

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 Chambers

... how to cite? w/o page numbers cite ?. citet ?.
citet ?).

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

citet (?, 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 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.

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:

```
> 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. A 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 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 'reg spline'.

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`. (If other packages had the same number of matches as the twelfth package, they would also appear in the 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 a '*.xls' 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.

If `WriteXLS` cannot produce an Excel file with your installation, it will write three *.csv files. (NOTE: 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.

citation for Open Office Calc?)

The `PackageSum2` sheet (or file) 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.

For packages not already installed, the standard `install.packages` function can be used. To simplify this task, the `findFn` object can be passed to the `installPackages` function to check all packages for which the number of matches exceeds a second argument `minCount` and install any of those that are not already installed; the default `minCount` is the square root of the largest Count. The results from `PackageSum2` and the `PackageSum2` sheet of `writeFindFn2xls` will typically contain more information after running `installPackages` than before.

This analysis quickly gave us 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.

Summary

In sum, we have found `findFn` in the `sos` 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