Searching help pages of R packages

by Spencer Graves, Sundar Dorai-Raj, and Romain François

Abstract The **sos** package provides a means to quickly and flexibly search the help pages of contributed packages, finding functions and datasets in seconds or minutes that could not be found in hours or days by any other means we know. Its findFn function accesses Jonathan Baron's R Site Search database and returns the matches in a data frame of class "findFn", which can be further manipulated by other sos functions to produce, for example, an Excel file that starts with a summary sheet that makes it relatively easy to prioritize alternative packages for further study. As such, it provides a very powerful way to do a literature search for functions and packages relevant to a particular topic of interest and could become virtually mandatory for authors of new packages or papers in publications such as The R Journal and the Journal of Statistical Software.

Introduction

The **sos** package provides a means to quickly and flexibly search the help pages of contributed packages, finding functions and datasets in seconds or minutes that could not be found in hours or days by any other means we know.

The main capability of this package is the findFn function, which scans the "function" entries in Jonathan Baron's *R site search* database (Baron, 2009) and returns the matches in a data frame of class "findFn". Baron's site is one of five search capabilities currently identified under "Search" from the main http://www.r-project.org/ web site. It includes options to search the help pages of R packages contributed to CRAN (the Comprehensive R Archive Network) plus a few other publicly available packages, as well as selected mailing list archives primarily R-help. The findFn function focuses only on the help pages in this database. (CRAN grew from 1700 contributed packages and bundles on 2009-03-11 to 1954 on 2009-09-18, adding over 40 packages per day, an annual growth rate of 31 percent.)

The print method for objects of class "findFn" displays the results as a table in a web browser with links to the individual help pages, sorted by package, displaying the results from the package with the most matches first. This behaviour differs from that of the RSiteSearch function in the utils package in more ways than the sort order. First, findFn returns the results in R as a data frame, which can be further manipulated. Second, the ultimate display in a web browser is a table, unlike the list produced by

RSiteSearch.

Other sos functions provide summaries with one line for each package, support the union and intersection of "findFn" objects, and translate a "findFn" object into an Excel file with three sheets: (1) PackageSum2, which provides an enhanced summary of the packages with matches, (2) the findFn table itself, and (3) the call used to produce it.

Three examples are considered below: First we find a data set containing a variable Petal.Length. Second, we study R capabilities for splines, including looking for a function named spline. Third, we search for contributed R packages with capabilities for solving differential equations.

Finding a variable in a data set

Chambers (2009, pp. 282-283) uses a variable Petal.Length from a famous Fisher data set but without naming the data set nor indicating where it can be found nor even if it exists in any contributed R package. The sample code he provides does not work by itself. To get his code to work to produce his Figure 7.2, we must first obtain a copy of this famous data set in a format compatible with his code.

To look for this data set, one might first try the help.search function. Unfortunately, this function returns nothing in this case:

```
> help.search('Petal.Length')
```

No help files found with alias or concept or title matching 'Petal.Length' using regular expression matching.

When this failed, many users might then try

```
> RSiteSearch('Petal.Length')
```

A search query has been submitted to http://search.r-project.or The results page should open in your browser shortly

This produced 80 matches when it was tried one day (and 62 matches a few months later). RSiteSearch('Petal.Length', 'function') will identify only the help pages. We can get something similar and for many purposes more useful, as follows:

```
> library(sos)
> PL <- findFn('Petal.Length')
found 65 matches; retrieving 4 pages
2 3 4</pre>
```

PL is a data frame of class "findFn" identifying all the help pages in Jonathan Baron's data base matching the search term. An alias for findFn is ???, and this same search can be performed as follows:

```
> PL <- ???Petal.Length
```

```
found 65 matches; retrieving 4 pages
2 3 4
```

(The ??? alias only works in an assignment, so to print immediately, you need something like (PL <-???Petal.Length).)

This data frame has columns Count, MaxScore, TotalScore, Package, Function, Date, Score, Description, and Link. Function is the name of the help page, not the name of the function, as multiple functions may be documented on a single help page, and some help pages document other things such as data sets. Score is the index of the strength of the match. It is used by Baron's search engine to decide which items to display first. Package is the name of the package containing Function. Count gives the total number of matches in Package found in this findFn call. By default, the findFn object is sorted by Count, MaxScore, TotalScore, and Package (to place the most important Package first), then by Score and Function.

The summary method for such an object prints a table giving for each Package the Count (number of matches), MaxScore (max of Score), TotalScore (sum of Score), and Date, sorted like a Pareto chart to place the Package with the most help pages first:

(The Date here is the date that this package was added to Baron's database.)

One of the listed packages is **datasets**. Since it is part of the default R distribution, we decide to look there first. We can select that row of PL just like we would select a row from any other data frame:

```
> PL[PL$Package == 'datasets', 'Function']
[1] "iris"
```

Problem solved in less than a minute! Any other method known to the present authors would have taken substantially more time.

Finding packages with spline capabilities

Almost four years ago, the lead author of this article decided he needed to learn more about splines. A literature search began as follows:

```
> RSiteSearch('spline')
```

(using the RSiteSearch function in the **utils** package). While preparing this manuscript, this command identified 1372 documents as of 2011-07-03. That is too many. It can be restricted to functions as follows:

```
> RSiteSearch('spline', 'fun')
```

This identified only 1169 one day (631 a few months earlier). That's an improvement over 1372 but is still too many. To get a quick overview of these 1169, we can proceed as follows:

```
> splinePacs <- findFn('spline')</pre>
```

This downloaded a summary of the 0 highest-scoring help pages in the 'RSiteSearch' data base in roughly 5-15 seconds, depending on the speed of the Internet connection. To get all 1169 matches, increase the maxPages argument from its default 20:

```
> splineAll <- findFn('spline', maxPages = 999)</pre>
```

If we want to find a function named spline, we can proceed as follows:

```
> selSpl <- splineAll[, 'Function'] == 'spline'
> splineAll[selSpl, ]
```

This has 0 rows, because there is no help page named spline. This does not mean that no function with that exact name exists, only that no help page has that name.

To look for help pages whose name includes the characters 'spline', we can use grepFn:

```
> grepFn('spline', splineAll, ignore.case = TRUE)
```

This returned a "findfn" object identifying 111 help pages. When this was run while preparing this manuscript, the sixth row was inprod.Bspline in the fda package, which has a Score of 26. (On another day, the results could be different, because CRAN changes over time.) This was the sixth row in this table, because it is in the fda package, which had a total of 36 help pages matching the search term, but this was the only one whose name matched the pattern passed to grepfn.

We could next print the splineAll "findFn" object. However, it may not be easy to digest a table with 111 rows.

summary (splineAll) would tell us that the 1057 help pages came from 0 different packages and display the first minPackages = 12 such packages. (If other packages had the same number of matches as

the twelfth package, they would also appear in this summary.)

A more complete view can be obtained in MS Excel format using the writeFindFn2xls function:

```
> writeFindFn2xls(splineAll)
```

(findFn2xls is an alias for writeFindFn2xls. We use the longer version here, as it may be easier to remember.)

If either the **WriteXLS** package and compatible Perl code are properly installed or if you are running Windows with the **RODBC** package, this produces an Excel file in the working directory named 'splineAll.xls', containing the following three worksheets:

- The PackageSum2 sheet includes one line for each package with a matching help page, enhanced by providing information for locally installed packages not available in the "findFn" object.
- The **findFn** sheet contains the search results.
- The **call** sheet gives the call to findFn that generated these search results.

If writeFindFn2xls cannot produce an Excel file with your installation, it will write three 'csv' files with names 'splineAll-sum.csv', 'splineAll.csv', and 'splineAll-call.csv', corresponding to the three worksheets described above. (Users who do not have MS Excel may like to know that Open Office Calc can open a standard 'xls' file and can similarly create such files (Openoffice.org, 2009).)

The PackageSum2 sheet is created by the PackageSum2 function, which adds information from installed packages not obtained by findFn. The extended summary includes the package title and date, plus the names of the author and the maintainer, the number of help pages in the package, and the name(s) of any vignettes. This can be quite valuable in prioritizing packages for further study. Other things being equal, we think most people would rather learn how to use a package being actively maintained than one that has not changed in five years. Similarly, we might prefer to study a capability in a larger package than a smaller one, because the rest of the package might provide other useful tools or a broader context for understanding the capability of interest.

These extra fields, package title, etc., are blank for packages in the "findFn" object not installed locally. For installed packages, the Date refers to the *packaged* date rather than the date the package was added to Baron's database.

Therefore, the value of PackageSum2 can be increased by running install.packages (from the

utils package) to install packages not currently available locally and update.packages() to ensure the local availability of the latest versions for all installed packages.

To make it easier to add desired packages, the sos package includes an installPackages function, which checks all the packages in a "findFn" object for which the number of matches exceeds a second argument minCount and installs any of those not already available locally; the default minCount is the square root of the largest Count. Therefore, the results from PackageSum2 and the PackageSum2 sheet created by writeFindFn2xls will typically contain more information after running installPackages than before.

To summarize, three lines of code gave us a very powerful summary of spline capabilities in contributed R packages:

```
> splineAll <- findFn('spline', maxPages = 999)
> # Do not include in auto test
> #installPackages(splineAll)
> writeFindFn2xls(splineAll)
```

The resulting 'splineAll.xls' file can help establish priorities for further study of the different packages and functions. An analysis of this nature almost four years ago led the lead author to the **fda** package and its companion books, which further led to a collaboration that has produced joint presentations at three different conferences and a joint book (Ramsay et al., 2009).

Combining search results

The lead author of this article recently gave an invited presentation on "Fitting Nonlinear Differential Equations to Data in R''^1 . A key part of preparing for that presentation was a search of contributed R code, which proceeded roughly as follows:

```
> de <- findFn('differential equation')
> des <- findFn('differential equations')
> de. <- de | des</pre>
```

The object de has 131 rows, while des has 193. If this search engine were simply searching for character strings, de would be larger than des, rather than the other way around. The last object de. is the union of de and des; '|' is an alias for unionFindFn. The de. object has 262 rows, which suggests that the corresponding intersection must have (131 + 193 - 262 = 62). This can be confirmed via nrow (de & des). ('&' is an alias for intersectFindFn.)

To make everthing in de. locally available, we can use installPackages (de., minCount = 1). This installed all referenced packages except rmutil and a dependency Biobase, which were not available on

¹Workshop on Statistical Methods for Dynamic System Models, Vancouver, 2009: http://stat.sfu.ca/~dac5/workshop09/Spencer_Graves.html

DISCUSSION BIBLIOGRAPHY

CRAN but are included in Jonathan Baron's R Site Search data base.

Next, writeFindFn2xls(de.) produced a file 'de..xls' in the working directory. (The working directory can be identified via getwd().)

The PackageSum2 sheet of that Excel file provided a quick summary of packages with matches, sorted to put the package with the most matches first. In this case, this first package was deSolve, which provides, "General solvers for initial value problems of ordinary differential equations (ODE), partial differential equations (PDE) and differential algebraic equations (DAE)". This is clearly quite relevant to the subject. The second package was PKfit, which is "A Data Analysis Tool for Pharmacokinetics". This may be too specialized for general use. I therefore would not want to study this first unless my primary interest here was in pharmacokinetic models.

By studying the summary page in this way, I was able to decide relatively quickly which packages I should consider first. In making this decision, I gave more weight to packages with one or more vignettes and less weight to those where the Date was old, indicating that the code was not being actively maintained and updated. I also checked the conference information to make sure I did not embarrass myself by overlooking a package authored or maintained by another invited speaker.

Discussion

We have found findFn in the sos package to be very quick, efficient and effective for finding things in contributed packages. The grepFn function helps quickly look for functions (or help pages) with particular names. The capabilities in unionFindFn and intersectFindFn (especially via their '|' and '&' aliases) can be quite useful where a single search term seems inadequate; they make it easy to combine multiple searches to produce something closer to what is desired. An example of this was provided with searching for both "differential equation" and "differential equations".

The PackageSum2 sheet of an Excel file produced by writeFindFn2xls (after also running the installPackages function) is quite valuable for understanding the general capabilities available for a particular topic. This could be of great value for authors to find what is already available so they don't duplicate something that already exists and so their new contributions appropriately consider the contents of other packages.

The findFn capability can also reduce the risk of "the researcher's nightmare" of being told after substantial work that someone else has already done it.

Users of **sos** may also wish to consult Crantastic (http://www.crantastic.org/), which allows users to tag, rate, and view packages. (The coverage

of Crantastic includes current and orphaned CRAN packages, while Baron (2009) also includes 'most of the default packages from Bioconductor and all of Jim Lindsey's packages.')

Acknowledgments

The capabilities described here extend the power of the R Site Search search engine maintained by Jonathan Baron. Without Prof. Baron's support, it would not have been feasible to develop the features described here. Duncan Murdoch, Marc Schwarz, Dirk Eddelbuettel, Gabor Grothendiek and anonymous referees contributed suggestions for improvement, but of course can not be blamed for any deficiencies. The collaboration required to produce the current sos package was greatly facilitated by R-Forge (R-Forge Team, 2009). The sos package is part of the R Site Search project hosted there. This project also includes code for a Firefox extension to simplify the process of finding information about R from within Firefox. This Firefox extension is still being developed with the current version downloadable from http://addictedtor.free.fr/rsitesearch.

Bibliography

- J. Baron. R site search. URL http://finzi.psych. upenn.edu/search.html, September 2009.
- J. Chambers. *Software for Data Analysis: Programming with R.* Springer, New York, 2009.
- Openoffice.org. *Open Office Calc.* Sun Microsystems, California, USA, 2009. URL http://www.openoffice.org.
- J. Ramsay, G. Hooker, and S. Graves. Functional Data Analysis with R and MATLAB. Springer, New York, 2009.
- S. Theußl and A. Zeileis "Collaborative software development using R-Forge". *The R Journal*. 1(1):9–14.

Spencer Graves
Structure Inspection and Monitoring
San Jose, CA
spencer.graves@prodsyse.com

Sundar Dorai-Raj

Google
Mountain View, CA
sdorairaj@google.com

Romain François Independent R Consultant Montpellier, France francoisromain@free.fr