HowTo: Automated Querying of PubMed Data

Jeff Gentry

June 23, 2003

1 Overview

This article demonstrates how you can make use of the `query` toolset available in the Bioconductor project to automatically search PubMed and other resources, and utilize that data within your R session. To do this you will need the *Biobase*, *XML*, and *annotate* packages. These must be installed in your version of R and when you start R you must load with the `library` command.

2 Accessing PubMed information

First we load the *annotate* package (this will automatically load the *Biobase* package, as well as the *XML* package if it is needed). For this example, we are utilizing the example data in `eset` to show how this could work with a real world analysis.

```
> library(annotate)
> data(eset)
> affys <- geneNames(eset)[490:500]
> affys

 [1] "31729_at" "31730_at" "31731_at" "31732_at" "31733_at" "31734_at"
 [7] "31735_at" "31736_at" "31737_at" "31738_at" "31739_at"
```

Here we have selected an arbitrary set of 10 genes to be interested in from our sample data. However, `eset` provided us with Affymetrix identifiers, and for the `pubmed` function, we need to use PubMed ID values. To obtain these, we can use the annotation tools within package *annotate*.

```
> library(hgu95av2)
> ids <- multiget(affys, envir = hgu95av2PMID)
> ids <- unlist(ids, use.names = FALSE)
> ids <- ids[!is.na(as.numeric(ids))]
> ids
```

```
 [1] "9695952"  "7729427"  "12408966" "8325638"  "8175896"  "1889752"
 [7] "9315667"  "12590922" "12198562" "10750025" "10601981" "9730618"
[13] "8735594"  "7958621"  "7829601"  "6548703"  "6548702"  "2040595"
[19] "2005217"  "1572287"  "12270951" "11069162" "9221902"  "9016352"
[25] "7566110"  "9582375"  "8646877"  "2878432"  "2574852"  "1973146"
[31] "1358459"  "11883959" "11076525" "10508519" "10072583" "8422497"
[37] "8258301"  "7780165"  "6865942"  "6190133"  "3842206"  "2907503"
[43] "7590364"
```

At this point, we have genes identified in a proper manner for use with the PubMed databases. So if we want to see what, if any, material is stored there for these genes, we can retrieve it in the following manner (Note: This is the point where the *XML* package gets loaded):

```
> x <- pubmed(ids)

Loading required package: XML

> a <- xmlRoot(x)
> numAbst <- length(xmlChildren(a))
> numAbst

[1] 43
```

Our search of the 30 PubMed IDs (from the 10 Affymetrix IDs) has resulted in the same number of abstracts from PubMed (stored in R using XML format). The *annotate* package also provides a `PubMedAbst` class, which will take the raw XML format from PubMed and extract the interesting sections for easy reviewal - at this time we will generate an instance of this class for each returned abstract. (And for a future example, at the same time, we will extract from each class the actual abstract text).

```
> arts <- vector("list", length = numAbst)
> absts <- rep(NA, numAbst)
> for (i in 1:numAbst) {
+     arts[[i]] <- buildPubMedAbst(a[[i]])
+     absts[i] <- abstText(arts[[i]])
+ }
> arts[[7]]

An object of class pubMedAbs
Slot "pmid":
[1] "9315667"

Slot "authors":
 [1] "DP Satijn"      "DJ Olson"       "J van der Vlag" "KM Hamer"
 [5] "C Lambrechts"   "H Masselink"    "MJ Gunster"     "RG Sewalt"
```

```
 [9] "R van Driel"    "AP Otte"

Slot "abstText":
        Polycomb (Pc) is involved in the stable and heritable repression of ho...

Slot "articleTitle":
        Interference with the expression of a novel human polycomb protein, hP...

Slot "journal":
[1] "Mol Cell Biol"

Slot "pubDate":
[1] "Oct 1997"

Slot "abstUrl":
[1] "No URL Provided"
```

As you can see, the `PubMedAbst` class provides several key pieces of information: authors, abstract text, article title, journal, publication date of the journal and any related URL. These can all be individually extracted utilizing the provided methods (such as 'abstText' in the above example).

Next, suppose we are interested in only the abstracts that deal with cDNA. We can utilize the power of R here, now that we have all the abstracts relating to our genes of interest neatly organized.

```
> found <- grep("cDNA", absts)
> goodAbsts <- arts[found]
> length(goodAbsts)

[1] 15
```

So 12 of the articles relating to our genes of interest mention the term cDNA in their abstracts. Suppose at this point you identify one particular abstract that you want to look further at the 2nd abstract obtained. A brief scan of its contents (from above) shows that it contains a URL for its journal:

```
> abstUrl(goodAbsts[[2]])

[1] "No URL Provided"
```

This URL can then be used (either using the `browseURL` function from *annotate* or directly from your favorite browser) to view the full article as well as any related information provided by the authors.

Lastly, as a demonstration for how one can use the `query` toolset to cross reference several databases, we can use the same set of PubMed IDs with another function:

```
> y <- genbank(ids, type = "uid")
> b <- xmlRoot(x)
```

At this point the object `b` can be manipulated in a manner similar to `a` from the PubMed example.

Also, note that both `pubmed` and `genbank` have an option to display the data directly in the browser instead of XML, by specifying `disp="browser"` in the parameter listing.