SWC May 2013 NC

Jenny Bryan

R day
Slides I showed that were borrowed from my other teaching materials

block01

RStudio ... console, editor, help, completion, sending code from editor to console, history to editor, (assign and print!), keyboard shortcuts

organize a project within a local directory
read data, write results and figs there
store code that converts data into results and figs there

make that directory R's working directory when working on that project

RStudio Projects make all this magic happen effortlessly
use them

save the "keeper" code in a script
make it complete, e.g. load data and write figures to file w/ code not mouse
re-run start-to-finish in a fresh R session (i.e. workspace empty to start)
periodically to verify

block02

use str(), head(), tail(), peek()*, names(), length(), nrow(), etc incessantly to keep informed about what sort of objects you're working with

make plots early and often

data.frame = default data storage receptacle for anything rectangular, spreadsheet-y

become the boss of read.table and friends

work *in situ* within your data.frames (vs. creating little copies of certain variables, of certain rows, etc etc) unless you have persistent need for the data excerpt

- pass data.frame as "data =" argument, use "subset =" argument
- fake it with "with()" when above not available

subset() to subset data.frame (and other things)

use names instead of, e.g. column numbers to make code more robust and self-documenting

* JB personal function to look at random sample of rows; we made an entry-level version live at bootcamp

block03

if you need to compute or graph something for various subsets of your data, don't settle for explicit, user-directed "subset and loop"; use pro tools
- graphing: y ~ x | z in lattice or facetting in ggplot2
- apply family of functions (base R)
- plyr package "split-apply-combine"

(we didn't get to reshaping at bootcamp)
data reshaping: comes up alot if you are graphically ambitious
- data has a way of getting wider and shorter ... effort required to keep tall and skinny
- just do it.
- JB likes DIY
- ? reshape package ?

block04

reorder a factor rationally, e.g. continent based on lifeExpectancy or slope
drop unused factor levels

practice writing and reading data.frame

writing non-rectangular output to file, e.g. sink() and how probably replaced by Compile notebook ... (a button RStudio offers that relies on the knitr package)

we didn't get to all of this ...

writing figures to file dev.print() trick vs. opening device .... closing

say something about a for loop? sapply trick? plyr looping function with no output?
show a more complicated project w/ sub directories for code, data, figs, results
constructing filenames paste0, file.path, list.files

challenge:
write a script to write mini-datasets, e.g. one per country, to file
write another script to read them back in and write some PDF to file, on per country

Reach out and touch -- but do not print to screen - your data

```
str()
summary()
head()
tail()
peek() -- not built-in
mode()
class()
```

Reminder of other functions that help you to get and stay acquainted with your R objects. Use them early, use them often.

Simple view of simple R objects that will get you pretty far

Simple view	Technically correct R view		
	mode	class	typeof
character	character	character	character
logical	logical	logical	logical
numeric	numeric	integer or numeric	integer or double
factor	numeric	factor	integer

Simple view of simple R objects that will get you pretty far

Simple view	Technically correct R view		
	mode	class	typeof
character	character	character	character
logical	logical	logical	logical
numeric	numeric	integer or numeric	integer or double
factor	numeric	factor	integer

Factors

See Chapter 5 of Spector (2008).

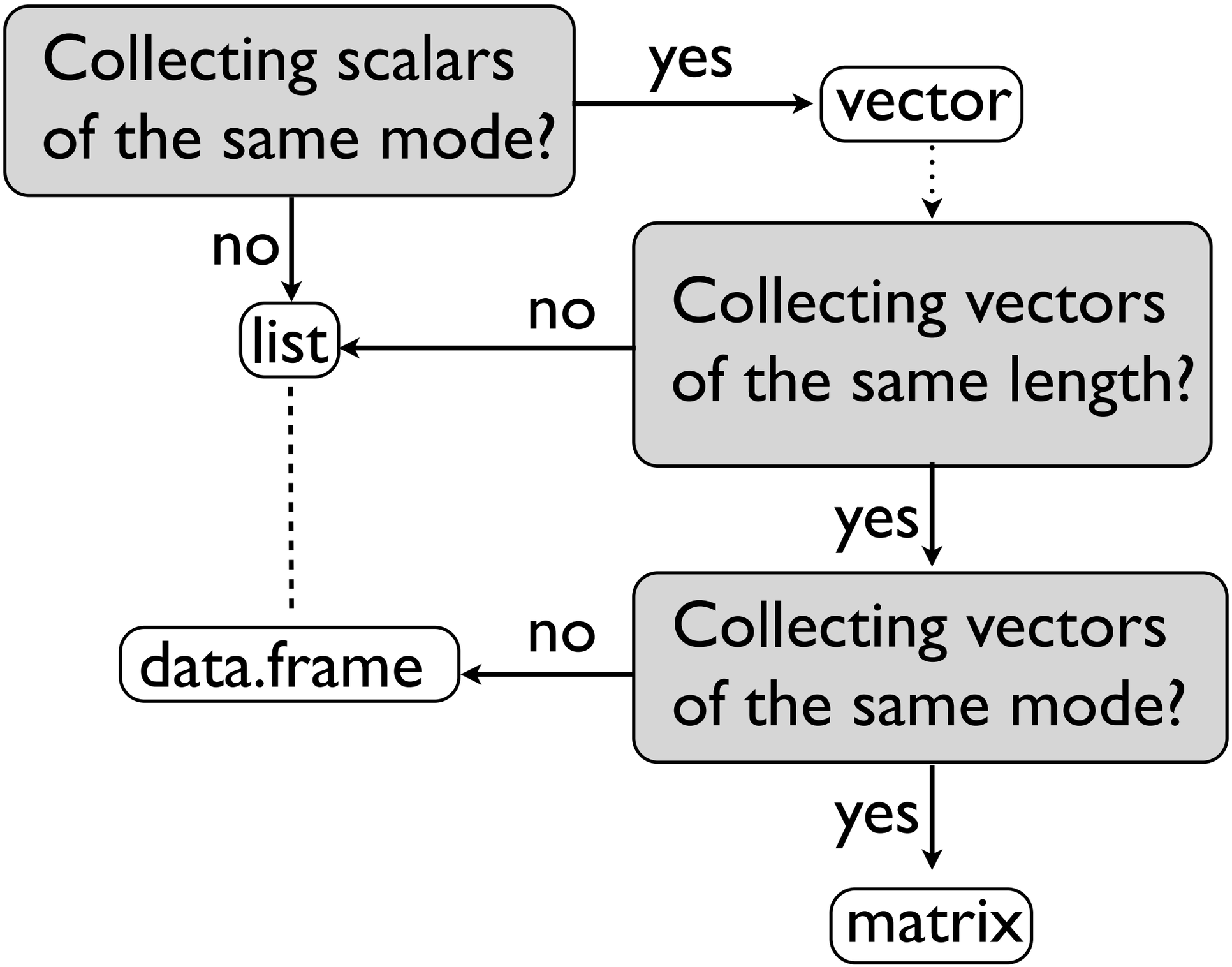
- Valuable way to store categorical data BUT ...
  - Jenny's Law: A factor variable will be the source of at least one major headache in each data analysis, costing me ~~hours~~ several minutes of valuable time.
- Why needed
  - In modelling: proper use of factors will make it much easier to specify models, construct contrasts, etc.
  - In visualization: lattice is smart about conditioning on factors or conveying factor levels through color, line type, etc.

Factors

See Chapter 5 of Spector (2008).

- Basic trickiness: Factors are stored as integers, with an associated set of labels (usually character strings). The character info is more visible/interpretable, but *don't ever forget* factors are really numeric.
- Factors are "high-maintenance" variables, but I still advise you to Embrace Factors and Their Labels/Levels.
  - Make the labels informative yet concise.
  - Make a deliberate choice of the first or reference level, when relevant.
  - Choose the overall order in a principled way, when relevant. Be prepared to change the order or drop levels at various points in an analysis.

"Simple view" of data collections

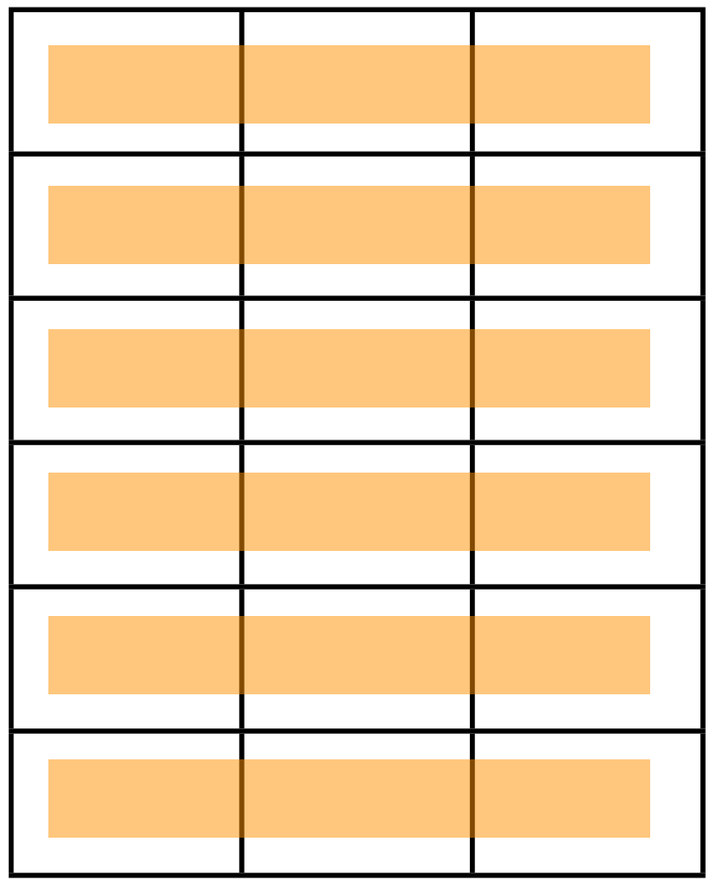


For those situations ... when you need to do <sthg> for various ‘chunks’ of your dataset

Best method depends on the nature of these chunks

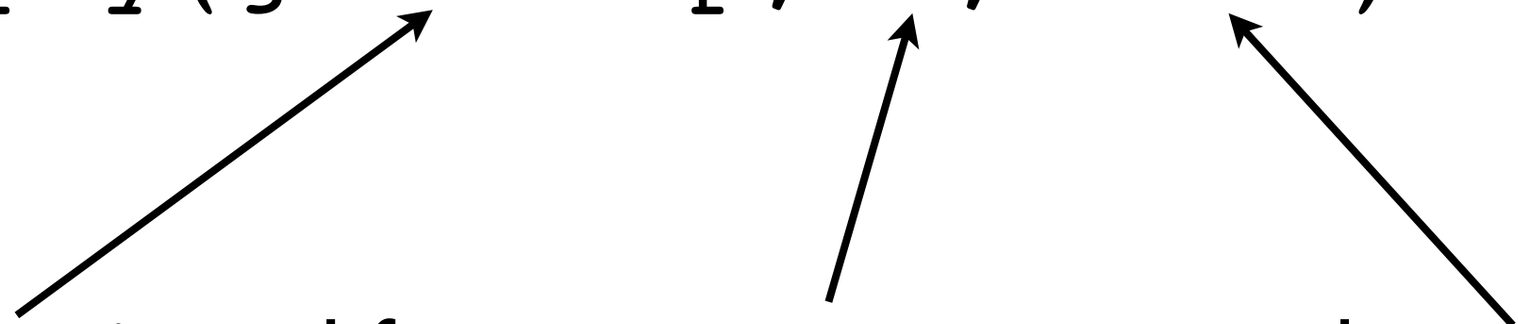
chunks are ...	relevant functions
rows, columns, etc. of matrices / arrays	apply
components of a list(remember data.frames are lists!)	sapply, lapply
groups induced by levels of ≥ 1 factor(s)	aggregate tapply by split (+ [sl]apply)

chunks are ...	relevant functions
rows, columns, etc. of matrices / arrays	apply
components of a list(remember data.frames are lists!)	sapply, lapply
groups induced by levels of ≥ 1 factor(s)	aggregate tapply by split (+ [sl]apply)



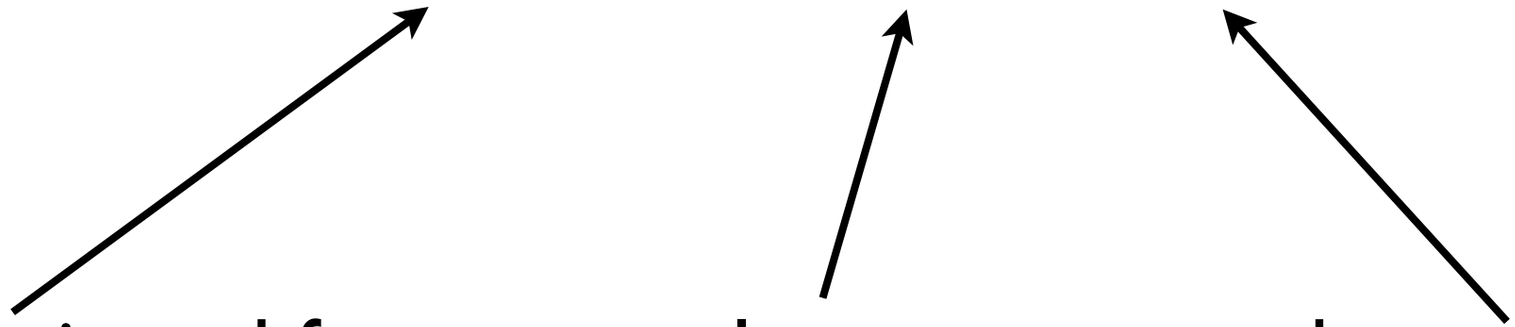
Let chunk = row or column of matrix

```
apply(jLifeExp, 1, mean)
```



"Take this matrix and for every row, compute the mean."

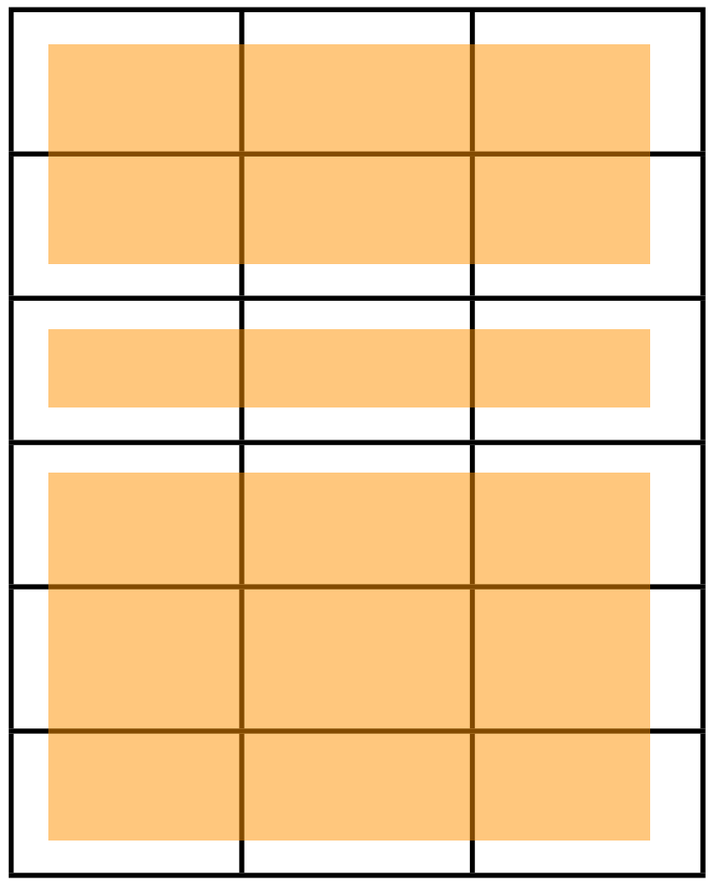
```
apply(jLifeExp, 2, median)
```



"Take this matrix and for every column, compute the median."

Note: apply() works perfectly well on arrays of dimension 3 and higher. Read the docs and proceed with care.

chunks are ...	relevant functions
rows, columns, etc. of matrices / arrays	apply
components of a list(remember data.frames are lists!)	sapply, lapply
groups induced by levels of ≥ 1 factor(s)	tapply aggregate by split (+ [sl]apply)



Let chunk = ragged groups of elements of a vector or rows of a data.frame

Divide this vector into chunks ...

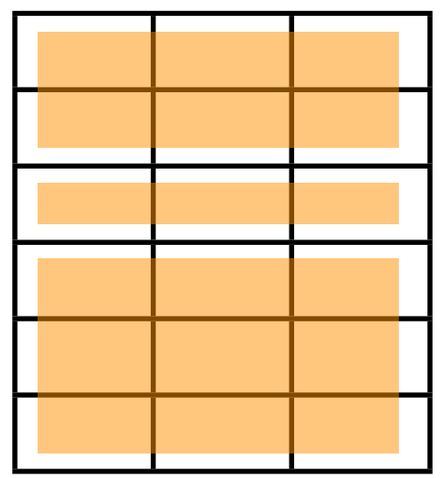
based on this factor and ...

apply this function to each chunk

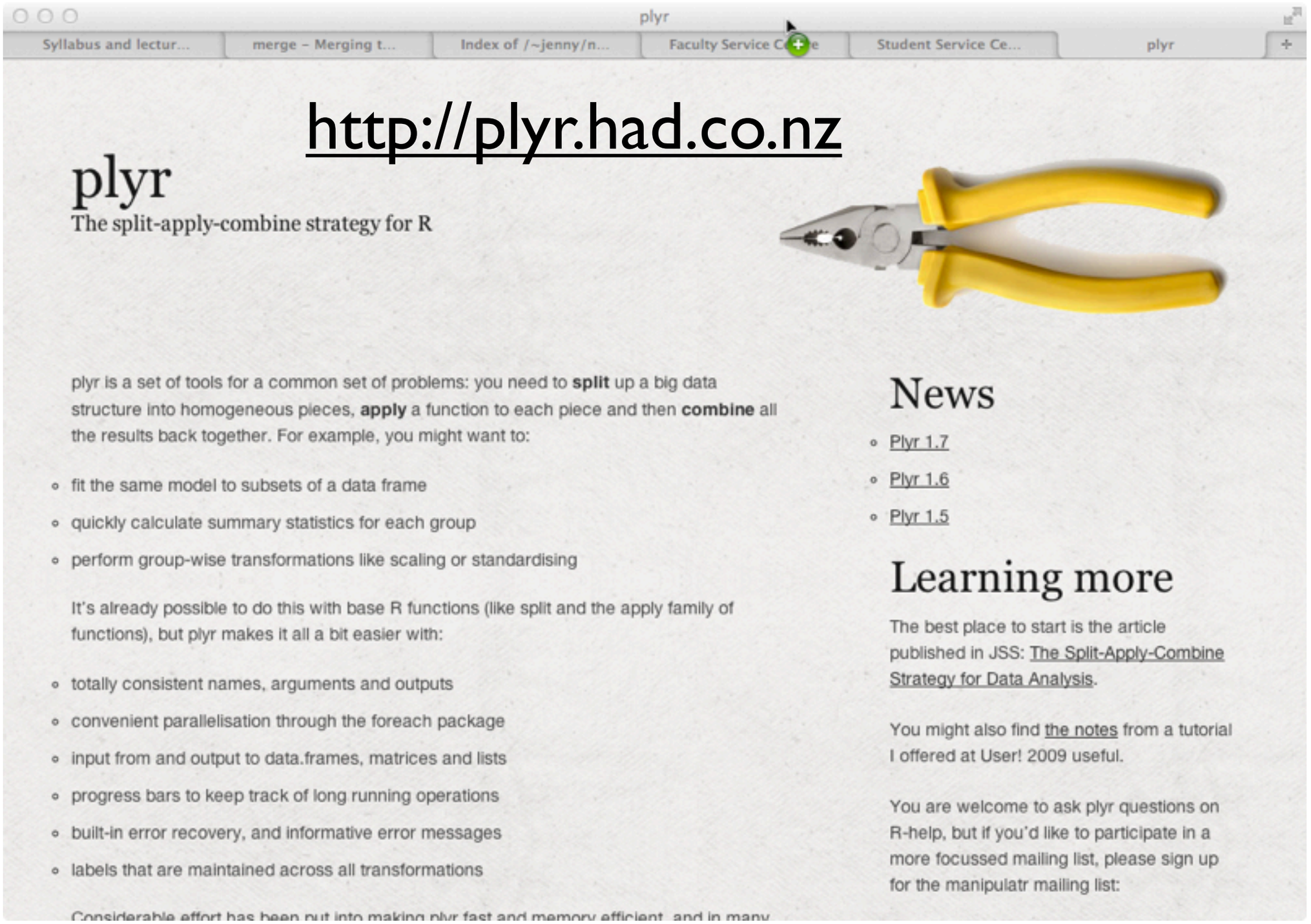
```
## introducing tapply
with(gDat,
     tapply(lifeExp, continent, max))
  Africa Americas     Asia   Europe  Oceania
  76.442   80.653   82.603   81.757   81.235
```

The function to evaluate can be built-in, like max() above, custom but defined in advance, or custom and defined 'on the fly', as I've done below and later in this class.

```
> ## how many countries for each continent?
> with(gDat,
+      tapply(country, continent, function(x) {
+        length(unique(x))
+      }))
  Africa Americas     Asia   Europe  Oceania
      52       25       33       30        2
```



The plyr package is what I advise long-term for data aggregation. Still good to know about the base R functions, though.....



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.