - (Top Level) Build tab selected.
 - Progress steps: "Building", "Calling to library", "Calling *source* package", "Preparing package for lazy loading", "help", "installing help indices", "building package indices", "testing if installed package can be loaded", and "DONE (firstpackage)".

Writing R packages: documentation

1. Delete the default NAMESPACE file and the man folder
2. Write roxygen comments in your .R file
 - ▶ roxygen comments start with #'
3. cmd/ctrl + shift + b builds the package
4. cmd/ctrl + shift + d autogenerated documentation based on your roxygen comments

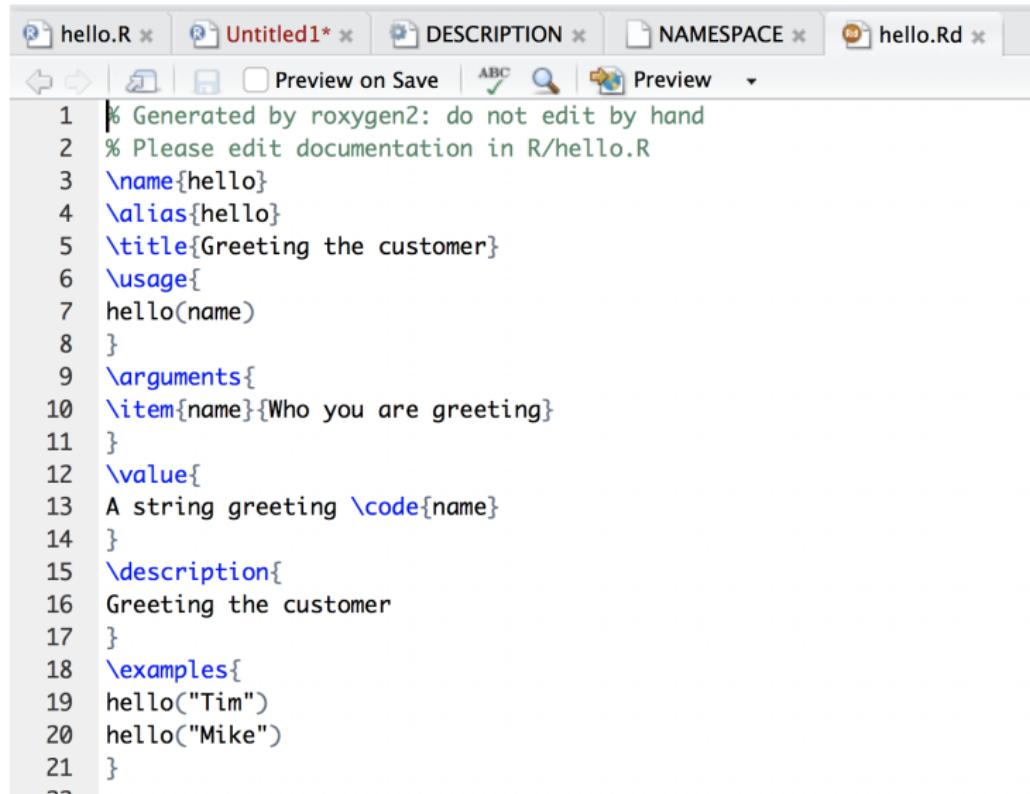
Writing R packages: documentation



The screenshot shows the RStudio interface with the following tabs visible: hello.R, Untitled1*, DESCRIPTION, NAMESPACE, and hello.Rd. The hello.R file contains the following R code:

```
1 #' Greeting the customer
2 #
3 #' @param name Who you are greeting
4 #
5 #' @return A string greeting \code{name}
6 #' @examples
7 #' hello("Tim")
8 #' hello("Mike")
9 #' @export hello
10 hello <- function(name) {
11   paste("Hello, ", name, "!", sep = "")}
```

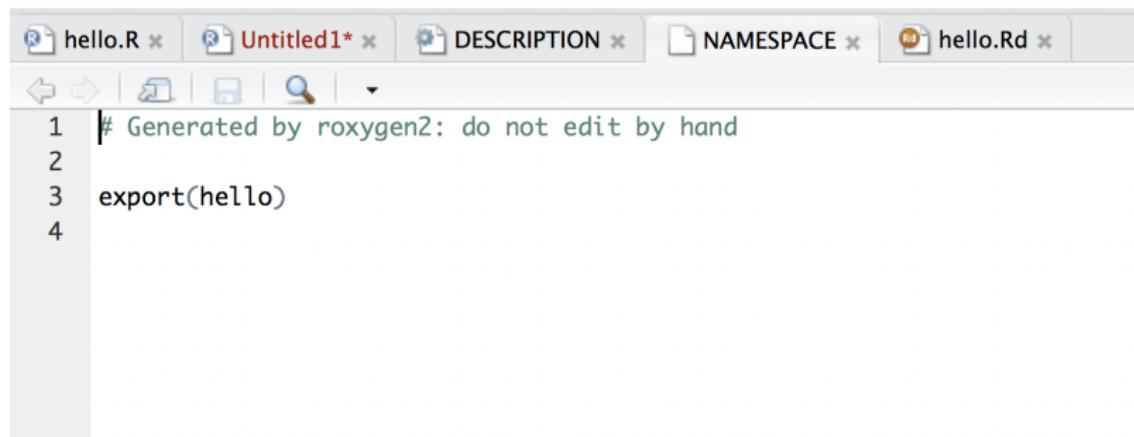
Writing R packages: documentation



The screenshot shows the RStudio interface with the DESCRIPTION tab selected. The code editor displays roxygen2 documentation for a package named 'hello'. The code includes sections for \name, \alias, \title, \usage, \arguments, \item, \value, \description, and \examples.

```
1 # Generated by roxygen2: do not edit by hand
2 % Please edit documentation in R/hello.R
3 \name{hello}
4 \alias{hello}
5 \title{Greeting the customer}
6 \usage{
7   hello(name)
8 }
9 \arguments{
10   \item{name}{Who you are greeting}
11 }
12 \value{
13   A string greeting \code{name}
14 }
15 \description{
16   Greeting the customer
17 }
18 \examples{
19   hello("Tim")
20   hello("Mike")
21 }
```

Writing R packages: documentation



The screenshot shows the RStudio interface with the following details:

- Top Bar:** Shows tabs for "hello.R", "Untitled1*", "DESCRIPTION", "NAMESPACE", and "hello.Rd".
- Toolbar:** Includes icons for back, forward, file operations, and search.
- Code Editor:** Displays the following R code:

```
1 # Generated by roxygen2: do not edit by hand
2
3 export(hello)
4
```

Writing R packages: documentation

```
Console ~/teaching/17-561/lecture7/firstpackage/ ⌂
> library(firstpackage)
Restarting R session...
> library(firstpackage)
> ?hello
> |
```

Files Plots Packages Help Viewer

R: Greeting the customer ▾ Find in Topic

hello {firstpackage} R Documentation

Greeting the customer

Description

Greeting the customer

Usage

hello(name)

Arguments

name Who you are greeting

Value

A string greeting name

Examples

```
hello("Tim")
hello("Mike")
```

[Package *firstpackage* version 0.1.0 [Index](#)]

```
hello.R ⌂ Untitled1* ⌂ DESCRIPTION ⌂ NAMESPACE ⌂ hello.Rd ⌂
Source on Save ⌂ Run ⌂ Source ⌂
1 #' Greeting the customer
2 #
3 #' @param name Who you are greeting
4 #
5 #' @return A string greeting \code{name}
6 #' @examples
7 #' hello("Tim")
8 #' hello("Mike")
9 #' @export hello
10 hello <- function(name) {
11   paste("Hello, ", name, "!", sep = "")
12 }
13 |
```

13:1 (Top Level) ⌂ R Script ⌂

Environment History Build Git

Build & Reload Check More ▾

```
=> devtools::document(roclets=c('rd', 'namespace'))
```

Updating *firstpackage* documentation
Loading *firstpackage*
Documentation completed

Writing R packages: tags

```
#' \item{seest}{ The standard error in the diversity estimate. } \item{full}{  
#' The chosen nonlinear model for frequency ratios. } \item{ci}{ An asymmetric  
#' 95\% confidence interval for diversity. }  
#' @note \code{breakaway} presents an estimator of species richness that is  
#' well-suited to the high-diversity/microbial setting. However, many microbial  
#' datasets display more diversity than the Kemp-type models can permit. In  
#' this case, the log-transformed WLRM diversity estimator of Rocchetti et. al.  
#' (2011) is returned. The authors' experience suggests that some datasets that  
#' require the log-transformed WLRM contain ``false'' diversity, that is,  
#' diversity attributable to sequencing errors (via an inflated singleton  
#' count). The authors encourage judicious use of diversity estimators when the  
#' dataset may contain these errors, and recommend the use of  
#' \code{\link{breakaway_noF1}} as an exploratory tool in this case.  
#' @author Amy Willis  
#' @seealso \code{\link{breakaway_noF1}}; \code{\link{apples}}  
#' @references Willis, A. and Bunge, J. (2015). Estimating diversity via  
#' frequency ratios. Biometrics, 71(4), 1042--1049.  
#'  
#' Rocchetti, I., Bunge, J. and Bohning, D. (2011). Population size estimation  
#' based upon ratios of recapture probabilities. Annals of Applied  
#' Statistics, 5.  
#' @keywords diversity microbial models nonlinear  
#' @examples  
#'  
#|  
#' breakaway(apples)  
#' breakaway(apples, plot = FALSE, output = FALSE, answers = TRUE)  
#'  
#'  
#' @export breakaway  
breakaway <- function(my_data, output=TRUE, plot=FALSE, answers=FALSE, force=FALSE, useAll:  
  
## read in data  
if( !(is.matrix(my_data) || is.data.frame(my_data))) {  
  ...  
}
```

Writing R packages: documentation

```
# Generated by roxygen2: do not edit by hand

export(betta)
export(betta_pic)
export(betta_points)
export(betta_random)
export(breakaway)
export(breakaway_clean)
export(breakaway_nof1)
export(build_frequency_count_tables)
export(choo1)
export(choo1_bc)
export(choo_bunge)
export(estimate_alpha_better)
export(frequency_count_or_proportion_or_column)
export(gini)
export(hill)
export(hill_better)
export(hill_pic)
export(inverse_simpson)
export(inverse_simpson_better)
export(make_frequency_count_table)
export(objective_bayes_geometric)
export(objective_bayes_mixedgeo)
export(objective_bayes_negbin)
export(objective_bayes_poisson)
export(plot_alpha)
export(proportions_instead)
export(resample_estimate)
export(rnbinomtable)
export(rztnbinomtable)
export(sample_size_estimate)
export(sample_size_figure)
export(shannon)
export(shannon_better)
export(simpson)
export(subsample_otu)
export(to_proportions)
export(wirm_transformed)
export(wirm_untransformed)
import(reshape2)
importFrom(MASS,mvrnorm)
importFrom(ggplot2,element_text)
importFrom(ggplot2,facet_wrap)
importFrom(ggplot2,geom_errorbar)
importFrom(plyr,is_discrete)
importFrom(stats,acf)
```

Building packages

Core “other stuff” for a package

- ▶ Code: written by you, goes into `R` folder (no subdirectories allowed)
- ▶ `DESCRIPTION`: written by you
- ▶ Documentation: autogenerated by `roxygen` comments, goes into `man` folder
- ▶ `NAMESPACE`: autogenerated by `roxygen` comments

documentation: vignettes

```
1+ ---
2 title: "Getting started with breakaway"
3 author: "Amy Willis"
4 date: `r Sys.Date()`
5 output: rmarkdown::html_vignette
6 vignette: >
7   %>%VignetteIndexEntry{getting-started}
8   %>%VignetteEngine{knitr::rmarkdown}
9   %>%VignetteEncoding{UTF-8}
10 ---
11
12 `breakaway` is a package for species richness estimation and modelling. As the package has grown and users have requested more functionality, it contains some basic generalities about the statistical philosophy that underpins `breakaway` to the general alpha diversity field. Because of the flexibility of the modelling strategies, most users of breakaway are microbial ecologists with very large OTU tables, however, nothing should exclude a macroecologist from using the same tools. If you have a macroecology dataset and would like to use this package, I would love to hear from you so please feel free to contact me (either via GitHub's Issues/Projects infrastructure).
13
14+ ## Vignette Info
15
16 This vignette will lead you through
17
18 - loading in your data and covariate information
19 - creating frequency tables
20 - using `breakaway` to estimate species richness
21 - using the objective bayes methods to estimate species richness
22 - estimating various alpha diversity indices and providing a very basic resampling method for ascribing variability to them
23 - using `beta` to model changes in alpha diversity with covariates
24
25 If there is something that you would like explained please feel free to request it
26
27+ ## Loading the package and data
28
29 Download the latest version of the package from github.
30
31+ ````{r}
32+ ### Run the first two lines at home! #####
33+ # install.packages("devtools")
34+ # devtools::install_github("adw96/breakaway")
35+ library(breakaway)
36+ data(toy_otu_table)
37+ data(toy_metadata)
```

documentation: vignettes

277 lines (199 sloc) | 13.3 KB

Raw Blame History

title	author	date	output	vignette
Getting started with breakaway	Amy Willis	'r Sys.Date()'	rmarkdown::html_vignette	%VignetteIndexEntry(getting-started)%VignetteEngine(knit::rmarkdown)%VignetteEncoding(UTF-8)

`breakaway` is a package for species richness estimation and modelling. As the package has grown and users have requested more functionality, it contains some basic generalisations of the statistical philosophy that underpins `breakaway` to the general alpha diversity case. Because of the flexibility of the modelling strategies, most users of `breakaway` are microbial ecologists with very large OTU tables, however, nothing should exclude a macroecologist from using the same tools. If you have a macroecology dataset and want to use this package, I would love to hear from you so please feel free to contact me (email or via Github's Issues/Projects infrastructure).

Vignette Info

This vignette will lead you through

- loading in your data and covariate information
- creating frequency tables
- using `breakaway` to estimate species richness
- using the objective bayes methods to estimate species richness
- estimating various alpha diversity indices and providing a very basic resample-based method for ascribing variability to them
- using `betta` to model changes in alpha diversity with covariates

If there is something that you would like explained please feel free to request it!

Loading the package and data

Download the latest version of the package from github.

```
## Run the first two lines at home! ####
# install.packages("devtools")
# devtools::install_github("adw96/breakaway")
library(breakaway)
data(toy_otu_table)
data(toy_metadata)

## For historical reasons we going to call them:
otu_data <- toy_otu_table
meta_data <- toy_metadata
```

Other things to keep in mind

1. `testthat` is a H.W. package to write unit tests
2. Datasets take a special format
 - ▶ Don't release data in the same way as functions
3. `rm(list=ls(all=TRUE))` removes everything in the global environment
 - ▶ But not packages!
4. Consider making custom classes (S3 or S4)
5. `pkgdown` is a H.W. package that can autogenerate a website `build_site()`

Coming up

- ▶ New homework: your first R package
- ▶ Homework due Tuesday 21 November
 - ▶ Good luck with your 512 mid-term!
 - ▶ Move office hours from 3 p.m. Tuesday 14 November to 3 p.m. Friday 17 November?
- ▶ Next week: scripting + departmental computing with Brian Williamson