Searching help pages of R packages

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

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 RSiteSearch data base and returns the matches in a data.frame of class findFn. The corresponding print method 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. 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.

Other R functions can then be used to quickly find what you want among possibly hundreds of matches.

Two examples are considered below: First we find a dataset containing a variable Petal.Length, used without indicating the source by w/o page numbers cite **?**. citet **?**.

citep (**?**).

with page numbers cite (**?**, pp. 282-283). citet **?**, pp. 282-283.

citep (**?**, pp. 282-283).

Second, we study R capabilities for splines, including looking for a function named spline.

Petal.Length

(**?**, pp. 282-283) Chambers (2009, p. 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 R. The sample code he provides does not work by itself. To reproduce his Figure 7.2, we must first obtain a copy of this famous data set in a format compatible with Chambers' code.

How to add Bibliography ? Chambers, John (2009) Software for Data Analysis (Springer)

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 on 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 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.

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

```
> 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.

spline

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

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

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 631 on one day (739 a few months later). That's an improvement over 1526 but is still too many. To get a quick overview of these 631 (or 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 will open the result in a web browser.

However, a table with 739 rows is rather large to digest easily. The 739 help pages came from 191 different packages. The summary method by default will display the first minPackages = 12 plus others with the same number of matches as package number minPackages.

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

```
writeFindFn2xls(splineAll)
```

This produces an Excel file with three sheets:

The PackageSum2 sheet contains information on locally installed packages not available summary.

The findFn sheet contains the search results.

The call sheet gives the call to findFn that generated these search results.

(An Excel file can be opened and edited with Open Office Calc

citation for Open Office Calc?)

The PackageSum2 sheet is created by the PackagesSum2 function, which adds information from installed packages not obtained on all functions by findFn. This includes 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.

To install packages not already available locally, the names of the desired packages can be passed to install.packages or the findFn object can be passed to the installPackages object, which includes a second argument minCount so any any package with at least minCount matches not already installed locally will be installed; the default minCount is the square root of the largest Count. The PackageSum2 results and the PackageSum2 sheet of writeFindFn2xls will typically contain more information after running installPackages than before.

Returning to the findFn object, suppose we want to find a function named spline. For this, 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 search can be generalized using grepFn:

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

This returned a findFn object identifying 60 help pages. The print method for an object of class findFn presents the result in a web browser. In this case, the fifth 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 (as opposed to contents) matched the grep pattern.

To establish priorities among different packages for further study, it might be nice to have a Pareto chart showing the 10 packages with the most help pages relevant to our search string. We can get this as follows:

```
> spSm <- attr(splineAll,'PackageSummary')
> spSm[1:10,'Count']
    assist     fda           gss      mgcv
        34      30            25        22
      VGAM kernlab DierckxSpline bayesSurv
        17      17            16        16
smoothSurv splines
        15      14
```

To obtain a similar Pareto by 'TotalScore' requires a little more effort:

```
> o <- rev(order(spSm[, 'TotalScore']))
> splineSum[o, ][1:10, ]
              Count MaxScore TotalScore
gss              25       35        448
splines          14       45        354
fda              30       48        275
<...>
```

This analysis gave us in seconds a very informative overview of spline capabilities in contributed R packages in a way that can help establish priorities for further study of the different packages and functions.

HTML

The HTML function writes an RSiteSearch object to a file in HTML format and opens it in a browser from which a mouse click will open a desired help page.

The power of this can be seen by applying this function to the grep'ed subset of help pages with names including the phrase spline:

```
HTML(splineAll[select, ])
```

Of the 631 help pages containing spline, this displayed only those whose name included the phrase spline. Similar analyses could display any desired subset of an RSiteSearch object created from merging several calls to RSiteSearch.function.

Summary

In sum, we have found `RSiteSearch.function` in the `RSiteSearch` package to be a very quick and efficient method for finding things in contributed packages.

Acknowledgments

The `RSiteSearch` capabilities 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 improving, but of course can not be blamed for any of its deficiencies.

Spencer Graves
President and Chief Operating Officer
Structure Inspection and Monitoring
San Jose, CA
email: `spencer.graves@prodsyse.com`

Sundar Dorai-Raj
Google
Mountain View, CA
email: `sdorairaj@google.com`

Romain Francois

email: `romain.francois@dbmail.com`