A PRACTICAL HISTORY OF R (WHERE THINGS CAME FROM)

Roger Bivand

16 May 2018

SOURCE AND LINKS

The keynote presentation files are on github at: rsbivand/eRum18

INTRODUCTION

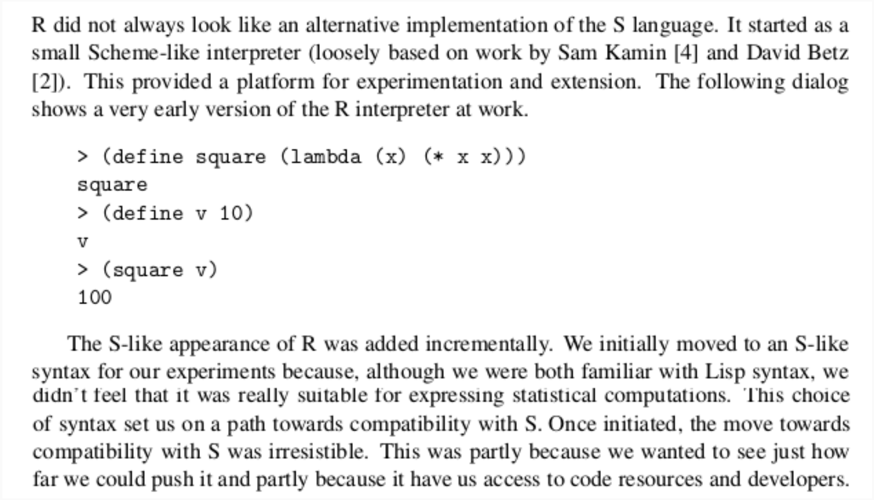
- Not infrequently, we wonder why choices such as `stringsAsFactors=TRUE` or `drop=TRUE` were made.
- Understanding the original uses of S and R (in the 1980s and 1990s), and seeing how these uses affected the development of R lets us appreciate the robustness of R's ecosystem.
- This keynote uses readings of the R sources and other information to explore R's history. The topics to be touched on include the "colour" books (brown, blue, white, green), interlinkages to SICP (Scheme) and LispStat
- We'll also touch on the lives of R-core, the mailing lists and CRAN, and Ancients and Moderns (see Exploring the CRAN social network).

History of R and its data structures

SOURCES

- Rasmus Bååth has a useful blog piece on R's antecedents in the S language
- Something similar is present in the second chapter of (Chambers, 2016), from the viewpoint of one of those responsible for the development of the S language
- In addition to S, we need to take SICP and Scheme into account (Abelson and Sussman, 1996, second edition), as described by Ihaka and Gentleman (1996) and Wickham (2014)
- Finally, LispStat and its creators have played and continue to play a major role in developing R (Tierney, 1990, 1996, 2005)

EARLY R WAS SCHEME VIA SICP



R did not always look like an alternative implementation of the S language. It started as a small Scheme-like interpreter (loosely based on work by Sam Kamin [4] and David Betz [2]). This provided a platform for experimentation and extension. The following dialog shows a very early version of the R interpreter at work.

```
> (define square (lambda (x) (* x x)))
square
> (define v 10)
v
> (square v)
100
```

The S-like appearance of R was added incrementally. We initially moved to an S-like syntax for our experiments because, although we were both familiar with Lisp syntax, we didn't feel that it was really suitable for expressing statistical computations. This choice of syntax set us on a path towards compatibility with S. Once initiated, the move towards compatibility with S was irresistible. This was partly because we wanted to see just how far we could push it and partly because it have us access to code resources and developers.

FROM S TO R: BROWN BOOKS

Becker and Chambers (1984): S: An Interactive Environment for Data Analysis and Graphics, A.K.A. the Brown Book

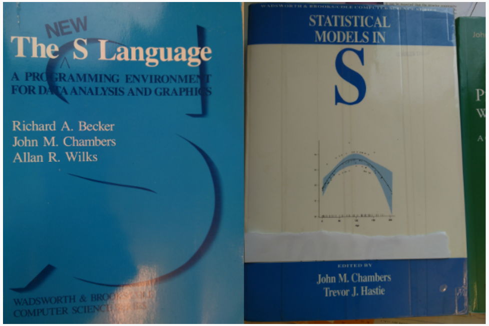
Becker and Chambers (1985): Extending the S System

S
AN INTERACTIVE ENVIRONMENT FOR DATA ANALYSIS AND GRAPHICS
Richard A. Becker
John M. Chambers
WADSWORTH STATISTICS/PROBABILITY SERIES
EXTENDING THE S SYSTEM
Richard A. Becker
John M. Chambers
WADSWORTH STATISTICS/PROBABILITY SERIES

FROM S TO R: BLUE AND WHITE BOOKS

Becker et al. (1988): The New S Language: A Programming Environment for Data Analysis and Graphics, A.K.A. the Blue Book.

Chambers and Hastie (1992): Statistical Models in S, A.K.A. the White Book.



FROM S TO R: GREEN BOOK

Chambers (1998): Programming with Data: A Guide to the S Language, A.K.A. the Green Book.
Venables and Ripley (2000): S Programming

S2 TO S3 TO S4

- The S2 system was described in the Brown Book, S3 in the Blue Book and completed in the White Book, finally S4 in the Green Book
- The big advance from S2 to S3 was that users could write functions; that data.frame objects were defined; that formula objects were defined; and that S3 classes and method dispatch appeared
- S4 brought connections and formal S4 classes, the latter seen in R in the **methods** package (still controversial)
- S-PLUS was/is the commercial implementation of S and its releases drove S3 and S4 changes

S, BELL LABS, S-PLUS

- S was a Bell Labs innovation, like Unix, C, C++, and many interpreted languages (like AWK); many of these share key understandings
- Now owned by Nokia, previously Alcatel-Lucent, Lucent, and AT&T
- Why would a telecoms major (AT&T) pay for fundamental research in computer science and data analysis (not to sell or market other products better)?
- Some Green Book examples are for quality control of telecoms components

S-PLUS AND R

- S-PLUS was quickly adopted for teaching and research, and with S3, provided extensibility in the form of libraries
- Most links have died by now, but see this FAQ for a flavour - there was a lively community of applied statisticians during the 1990s
- S built on a long tradition of documentation through examples, with use cases and data sets taken from the applied statistical literature; this let users compare output with methods descriptions
- … so we get to R

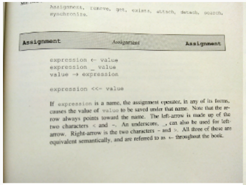
AND WHAT ABOUT LISPSTAT?

- Luke Tierney was in R core in 1997, and has continued to exert clear influence over development
- Because R uses a Scheme engine, similar to Lisp, under the hood, his insight into issues like the garbage collector, namespaces, byte-compilation, serialization, parallelization, and now ALTREP has been crucial (see also the proposal by Luke Tierney, Gabe Becker and Tomas Kalibera)
- Many of these issues involve the defensive copy on possible change policy involved in lazy evaluation, which may lead to multiple redundant copies of data being present in memory
- Luke Tierney and Brian Ripley have fought hard to let R load fast, something that is crucial to ease the use of R on multicore systems or inside databases

Vintage R

USE QUESTIONS: LEFT ASSIGN

Why was `underscore_separated` not a permitted naming convention in R earlier (see Bååth (2012))? `_` was not a permitted character in names until it had lost its left assign role, the same as `<-`, in 1.9.0 in 2004. (Brown Book p. 256, Blue Book p. 387)



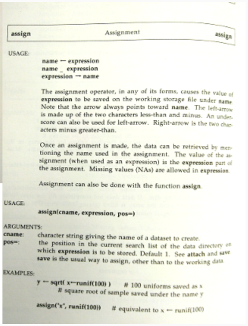
Assignment, remove, get, exists, attach, detach, search, synchronize.

Assignment Assignment **Assignment**

```
expression ← value
expression _ value
value → expression

expression <<- value
```

If `expression` is a name, the assignment operator, in any of its forms, causes the value of `value` to be saved under that name. Note that the arrow always points toward the name. The left-arrow is made up of the two characters < and -. An underscore, _, can also be used for left-arrow. Right-arrow is the two characters - and >. All three of these are equivalent semantically, and are referred to as ← throughout the book.



assign Assignment **assign**

USAGE:

name ← expression
name _ expression
expression → name

The assignment operator, in any of its forms, causes the value of **expression** to be saved on the working storage file under **name**. Note that the arrow always points toward **name**. The left-arrow is made up of the two characters less-than and minus. An underscore can also be used for left-arrow. Right-arrow is the two characters minus greater-than.

Once an assignment is made, the data can be retrieved by mentioning the name used in the assignment. The value of the assignment (when used as an expression) is the **expression** part of the assignment. Missing values (NAs) are allowed in **expression**.

Assignment can also be done with the function **assign**.

USAGE:

assign(cname, expression, pos=)

ARGUMENTS:

cname:	character string giving the name of a dataset to create.
pos=:	the position in the current search list of the data directory on which **expression** is to be stored. Default 1. See **attach** and **save** **save** is the usual way to assign, other than to the working data.

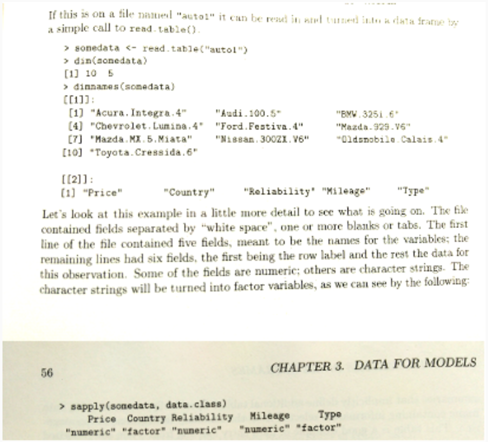
EXAMPLES:

```
y ← sqrt( x←runif(100) )      # 100 uniforms saved as x
          # square root of sample saved under the name y

assign("x", runif(100))    # equivalent to x ← runif(100)
```

USE QUESTIONS: `stringsAsFactors`

Why is the `factor` storage mode still so central? `stringsAsFactors = TRUE` was the legacy `as.is = FALSE`; analysis of categorical variables was more important, and `factor` only needed to store `nlevels()` strings (White Book p. 55-56, 567)



If this is on a file named "auto1" it can be read in and turned into a data frame by a simple call to read.table().

```
> somedata <- read.table("auto1")
> dim(somedata)
[1] 10  5
> dimnames(somedata)
[[1]]:
 [1] "Acura.Integra.4"     "Audi.100.5"          "BMW.325i.6"
 [4] "Chevrolet.Lumina.4"  "Ford.Festiva.4"      "Mazda.929.V6"
 [7] "Mazda.MX.5.Miata"    "Nissan.300ZX.V6"     "Oldsmobile.Calais.4"
[10] "Toyota.Cressida.6"

[[2]]:
[1] "Price"       "Country"     "Reliability" "Mileage"     "Type"
```

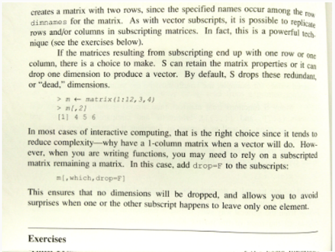
Let's look at this example in a little more detail to see what is going on. The file contained fields separated by "white space", one or more blanks or tabs. The first line of the file contained five fields, meant to be the names for the variables; the remaining lines had six fields, the first being the row label and the rest the data for this observation. Some of the fields are numeric; others are character strings. The character strings will be turned into factor variables, as we can see by the following:

56 *CHAPTER 3. DATA FOR MODELS*

```
> sapply(somedata, data.class)
     Price   Country Reliability   Mileage      Type
 "numeric"  "factor"   "numeric" "numeric"  "factor"
```

USE QUESTIONS: `drop`

`drop = TRUE` for array-like objects; since matrices are vectors with a `dim` atrribute, choosing (part of) a row or column made `dim` redundant (Blue Book p. 128, White Book p. 64)



creates a matrix with two rows, since the specified names occur among the row `dimnames` for the matrix. As with vector subscripts, it is possible to replicate rows and/or columns in subscripting matrices. In fact, this is a powerful technique (see the exercises below).

If the matrices resulting from subscripting end up with one row or one column, there is a choice to make. S can retain the matrix properties or it can drop one dimension to produce a vector. By default, S drops these redundant, or "dead," dimensions.

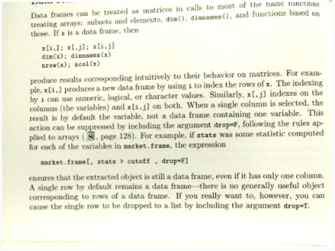
```
> m ← matrix(1:12,3,4)
> m[,2]
[1] 4 5 6
```

In most cases of interactive computing, that is the right choice since it tends to reduce complexity—why have a 1-column matrix when a vector will do. However, when you are writing functions, you may need to rely on a subscripted matrix remaining a matrix. In this case, add `drop=F` to the subscripts:

```
m[,which,drop=F]
```

This ensures that no dimensions will be dropped, and allows you to avoid surprises when one or the other subscript happens to leave only one element.

Exercises



calls to most of the basic functions

Data frames can be treated as matrices in calls to most of the basic functions treating arrays: subsets and elements, `dim()`, `dimnames()`, and functions based on those. If `x` is a data frame, then

```
x[i,]; x[,j]; x[i,j]
dim(x); dimnames(x)
nrow(x); ncol(x)
```

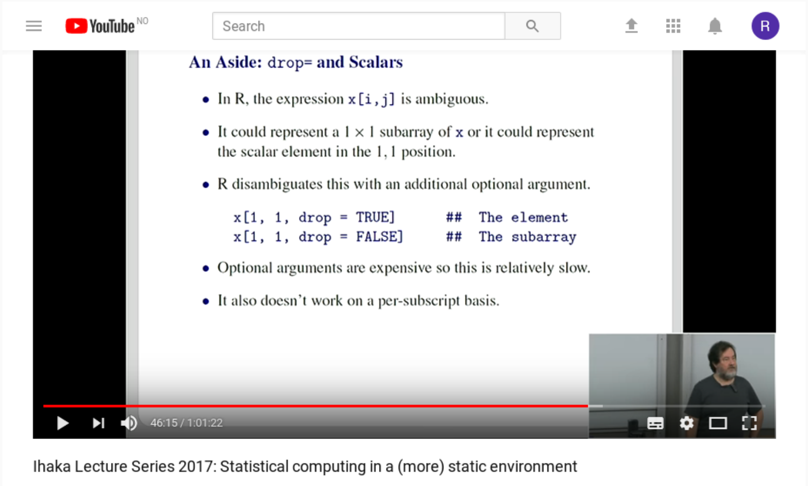
produce results corresponding intuitively to their behavior on matrices. For example, `x[i,]` produces a new data frame by using `i` to index the rows of `x`. The indexing by `i` can use numeric, logical, or character values. Similarly, `x[,j]` indexes on the columns (the variables) and `x[i,j]` on both. When a single column is selected, the result is by default the variable, not a data frame containing one variable. This action can be suppressed by including the argument `drop=F`, following the rules applied to arrays (S , page 128). For example, if `stats` was some statistic computed for each of the variables in `market.frame`, the expression

```
market.frame[, stats > cutoff , drop=F]
```

ensures that the extracted object is still a data frame, even if it has only one column. A single row by default remains a data frame—there is no generally useful object corresponding to rows of a data frame. If you really want to, however, you can cause the single row to be dropped to a list by including the argument `drop=T`.

BUT SCALARS ARE ALSO VECTORS ...

Treating scalars as vectors is not efficient:



An Aside: drop= and Scalars

- In R, the expression x[i,j] is ambiguous.
- It could represent a 1×1 subarray of x or it could represent the scalar element in the $1, 1$ position.
- R disambiguates this with an additional optional argument.

```
x[1, 1, drop = TRUE]     ##  The element
x[1, 1, drop = FALSE]    ##  The subarray
```

- Optional arguments are expensive so this is relatively slow.
- It also doesn't work on a per-subscript basis.

Ihaka Lecture Series 2017: Statistical computing in a (more) static environment

VINTAGE R

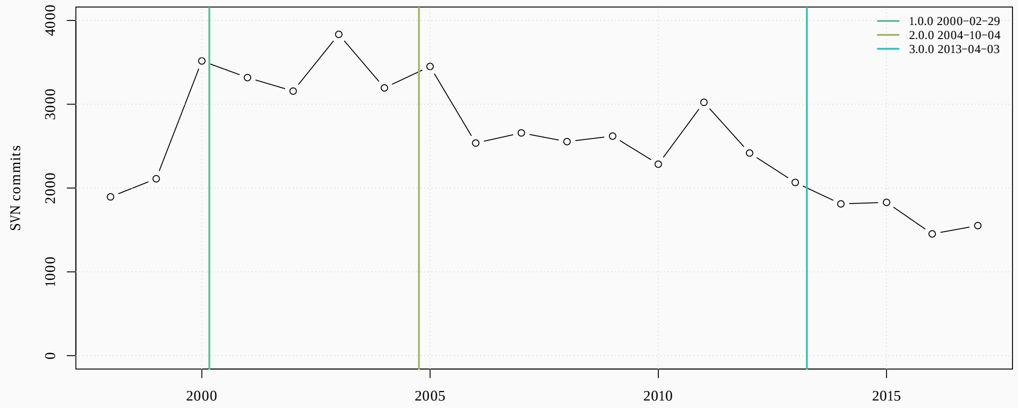
- An R-0.49 source tarball is available from CRAN
- Diffs for Fedora 27 (gcc 7.3.1) include setting compilers and `-fPIC` in `config.site`, putting `./` before `config.site` in `configure`, and three corrections in `src/unix`: in `dataentry.h` add `#include <X11/Xfuncproto.h>` and comment out `NeedFunctionPrototypes`; in `rotated.c` comment out `/*static*/ double round`; in `system.c` comment out `__setfpucw` twice; BLAS must be provided externally
- Not (yet) working: prototypes are missing in the **eda** and **mva** packages so the shared objects fail to build
- R-0.49 video

R SVN logs

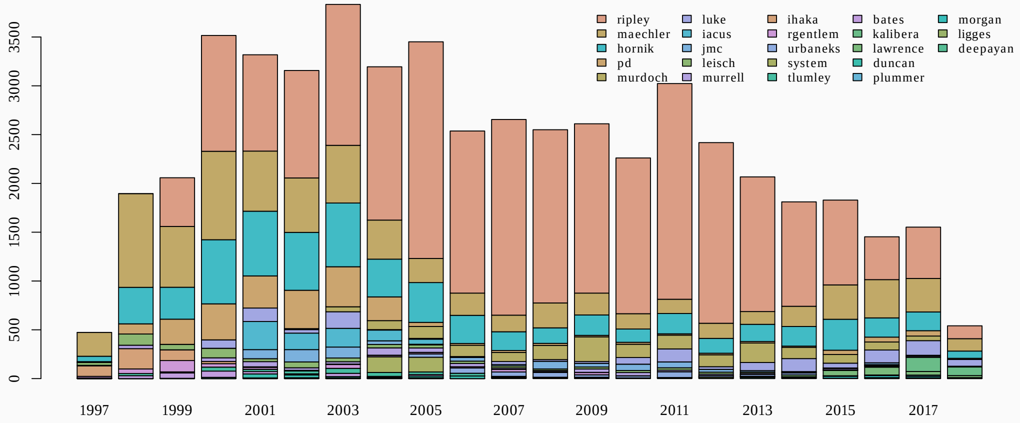
R SVN LOGS

- The command: `svn log --xml --verbose -r 6:74688 https://svn.r-project.org/R/trunk > trunk_verbose_log1.xml` provides a rich data source
- Each log entry has a revision number, author and timestamp, message and paths to files indicating the action undertaken for each file
- The XML version is somewhat easier to untangle than the plain-text version
- I haven't tried possible similar approaches to Winston Chang's github r-source repo

COMMITS 1998–2017



COMMITS BY AUTHOR AND YEAR BETWEEN R6 AND R74688



XML LOGENTRY STRUCTURE

```
## <logentry revision=”6”>
##  <author>ihaka</author>
##  <date>1997-09-18T04:41:25.000000Z</date>
##  <paths>
##   <path action=”M” prop-mods=”false” text-mods=”true” kind=”file”>/t
##  </paths>
##  <msg>New predict.lm from Peter Dalgaard</msg>
## </logentry>
```

XML LOGENTRY STRUCTURE

```
## <logentry revision="74688">
##  <author>ripley</author>
##  <date>2018-05-03T11:11:00.501520Z</date>
##  <paths>
##   <path text-mods="true" kind="file" action="M" prop-mods="false">/t
##  </paths>
##  <msg>TexLive 2018 may produce logs not in the current encoding</msg
## </logentry>
```

COMMIT MESSAGES BY NUMBER OF FILES AFFECTED, YEAR AND REVISION

```
## 2015 68948 2150 use https
## 2012 59039 1727 use preferred form of ’R Core Team’
## 2011 56186 1260 Revert r56184 and r56185
## 2011 56184 1249 Remove redundant \alias entries from man pages
## 2007 42333 1223 add copyright/licence header, remove CVS-style $Id f
## 2012 61433 620 remove trailing spaces
## 2012 60146 602 add copyright statements
## 2007 42338 559 add licence statements
## 2012 59780 524 update, including bug-reporting address
## 2003 27444 497 splitting base
```

EPOCH FILE COMMITS IN TRUNK/

```
##
##        tools          FAQ           m4          etc
##          396          467          579          601
## configure.ac        share    configure         BUGS
##          693         1085         1319         1485
##   date-stamp           po         NEWS        tests
##         2531         3762         5842        11445
##          doc          src
##        11484        97067
```

EPOCH FILE COMMITS IN TRUNK/SRC

```
##
##    windows   graphics      gnome macintosh       appl       unix
##         74         79        217       611        796       1524
##      nmath    scripts      extra   modules    include   gnuwin32
##       1764       1966       2315      2358       3528       9140
##       main    library
##      15011      57563
```

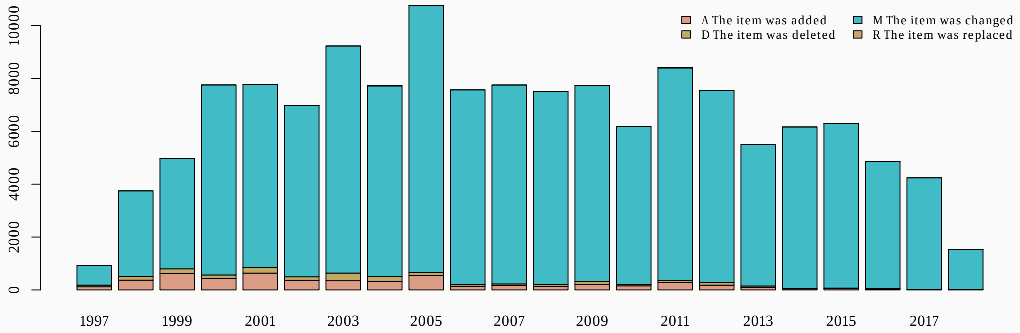
EPOCH FILE COMMITS IN TRUNK/SRC/LIBRARY

```
##
##     compiler      profile       stats4 translations
##          234          247          261          296
##          nls     datasets       modreg          mva
##          319          333          402          462
##      splines        ctest        tcltk           ts
##          464          549          865          900
##     parallel         grid     graphics    grDevices
##         1096         1931         2158         3695
##      methods        utils        stats        tools
##         3982         5397         6840         7028
##         base
##        19331
```

EPOCH FILE COMMITS IN TRUNK/SRC/LIBRARY/BASE

```
##
##       Makefile DESCRIPTION.in   makebasedb.R   Makefile.win
##              8             10             11             18
##   baseloader.R           demo           data    Makefile.in
##             32             61             66             75
##           inst             po              R            man
##            270            497           6719          11557
```

FILES BY YEAR AND COMMIT ACTION



CRAN and Bioconductor packages

CRAN

- Once S3 permitted extension by writing functions, and packaging functions in libraries, S and R ceased to be monolithic
- In R, a library is where packages are kept, distinguishing between base and recommended packages distributed with R, and contributed packages
- Contributed packages can be installed from CRAN (infrastructure built on CPAN and CTAN for Perl and Tex), Bioconductor, other package repositories, and other sources such as github
- With over 12000 contributed packages, CRAN is central to the R community, but is stressed by dependency issues (CRAN is not run by R core)

CRAN/BIOCONDUCTOR PACKAGE CLUSTERS

- Andrie de Vries Finding clusters of CRAN packages using igraph looked at CRAN package clusters from a page rank graph
- We are over three years further on now, so updating may be informative
- However, this is only CRAN, and there is the big Bioconductor repository to consider too
- Adding in the Bioconductor (S4, curated) repo does alter the optics, as you'll see, over and above the cluster dominated by **Rcpp**

CRAN/BIOCONDUCTOR PACKAGE PAGE RANK SCORES

```
##           Rcpp          MASS        ggplot2 AnnotationDbi
##       0.022891      0.012087       0.011861      0.011142
##         Matrix         dplyr          plyr        mvtnorm
##       0.006958      0.006744       0.005359      0.005113
##        Biobase      survival        stringr    data.table
##       0.005049      0.005023       0.004682      0.004223
##        lattice        igraph RcppArmadillo          httr
##       0.004112      0.003993       0.003944      0.003898
##       magrittr      jsonlite        IRanges      reshape2
##       0.003896      0.003831       0.003661      0.003515
```

FIRST PACKAGE CLUSTER

```
##         MASS      ggplot2         plyr      mvtnorm
##  0.012086924  0.011860517  0.005359345  0.005113266
##     survival      lattice       igraph     reshape2
##  0.005022831  0.004111811  0.003992951  0.003515287
##      foreach           sp RColorBrewer   doParallel
##  0.003309872  0.003280003  0.003092672  0.002297795
```

SECOND PACKAGE CLUSTER

```
##       dplyr     stringr        httr    magrittr    jsonlite
## 0.006744351 0.004681562 0.003898281 0.003895957 0.003830517
##       shiny       tidyr      tibble         XML       RCurl
## 0.002985308 0.002493951 0.002476508 0.002418055 0.002164565
##       purrr   lubridate
## 0.001852909 0.001549448
```

Third package cluster

```
##             Biobase               IRanges             S4Vectors
##         0.005048914           0.003661421           0.003473939
##       GenomicRanges            Biostrings          BiocGenerics
##         0.002733640           0.002583678           0.002437776
##                 DBI               RSQLite                 limma
##         0.002257874           0.001849297           0.001582772
##        GenomeInfoDb SummarizedExperiment          oligoClasses
##         0.001459829           0.001450142           0.001209420
```

FOURTH PACKAGE CLUSTER

```
##           Rcpp          Matrix     data.table RcppArmadillo
##   0.0228907439    0.0069578788   0.0042227075  0.0039442804
##      RcppEigen              BH          rstan  RcppParallel
##   0.0011843649    0.0011622371   0.0004800493  0.0004186803
##      bigmemory          Rdpack          magic      RSpectra
##   0.0003684402    0.0003410619   0.0002661583  0.0002060934
```

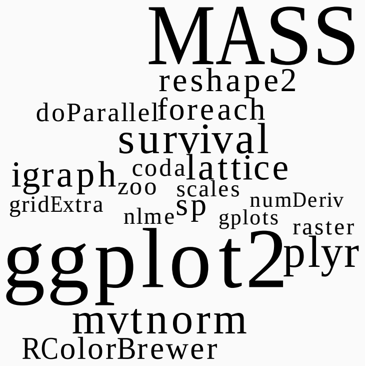
FIFTH PACKAGE CLUSTER

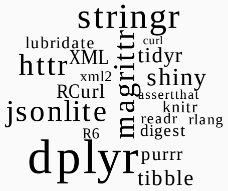
```
##    AnnotationDbi    org.Hs.eg.db GenomicFeatures    org.Mm.eg.db
##     1.114150e-02    1.233367e-03    1.098628e-03    6.799030e-04
##     org.Rn.eg.db     geneXtendeR          ChIPQC            rCGH
##     4.536656e-04    2.306122e-04    1.574986e-04    1.357878e-04
##          chimera    Homo.sapiens     OrganismDbi TxDb.Hsapiens.U
##     1.335513e-04    1.280833e-04    1.186413e-04    9.027467e-05
```

SIXTH PACKAGE CLUSTER

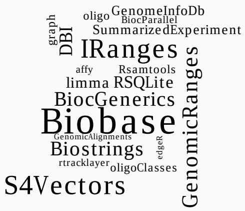
```
##       Formula     sandwich       lmtest       texreg
## 0.0010315610 0.0006274957 0.0005497347 0.0003765079
##        maxLik   prediction    stargazer     partykit
## 0.0003358931 0.0003106929 0.0002972425 0.0002900618
##     penalized    miscTools          AER          plm
## 0.0002295456 0.0002291684 0.0002026993 0.0001901027
```

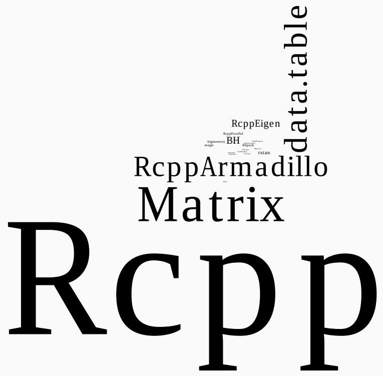
CRAN/Bioconductor top two page rank clusters



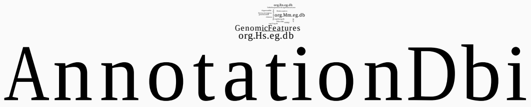


CRAN/BIOCONDUCTOR THIRD AND FOURTH PAGE RANK CLUSTERS





CRAN/BIOCONDUCTOR FIFTH AND SIXTH PAGE RANK CLUSTERS



Formula
sandwich
prediction
texreg
partykit
lmtest
stargazer

CRAN/BIOCONDUCTOR PACKAGE AUTHOR CLUSTERS

- Francois Keck explored CRAN package co-authorship in a more recent blog: Exploring the CRAN social network
- Once again, a little time has passed, so maybe things have shifted
- Thanks to Martin Morgan, I've added listings corresponding in part to `tools::CRAN_package_db()`
- It is refreshing to see that Bioconductor is clearly present, and the people implicated are active in upgrading R internals

FIRST TWO PACKAGE AUTHOR CLUSTERS

```
##                  Name Package
## 1          Kurt Hornik      62
## 2      Martin Maechler      52
## 3        Achim Zeileis      49
## 4   Dirk Eddelbuettel      48
## 5         Brian Ripley      35
## 6      Romain Francois      28
## 7     Torsten Hothorn       28
## 8           Ben Bolker      27
## 9        Douglas Bates      26
## 10        Roger Bivand      25
## 11       Thomas Lumley      24
## 12    Michael Friendly      21
```

```
##                  Name Package
## 1              Rstudio     110
## 2       Hadley Wickham     107
## 3                  Inc      50
## 4   Scott Chamberlain       49
## 5          Jeroen Ooms      40
## 6        R. Core Team       38
## 7            Yihui Xie      36
## 8           Bob Rudis       32
## 9         Jj Allaire        31
## 10      Kirill Muller       31
## 11       Gabor Csardi       30
## 12     Winston Chang        27
```

THIRD AND FOURTH

```
##                        Name Package
## 1  Bioconductor Package      37
## 2          Marc Carlson      29
## 3         Martin Morgan      28
## 4           Herve Pages      22
## 5  Bioconductor Core Te      19
## 6           Seth Falcon      15
## 7       Lihua Julie Zhu      13
## 8             Aaron Lun      12
## 9        Pierre Neuvial      11
## 10         Gordon Smyth      10
## 11    Valerie Obenchain      10
## 12          Jianhong Ou       9
```

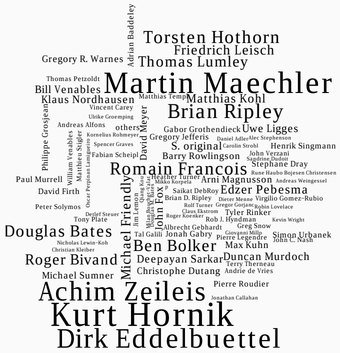
```
##                        Name Package
## 1         Wolfgang Huber      32
## 2      Robert Gentleman      28
## 3     Rafael A. Irizarry      28
## 4   Kasper Daniel Hansen      20
## 5      Matthew N. McCall      20
## 6   Hector Corrada Bravo      15
## 7             Andy Lynch      14
## 8           Mark Dunning      13
## 9         John D. Storey      12
## 10       Andrew E. Jaffe      11
## 11       Jeffrey T. Leek      11
## 12     Matthew Eldridge      10
```

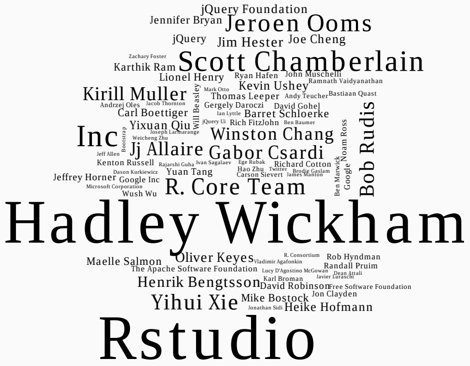
FIFTH AND SIXTH

```
##                      Name Package
## 1   Benjamin Haibe-Kains      21
## 2      Gianluca Bontempi      18
## 3       John Quackenbush      12
## 4           Levi Waldron      12
## 5           Marcel Ramos      10
## 6          Aedin Culhane       9
## 7        Catharina Olsen       9
## 8       Markus Schroeder       8
## 9      Christos Sotiriou       7
## 10    Ludwig Geistlinger       7
## 11     Deena M.A. Gendoo       6
## 12      Houtan Noushmehr       5
```

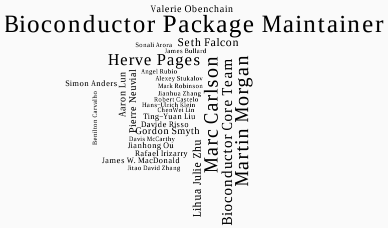
```
##                      Name Package
## 1            R. Gentleman      18
## 2        Raphael Gottardo      14
## 3            Arnaud Droit      13
## 4              Greg Finak      10
## 5                H. Pages       7
## 6              Mike Jiang       7
## 7        Astrid Deschenes       6
## 8          Pascal Belleau       6
## 9                F. Hahne       6
## 10              S. Falcon       6
## 11 Charles Joly Beaupar       5
## 12      N. Gopalakrishnan       5
```

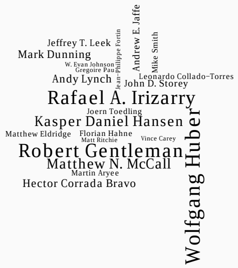
FIRST TWO PACKAGE AUTHOR CLUSTERS



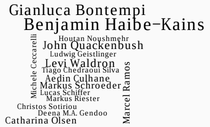


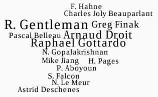
THIRD AND FOURTH PACKAGE AUTHOR CLUSTERS





FIFTH AND SIXTH PACKAGE AUTHOR CLUSTERS





ROUNDUP: HISTORY

- Many sources in applied statistics with an S-like syntax but Lisp/Scheme-like internals, and sustained tensions between these
- Many different opinions on prefered ways of structuring data and data handling, opening for adaptations to different settings
- More recently larger commercial interest in handling large input long data sets, previously also present; simulations also generate large output data sets; bioinformatics both wide and long
- Differing views of the world in terms of goals and approaches
- Differences provide ecological robustness