StackExchange client for R

Jeff Gentry

May 27, 2011

1 Introduction

The StackExchange web site family is an extremely rapidly growing collection of question and answer websites. The earliest examples focused on software development, system administration and the like but now topics are very far ranging. The way they work is that people can vote questions up or down based on how good of a question they think it is, as well as the answers given. If one believes in the wisdom of the crowds, this should lead to the best answers to the best questions floating to the top.

2 Initial Notes

2.1 Notes for this version

This package is still very much a work in progress, the edges are still quite rough. If a particular rough edge interests you and you'd like it smoothed out sooner rather than later, please contact me.

2.2 Notes on API coverage

The ultimate goal is to provide full coverage of the StackExchange API, although this is not currently the case. Aspects of the API will be added over time, although if there are particular places that you find missing, please contact me.

2.3 Notes for this document

This document is just attempting to point you in the right direction, and is not providing an encyclopedic overview of the *RStackExchange* package nor the StackExchange API. Refer to the individual man pages for the package to see a complete overview of functionality. For more complete documentation of the StackExchange API see `http://api.stackoverflow.com/1.0/usage`.

3 Managing multiple StackExchange sites

Any valid StackExchange site will work with this package. All functions take a `site` argument, which defaults to `stackoverflow.com`. You may override this default by supplying the domain name (NOT the full URL) for the StackExchange site that you wish to use. A complete list of StackExchange sites is available at `http://stackexchange.com/sites`.

4 API Keys

By default, access to the StackExchange API is considered to be more of a trial basis affair, and a particular IP address can only make 300 requests per day in that manner. Considering that a large query could result in dozens of API requests, this limit won't take you very far. There's a solution to this problem, obtaining a StackExchange API Key. To do this, you will create a dummy application at `http://stackapps.com/apps/register`, and at the end you'll be given an API Key. By using this key, you can up your limit to be 10,000 requests per day. To enable the key in your R session, do the following, where `key` is what was given to you:

```
> registerAPIKey(key)
```

5 You've got questions? We've got answers

There are multiple methods for searching questions. To explain the upcoming nomenclature, consider the following definitions. An *unanswered* question is one where there are no answers provided which have been upvoted. Contrast that with a *no answer* question, which literally has no answers provided. One can also look for *related* questions, which use a private algorithm to identify similar questions as well as *linked* questions which are questions which have been linked to in other questions.

```
> library(RStackExchange)
> allQs <- getQuestions(tagged = "r", fromDate = as.POSIXct("2010-10-01"),
+     toDate = as.POSIXct("2011-01-01"), num = 5)
> allQs

[[1]]
[1] "Modifying fonts in ggplot2"

[[2]]
[1] "paste quotation marks into character string, within a loop"

[[3]]
[1] "R: RGraphviz installation"
```

```
[[4]]
[1] "R: Sort multiple columns by another data.frame?"

[[5]]
[1] "Print the sourced R file to an appendix using Sweave"
```

```
> noAQs <- getNoAnswerQuestions(tagged = "r", fromDate = as.POSIXct("2010-10-01"),
+     toDate = as.POSIXct("2011-01-01"), num = 5)
> noAQs
```

```
list()
```

These questions are all instances of the *seQuestion* class, which provides the *answers* field already included. We can see the number of answers each provides:

```
> aQAnswers <- sapply(allQs, function(x) x$getAnswers())
> nAQAnswers <- sapply(noAQs, function(x) x$getAnswers())
> length(aQAnswers)
```

```
[1] 5
```

```
> length(nAQAnswers)
```

```
[1] 0
```

One can also see the comments for either a question or an answer:

```
> allQs[[1]]$getComments()
```

```
list()
```

```
> allComments <- sapply(allQs, function(x) x$getComments())
> allComments[[1]]
```

```
list()
```

6 Users

At the moment, finding users is a task that's split between two functions. The `searchusers` function is intended for cases where one does not know the specific users, while `getUsers` will return users specified by their user IDs.

In general, there should be little use for `getUsers`, as most situations where one will have actual user IDs (e.g. the asker of a question), the *seUser* object will already be attached to that object (using the same example, the *seQuestion* object).

In this example, we'll find the 15 oldest users with the word 'java' as part of their name, and then identify which one has the highest reputation.

```
> users <- searchUsers(num = 15, filter = "java", sort = "creation",
+     order = "asc")
> reps <- sapply(users, function(x) x$getReputation())
> maxRepUser <- users[[which.max(reps)]]
> maxRepUser$getDisplayName()

[1] "willcodejavaforfood"
```

7 Tags

Tags are a method of aggregating content by topic. A question may have any number of tags applied to it, which allows the question to not have to live under one single organizational layer. For example, a question might be tagged as pertaining to *R* as well as *time-series*.

Tags are not as richly supported by the StackExchange API, but one can still manipulate them a bit. For instance, we can see who the top question askers are for the *javascript* tag:

```
> tag <- getTags(num = 1, filter = "javascript")[[1]]
> tag$topAskers("month", num = 5)

[[1]]
[1] "actual"

[[2]]
[1] "Raynos"

[[3]]
[1] "Åăime Vidas"

[[4]]
[1] "Neal"

[[5]]
[1] "Pacerier"
```

Or given a user, we can see what tags they like to work with, first looking at the top 5 tags that they tend to answer questions about as well as ask questions about.

```
> ansTags <- maxRepUser$topTagsByAnswers(num = 5)
> ansTags

[[1]]
[1] "java"

[[2]]
```

```
[1] "iphone"

[[3]]
[1] "objective-c"

[[4]]
[1] "swing"

[[5]]
[1] "design-patterns"
```

```
> quesTags <- maxRepUser$topTagsByQuestions(num = 5)
> quesTags
```

```
[[1]]
[1] "java"

[[2]]
[1] "jpa"

[[3]]
[1] "hibernate"

[[4]]
[1] "code-review"

[[5]]
[1] "community"
```

Similarly, given a particular tag (or vector of tags), we can see what the top rated questions or answers are for that user.

```
> maxRepUser$topAnswersByTags(ansTags[[1]]$getName(), num = 3)
```

```
[[1]]
[1] "Do you really need the 'finally' block"

[[2]]
[1] "Can two threads run two different methods at the same point of time?"

[[3]]
[1] "What are the main tools/frameworks used nowadays in Java?"
```

```
> maxRepUser$topQuestionsByTags(quesTags[[1]]$getName(), num = 2)
```

```
[[1]]
[1] "Java books for preparing for interviews"
```

```
[[2]]
[1] "Best Practice for Using Java System Properties"
```

8 Badges

Badges are a current trend in websites which theoretically gives users more motivation to participate, using a system of video game like rewards. We can get a list of the badges available on a given StackExchange site, and see what users have been awarded this badge, for example:

```
> badges <- allBadges(num = 300, site = "programmers.stackexchange.com")
> badgeRecipients(badges[[25]]$getBadgeID(), num = 5)

[[1]]
[1] "Pat"

[[2]]
[1] "CMS"

[[3]]
[1] "Mike Taylor"

[[4]]
[1] "Samuh"

[[5]]
[1] "Samuh"
```

9 Session Information

The version number of R and packages loaded for generating the vignette were:

```
R version 2.13.0 (2011-04-13)
Platform: x86_64-apple-darwin9.8.0/x86_64 (64-bit)

locale:
[1] C/en_US.UTF-8/C/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] RStackExchange_0.1.2 RJSONIO_0.7-1        RCurl_1.5-0
[4] bitops_1.0-4.1
```

```
loaded via a namespace (and not attached):
[1] tools_2.13.0
```