HOWTO generate biocViews HTML

S. Falcon and V.J. Carey

April 25, 2007

1 Overview

The purpose of *biocViews* is create HTML pages that categorize packages in a Bioconductor package repository according to terms, or *views*, in a controlled vocabulary. The fundamental resource is the VIEWS file placed at the root of a repository. This file contains the complete DESCRIPTION file contents for each package along with additional meta data describing the location of package artifacts such as archive files for different platforms and vignettes.

The standard behavior of the view generation program is to query the repository over the internet. This package includes a static sample VIEWS file so that the examples in this document can run without internet access.

2 Establishing a vocabulary of terms

We use `dot` to describe the vocabulary. For details on the `dot` syntax, see `http://www.graphviz.org/doc/info/lang.html`.

```
> vocabFile <- system.file("dot/biocViewsVocab.dot", package = "biocViews")
> cat(readLines(vocabFile)[1:20], sep = "\n")

/* Bioc Views Vocabular Definition in dot format */

/* How To Process this file:
   1. Use dot2gxl from graphviz to transform into GXL format.
          dot2gxl biocViewsVocab.dot > biocViewsVocab.gxl
   2. use graph::fromGXL to obtain a graphNEL object

*/

digraph G {
```

```
/* Root */
BiocViews -> Software;
BiocViews -> AnnotationData;
BiocViews -> ExperimentData;

/* Software */
Software -> Microarray;
Software -> Annotation;
Software -> Visualization;

> cat("...\n")

...
```

The dot description is transformed to a GXL document using `dot2gxl`, a tool included in the graphviz distribution. The GXL is then converted to a *graphNEL* instance using `fromGXL` from the *graph* package. There is a helper script in the root of the *biocViews* package called `updateVocab.sh` that automates the update process if the required tools are available.

The definition of the vocabulary lacks a notion of order. Since the purpose of the vocabulary is primarily for display, a valuable improvement would be to use graph attributes to allow the ordering of the terms.

Another missing piece is a place to put a text description of each term. This could also be achieved using graph attributes.

2.1 Use Case: adding a term to the vocabulary

To add a new term to the vocabulary:

1. edit the *dot* file `dot/biocViewsVocab.dot` and add the desired term. Note that terms cannot contain spaces and that the underscore character, `_`, should be used instead.
2. ensure that R and dot2gxl are on your PATH.
3. cd into the biocViews working copy directory.
4. run the updateVocab.sh script.
5. reinstall the package and test that the new term is part of the vocabulary. In short, you will load the data using `data(biocViewsVocab)` and check that the term is a node of the graph instance.
6. commit changes to svn.

2.2 Use Case: updating BioConductor website

This is for BioConductor web administrator:

1. update local copy of biocViews using `svn update`.
2. find the correct instance R that is used to generate HTML pages on BioConductor website, and install the updated `biocViews`.
3. re-generate the related HTML packages by using `/home/biocadmin/bin/prepareRepos-*.sh` and `/home/biocadmin/bin/pushRepos-*.sh`.

3 Querying a repository

To generate a list of *BiocViews* objects that can be used to generate HTML views, you will need the repository URL and a graph representation of the vocabulary.

There are three main Bioconductor package repositories: a software repository containing analytic packages, an annotation data repository, and an experiment data repository. The vocabulary of terms has a single top-level node, all other nodes have at least one parent. The top-level node, *BiocViews*, has three children that correspond to the three main Bioconductor repositories: *Software*, *AnnotationData*, and *ExperimentData*. Views for each repository are created separately using `getBiocSubViews`. Below, we demonstrate how to build the *Software* set of views.

```
> data(biocViewsVocab)
> reposPath <- system.file("doc", package = "biocViews")
> reposUrl <- paste("file://", reposPath, sep = "")
> biocViews <- getBiocSubViews(reposUrl, biocViewsVocab, topTerm = "Software")
> print(biocViews[1:2])

$Software
Bioconductor View: Software
Parent Views:
[1] "BiocViews"
Subviews:
[1] "Microarray"        "Annotation"        "Visualization"
[4] "Statistics"        "GraphsAndNetworks" "Technology"
[7] "Infrastructure"
Contains packages:
 [1] "Ruuid"      "aCGH"       "affycomp"   "affydata"   "affypdnn"
 [6] "affxparser" "affylmGUI"  "RBGL"       "Rgraphviz"  "graph"
[11] "affy"
```

```
$Microarray
Bioconductor View: Microarray
Parent Views:
[1] "Software"   "Technology"
Subviews:
[1] "OneChannel"                "TwoChannel"
[3] "DataImport"                "QualityControl"
[5] "Preprocessing"             "Transcription"
[7] "DNACopyNumber"             "SNPsAndGeneticVariability"
Contains packages:
[1] "affy"       "affylmGUI"  "affxparser"
```

4 Generating HTML

By default, the set of HTML views will link to package description pages located in the html subdirectory of the remote repository.

```
> viewsDir <- file.path(tempdir(), "biocViews")
> dir.create(viewsDir)
> writeBiocViews(biocViews, dir = viewsDir)
> dir(viewsDir)[1:2]

[1] "Annotation.html"      "CellBasedAssays.html"
```