

STAT 545A

Class meeting #4

Monday, September 16, 2013

Dr. Jennifer (Jenny) Bryan

Department of Statistics and Michael Smith Laboratories



any questions from tutorial re: R objects and indexing?

Have a look around the bullet lists for HW 1 and HW 2 and take a hard look at your links and filenames. Do yours look funny? Don't wait for Song or me to alert you. Fix it.

No class this Wed.

HW #3 “Data Aggregation”. Due before class next Monday. Details coming soon to web. Submission process likely to change

Other stuff to do before next class: learn how to get help and participate in the discussion about datasets

A few words from Matt G re: anti-aliasing and PNGs and Windows; see his post on the Google Group and/or his demo on Rpubs.

How to ask a
question to maximize
chance of getting an
answer.

The R Inferno

Patrick Burns¹

30th April 2011

Read about the
9th circle of R hell
in The R Inferno.

Circle 9

Homework reading

Unhelpfully Seeking Help

Here live the thieves, guarded by the centaur Cacus. The inhabitants are bitten by lizards and snakes.

There's a special place for those who—not being content with one of the 8 Circles we've already visited—feel compelled to drag the rest of us into hell.

The road to writing a mail message should include at least the following stops:

9.1 Read the appropriate documentation.

“RTFM” in the jargon. There is a large amount of documentation about R, both official and contributed, and in various formats. A large amount of documentation means that it is often nontrivial to find what you are looking for—especially when frustration is setting in and blood pressure is rising.

Breathe.

There are various searches that you can do. R functions for searching include `help.search`, `RSiteSearch` and `apropos`.

If you are looking for particular functionality, then check the Task Views (found on the left-side menu of CRAN).

If you have an error, then look in rather than out—debug the problem. One way of debugging is to set the `error` option, and then use the `debugger` function:

```
options(error=dump.frames)
# command that causes the error
debugger()
```

The `debugger` function then provides a menu of the stack of functions that have been called at the point of the error. You can inspect the state of the objects inside these functions, and hopefully understand what the problem is.

Unhelpfully Seeking Help

Patrick Burns¹

30th April 2011

Here live the thieves, guarded by the centaur Cacus. The inhabitants are bitten by lizards and snakes.

There's a special place for those who—not being content with one of the 8 Circles we've already visited—feel compelled to drag the rest of us into hell.

“If someone has the wit and knowledge to answer your question, they probably have other things they would like to do. Making your message clear, concise and user-friendly gives you the best hope of at least one of those strangers diverting their attention away from their life towards your problem.”

How to ask a question to maximize chance of getting an answer.

Homework reading

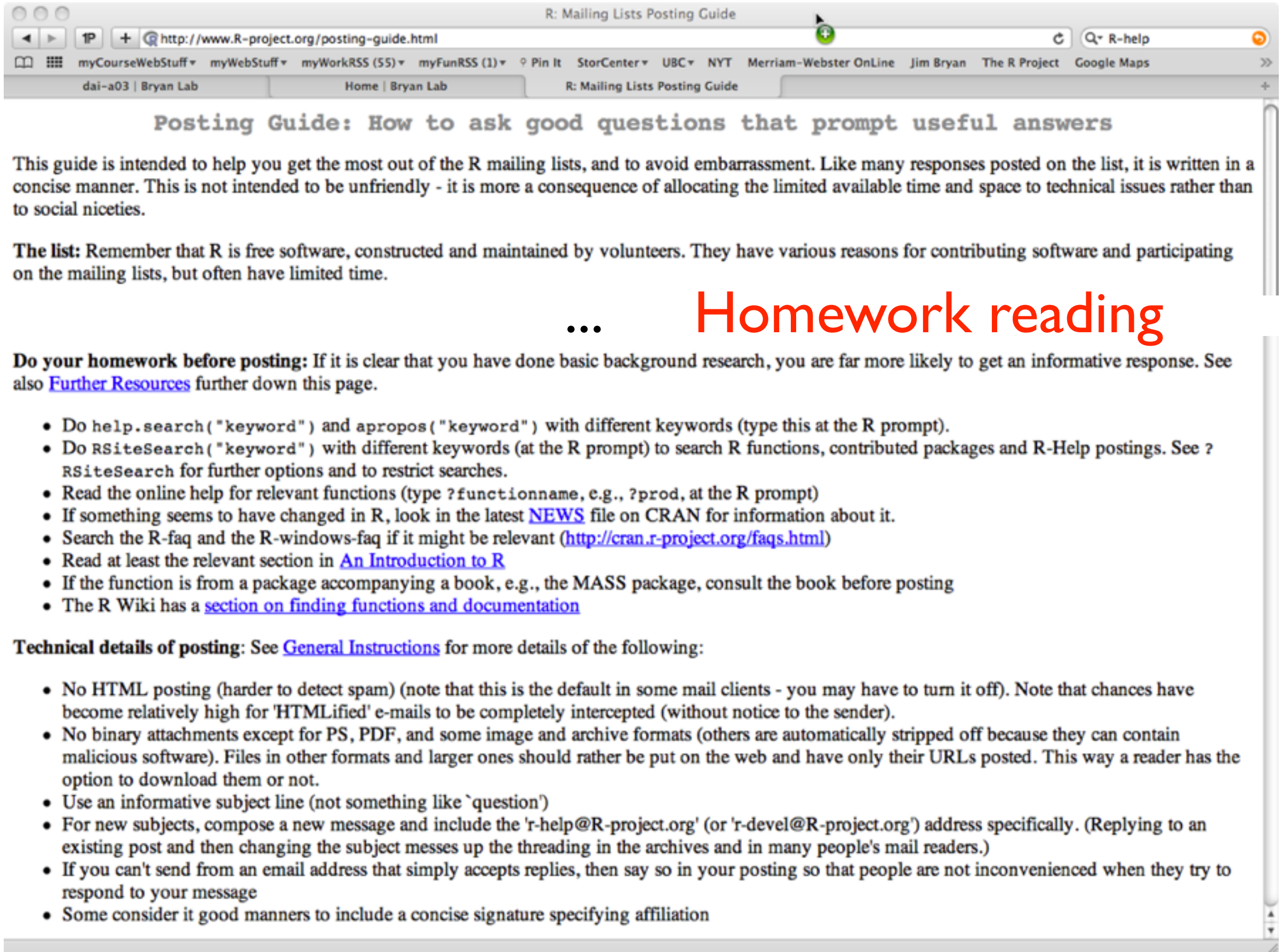
“How To Ask Questions The Smart Way”
by Eric Raymond and Rick Moen

It's OK to be ignorant; it's not OK to play stupid.

So, while it isn't necessary to already be technically competent to get attention from us, it is necessary to **demonstrate the kind of attitude that leads to competence: alert, thoughtful, observant, willing to be an active partner in developing a solution.**

The best way to get a rapid and responsive answer is to ask it like a person with smarts, confidence, and clues who just happens to need help on one particular problem.

R-help (you probably shouldn't be posting here!) has a good posting guide



The screenshot shows a web browser window titled "R: Mailing Lists Posting Guide". The address bar shows the URL "http://www.R-project.org/posting-guide.html". The browser's search bar contains "R-help". The page content includes a title "Posting Guide: How to ask good questions that prompt useful answers", an introductory paragraph, a section "The list:" with advice on the mailing lists, a section "Do your homework before posting:" with a list of preparatory steps, and a section "Technical details of posting:" with a list of formatting rules. The text is presented in a clean, readable font with some links highlighted in blue.

Posting Guide: How to ask good questions that prompt useful answers

This guide is intended to help you get the most out of the R mailing lists, and to avoid embarrassment. Like many responses posted on the list, it is written in a concise manner. This is not intended to be unfriendly - it is more a consequence of allocating the limited available time and space to technical issues rather than to social niceties.

The list: Remember that R is free software, constructed and maintained by volunteers. They have various reasons for contributing software and participating on the mailing lists, but often have limited time.

... **Homework reading**

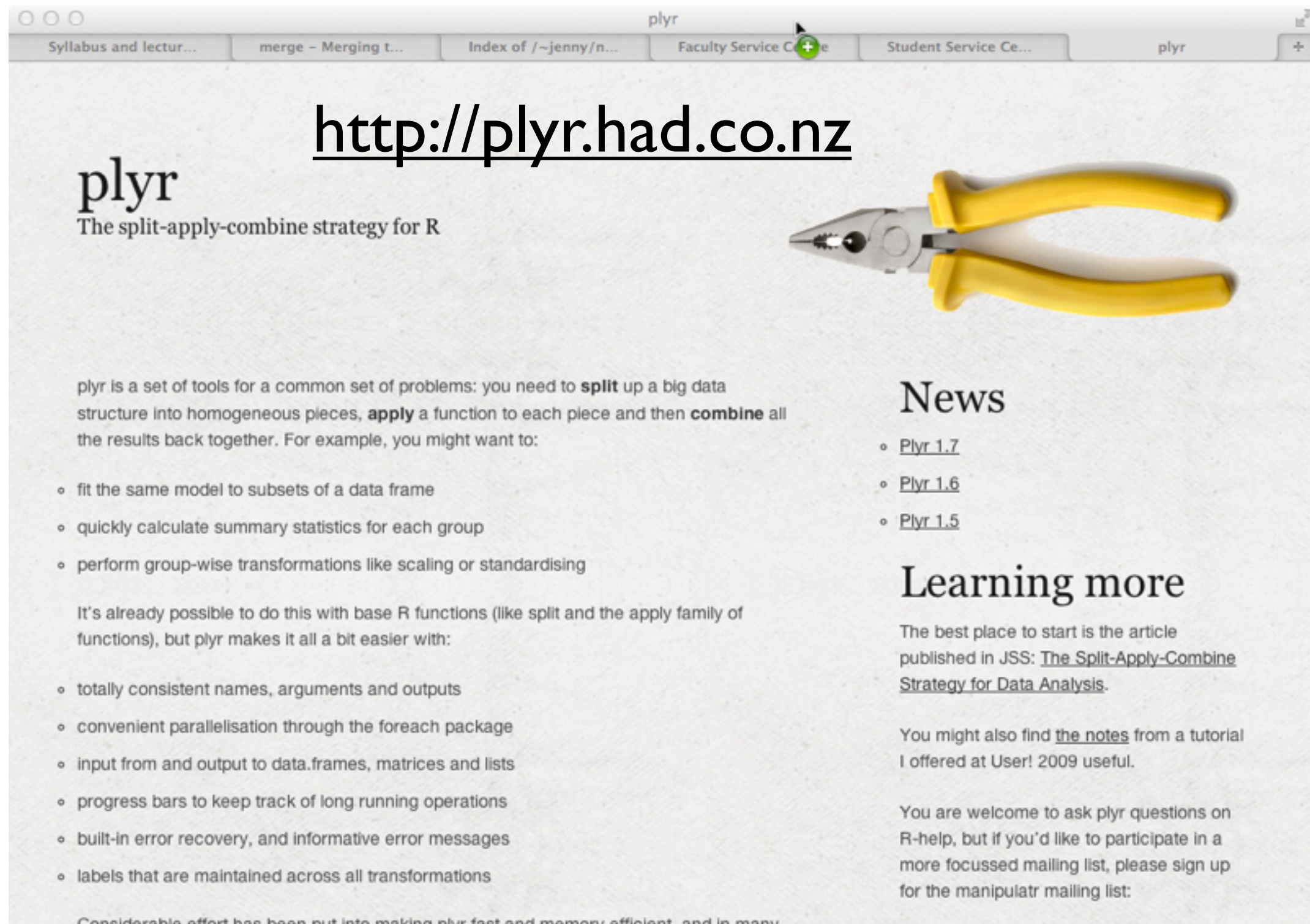
Do your homework before posting: If it is clear that you have done basic background research, you are far more likely to get an informative response. See also [Further Resources](#) further down this page.

- Do `help.search("keyword")` and `apropos("keyword")` with different keywords (type this at the R prompt).
- Do `RSiteSearch("keyword")` with different keywords (at the R prompt) to search R functions, contributed packages and R-Help postings. See `?RSiteSearch` for further options and to restrict searches.
- Read the online help for relevant functions (type `?functionname`, e.g., `?prod`, at the R prompt)
- If something seems to have changed in R, look in the latest [NEWS](#) file on CRAN for information about it.
- Search the R-faq and the R-windows-faq if it might be relevant (<http://cran.r-project.org/faqs.html>)
- Read at least the relevant section in [An Introduction to R](#)
- If the function is from a package accompanying a book, e.g., the MASS package, consult the book before posting
- The R Wiki has a [section on finding functions and documentation](#)

Technical details of posting: See [General Instructions](#) for more details of the following:

- No HTML posting (harder to detect spam) (note that this is the default in some mail clients - you may have to turn it off). Note that chances have become relatively high for 'HTMLified' e-mails to be completely intercepted (without notice to the sender).
- No binary attachments except for PS, PDF, and some image and archive formats (others are automatically stripped off because they can contain malicious software). Files in other formats and larger ones should rather be put on the web and have only their URLs posted. This way a reader has the option to download them or not.
- Use an informative subject line (not something like 'question')
- For new subjects, compose a new message and include the 'r-help@R-project.org' (or 'r-devel@R-project.org') address specifically. (Replying to an existing post and then changing the subject messes up the threading in the archives and in many people's mail readers.)
- If you can't send from an email address that simply accepts replies, then say so in your posting so that people are not inconvenienced when they try to respond to your message
- Some consider it good manners to include a concise signature specifying affiliation

The plyr package is what I advise long-term for data aggregation.



<http://plyr.had.co.nz>

plyr

The split-apply-combine strategy for R



plyr is a set of tools for a common set of problems: you need to **split** up a big data structure into homogeneous pieces, **apply** a function to each piece and then **combine** all the results back together. For example, you might want to:

- fit the same model to subsets of a data frame
- quickly calculate summary statistics for each group
- perform group-wise transformations like scaling or standardising

It's already possible to do this with base R functions (like `split` and the `apply` family of functions), but `plyr` makes it all a bit easier with:

- totally consistent names, arguments and outputs
- convenient parallelisation through the `foreach` package
- input from and output to `data.frames`, matrices and lists
- progress bars to keep track of long running operations
- built-in error recovery, and informative error messages
- labels that are maintained across all transformations

News

- [Plyr 1.7](#)
- [Plyr 1.6](#)
- [Plyr 1.5](#)

Learning more

The best place to start is the article published in JSS: [The Split-Apply-Combine Strategy for Data Analysis](#).

You might also find [the notes](#) from a tutorial I offered at User! 2009 useful.

You are welcome to ask plyr questions on R-help, but if you'd like to participate in a more focussed mailing list, please sign up for the manipulatr mailing list:

JB found it hard to get started with plyr by reading documentation for individual functions. You need to get the big picture and then it will all come into focus. Read this paper!

Hadley Wickham.

The split-apply-combine strategy for data analysis.

Journal of Statistical Software, vol. 40, no. 1, pp. 1–29, 2011.

<http://www.jstatsoft.org/v40/i01/paper>



Journal of Statistical Software

April 2011, Volume 40, Issue 1.

<http://www.jstatsoft.org/>

The Split-Apply-Combine Strategy for Data Analysis

Hadley Wickham
Rice University

Abstract

Many data analysis problems involve the application of a split-apply-combine strategy, where you break up a big problem into manageable pieces, operate on each piece independently and then put all the pieces back together. This insight gives rise to a new R package that allows you to smoothly apply this strategy, without having to worry about the type of structure in which your data is stored.

The paper includes two case studies showing how these insights make it easier to work with batting records for veteran baseball players and a large 3d array of spatio-temporal ozone measurements.

Keywords: R, apply, split, data analysis.

split apply combine

<i>Input</i>	<i>Output</i>			
	Array	Data frame	List	Discarded
Array	aaply	adply	alply	a_ply
Data frame	daply	ddply	dlply	d_ply
List	laply	ldply	llply	l_ply

- `a*ply(.data, .margins, .fun, ..., .progress = "none")`
- `d*ply(.data, .variables, .fun, ..., .progress = "none")`
- `l*ply(.data, .fun, ..., .progress = "none")`

we spent the rest of class time going through
this tutorial:

http://www.stat.ubc.ca/~jenny/STAT545A/block04_dataAggregation.html