Searching help pages of R packages

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

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 and returns the matches in a data.frame of class findFn (Baron, 2009). Baron's is one of five search capabilities currently identified under "search" from the main 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.

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 one with the most matches first. This is different from the RSiteSearch function, as findFn returns the results in R as a data.frame, which can be further manipulated, and 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 write the results to 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 dataset 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: Petal.Length

Chambers (2009, pp. 282-283) uses a variable Petal.Length from a famous Fisher data set but without naming the dataset 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 ...
```

When this failed, many users might then try RSiteSearch('Petal.Length'). 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 purpose more useful as follows:

```
> library(sos)
> PL <- findFn('Petal.Length')
```

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
```

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 following table has been
> # manually edited for clarity
> summary(PL)

Total number of matches: 27
Downloaded 27 links in 14 packages.
Packages with at least 1 match using search
                     pattern 'Petal.Length':
Package Count MaxScore TotalScore       Date
yaImpute    8        1          8 2009-08-16
<...>
datasets    1        2          2 2009-07-09
<...>
```

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 1526 documents one day. That is too many. It can be restricted to functions as follows:

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

This identified only 739 one day (631 a few months earlier). That's an improvement over 1526 but is still too many. To get a quick overview of these 739, we can proceed as follows:

```
splinePacs <- findFn('spline')
```

This downloaded a summary of the 400 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 739 matches, increase the `maxPages` argument from its default 20:

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

The `print` method for a `findFn` object displays the result as a table in a web browser.

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 find a function with that exact name, try `findFn('spline()')`. This produced one match for a function named `regspline`.

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

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

This returned a `findFn` object identifying 78 help pages. The `print` method for an object of class `findFn` presents the result in a web browser. In this case, the sixth row is `lspline` in the `assist` package, which has a `Score` of 1. It is the fifth row in this table, because it is in the `assist` package, which had a total of 34 help pages matching the search term, and this was the only one whose name matched the `grepFn pattern`.

To try to evaluate further the `splineAll findFn` object, we must first acknowledge that a table with 739 rows is too large to digest easily.

`summary(splineAll)` would tell us that the 739 help pages came from 191 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)
```

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 for locally installed packages with other information 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`. Each file corresponds 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 `PackagesSum2` function, which adds information from installed packages not obtained by `findFn`. This includes the package title and date, plus the names of author and maintainer, the date packaged, 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. Clearly, therefore, the value of PackageSum2 can be increased by running install.packages (from the utils package) to install packages not currently available locally.

To make it easier to do this,the sos package includes an installPackage function, which checks all the packages in a findFn 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 of 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);
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 to Find Functions to Solve Differential Equations

The lead author of this article recently gave an invited presentation on "Fitting Nonlinear Differential Equations to Data in R" (Graves et al., 2009). 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
```

The object de has 53 rows, while des has 105. 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 124 rows, which suggests that the corresponding intersection must have (53+105-124) = 34. This can be confirmed via nrow(de & des).

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 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 this 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 Packaged 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

In sum, 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".

Finally, 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 sub-

stantial work that someone else has already done it.

Acknowledgments

The capabilities described here extend the power of the RSiteSearch 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 and 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 RSiteSearch 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 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.

Spencer Graves
President and Chief Operating Officer
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

Bibliography

J. Baron. R site search. http://finzi.psych.upenn.edu/search.html, September 2009.

J. Chambers. *Software for Data Analysis: Programming with R*. Springer, New York, 2009.

S. Graves, G. Hooker, and J. Ramsay. Fitting nonlinear differential equations to data in r. http://stat.sfu.ca/~dac5/workshop09/Spencer_Graves.html, August 2009. conference presentation.

Openoffice.org. *Open Office Calc*. Sun Microsystems, California, USA, 2009. URL http://www.openoffice.org.

R-Forge Team. R-forge. http://r-forge.r-project.org, September 2009.

J. Ramsay, G. Hooker, and S. Graves. *Functional Data Analysis with R and MATLAB*. Springer, New York, 2009.