the *biocViews* package

VJ Carey

June 30, 2005

Contents

1 A vocabulary graph and some manipulations

A possible vocabulary for bioconductor package topics has been created as a directed graph, saved as `bcVoc` in the *biocViews* package.

```
> data(bcVoc)
> bcVoc

A graph with  directed  edges
Number of Nodes = 53
Number of Edges = 53
```

This graph was created with the graphviz dot language, then converted to GXL using graphviz dot2gxl utility, then imported to R using graph::fromGXL. To see the top-level terms of the vocabulary, use

```
> adj(bcVoc, "vocRoot")
```

```
$vocRoot
 [1] "ArrayCGH"
 [2] "FlowCytometry"
 [3] "Proteomics"
 [4] "DNAMicroarrayPreprocessing"
 [5] "StatisticalModelingForHighThroughputBiology"
 [6] "GraphsAndNetworks"
 [7] "Visualization"
 [8] "Ontologies"
 [9] "SequenceAnalysisAndDatabases"
[10] "QuantitativeGenetics"
[11] "AnnotationInfrastructure"
[12] "GeneralInfrastructure"
[13] "biocMiscellaneous"
```

To see all concept terms subordinate to "ontologies", use

```
> acc(bcVoc, "Ontologies")
```

```
$Ontologies
OntologyInfrastructure          MAGEasOntology            GOasOntology
                     1                       1                       1
```

The entire term set is listed in the appendix.

2 Associating packages with vocabulary terms

The `packAssoc` function will associate each element of a vector of package names with (at present) a single term in the vocabulary. This works using a trivial GUI. The R interpreter prompts the user to give top level, second level (if relevant given top level), and third level (if relevant given second level) terms associated with each package named in the `packlist` argument. The vocabulary graph is given as the second argument.

Suppose our package list is

```
> demop <- c("Biobase", "graph", "limma", "factDesign")
```

Then

```
> pal <- packAssoc(demop, bcVoc)
```

loads `pal` with a list, after the GUI has been used.

```
> names(pal)
```

```
[1] "Biobase"    "graph"      "limma"      "factDesign"
```

```
> pal$factDesign

$top
[1] "StatisticalModelingForHighThroughputBiology"

$second
[1] "DifferentialExpression"

$third
[1] "FactorialDesign"

$maintainer
[1] "Denise Scholtens <dscholte@hsph.harvard.edu>"

$packagename
[1] "factDesign"

$desc
[1] "This package provides a set of tools for analyzing data from a factorial designed

$title
[1] "Factorial designed microarray experiment analysis"
```

Some of the information saved is derived from calls to `packageDescription`.

The package:terminology associations can be permuted:

```
> vpal <- packAssoc2vlist(pal)
> vpal

$DifferentialExpression
[1] "limma"      "factDesign"

$FactorialDesign
[1] "factDesign"

$GeneralInfrastructure
[1] "Biobase"

$GraphInfrastructure
[1] "graph"

$GraphsAndNetworks
[1] "graph"
```

```
$LinearModels
[1] "limma"

$StatisticalModelingForHighThroughputBiology
[1] "limma"      "factDesign"
```

This enables us to build *ctv* structures.

3 Building ctv documents

ctv documents are XML markups of view-related metadata. The appendix includes a full example of a ctv view document.

The basic structural elements are currently:

- `<CRANTaskView>` is the root tag
- `<name>`, `<topic>`, `<maintainer>`; self-explanatory except for topic, which is a plaintext rendering of the topic; the maintainer must also be plaintext, apparently owing to XML syntax restrictions
- `<info>`; can hold a rich HTML markup of narrative about the view, including references to packages, which are marked up with `<pkg>`
- `<packagelist>`, a list of packages marked up with `<pkg>`
- `<links>`, a list of URLs marked up as pure HTML anchors

The `makeCTV` function helps to create such a document from the elements of a view-package-vocabulary association list created by packAssoc2vlist. A trivial illustration:

```
> vn <- names(vpal)
> c1 <- makeCTV(vn[1], vn[1], "None", vpal[[1]], "None", bcVoc)

Loading required package: XML
Loading required package: RBGL

> targ <- tempfile()
> saveXML(c1, file = targ)
> dem <- read.ctv(targ)
> dem
```

```
CRAN Task View
--------------
Name:       DifferentialExpression
Topic:      DifferentialExpression
Maintainer: None

Packages:   factDesign, limma
```

To run this over our entire view set, we can use:

```
> getCTVs <- function(pal, vocab) {
+     vpal <- packAssoc2vlist(pal)
+     vn <- names(vpal)
+     nv <- length(vn)
+     out <- list()
+     for (i in 1:length(vn)) {
+         tmp <- makeCTV(vn[i], vn[i], "None", vpal[[i]], "None",
+             bcVoc)
+         tf <- tempfile()
+         saveXML(tmp, file = tf)
+         out[[vn[i]]] <- read.ctv(tf)
+         unlink(tf)
+     }
+     out
+ }
> allc <- getCTVs(pal, bcVoc)
```

I now have a list of CTV structures in R. We'll serialize them to HTML:

```
> jnk <- sapply(allc, ctv2html)
```

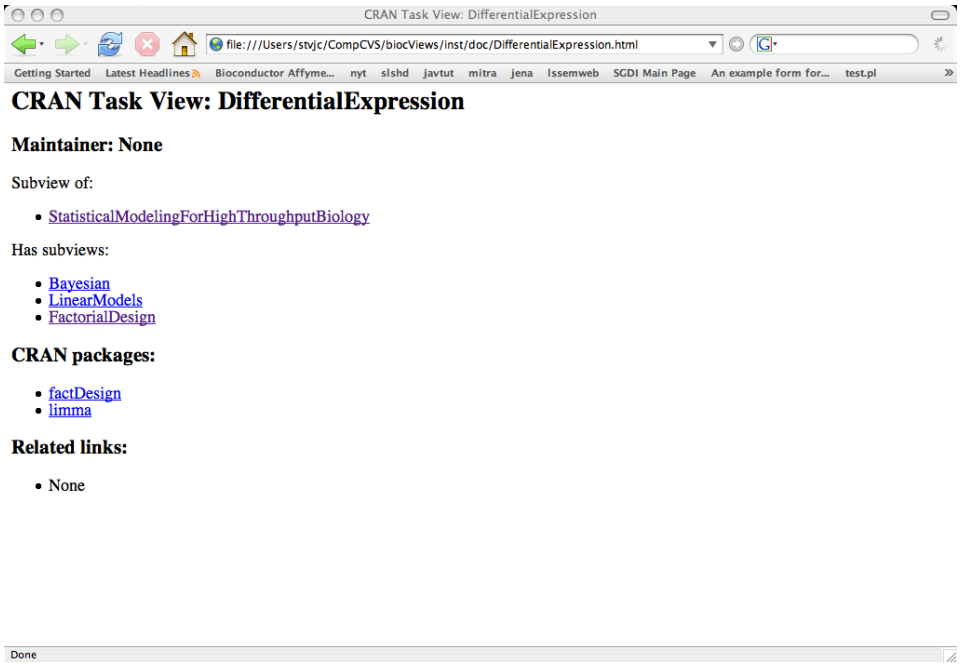

Figure 1: A view of one of the generated HTML pages. To proceed, we need systematic ways of populating the narrative components.

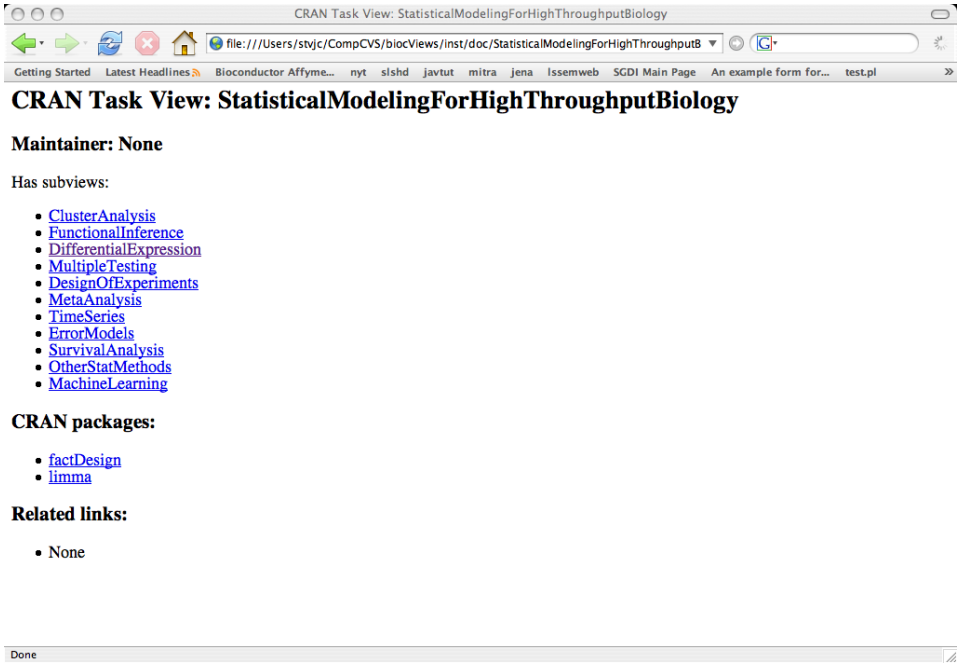

Figure 2: Here's a higher level page on a rich topic set.

4 Appendix

4.1 The full vocabulary

```
> sort(nodes(bcVoc))
```

```
 [1] "AffyAnnotation"
 [2] "AffyPreproc"
 [3] "AnnotationInfrastructure"
 [4] "ArrayCGH"
 [5] "Bayesian"
 [6] "ChromosomeVisualization"
 [7] "ClusterAnalysis"
 [8] "DNAMicroarrayPreprocessing"
 [9] "DesignOfExperiments"
[10] "DifferentialExpression"
[11] "ErrorModels"
[12] "ExternalQueryResolutionWithSequence"
[13] "FactorialDesign"
[14] "FlowCytometry"
[15] "FunctionalInference"
[16] "GOasOntology"
[17] "GeneFiltering"
[18] "GeneralInfrastructure"
[19] "GeneralVisualization"
[20] "GraphDataExamples"
[21] "GraphInfrastructure"
[22] "GraphVisualization"
[23] "GraphsAndNetworks"
[24] "InferenceOnGraphs"
[25] "LinearModels"
[26] "MAGEasOntology"
[27] "MachineLearning"
[28] "MetaAnalysis"
[29] "MultipleTesting"
[30] "Ontologies"
[31] "OntologyInfrastructure"
[32] "OtherDesign"
[33] "OtherPreproc"
[34] "OtherStatMethods"
[35] "ProbeLevelModels"
[36] "ProbeMatching"
[37] "Proteomics"
```

```
[38] "QuantitativeGenetics"
[39] "SampleSize"
[40] "SequenceAnalysisAndDatabases"
[41] "SequenceInfrastructure"
[42] "SequenceSimilarity"
[43] "StatisticalModelingForHighThroughputBiology"
[44] "SurvivalAnalysis"
[45] "TimeSeries"
[46] "Visualization"
[47] "VisualizationInfrastructure"
[48] "WithGO"
[49] "WithPubMed"
[50] "arrayQC"
[51] "biocMiscellaneous"
[52] "cDNAPreproc"
[53] "vocRoot"
```

4.2 Snapshots of vocabulary subgraphs

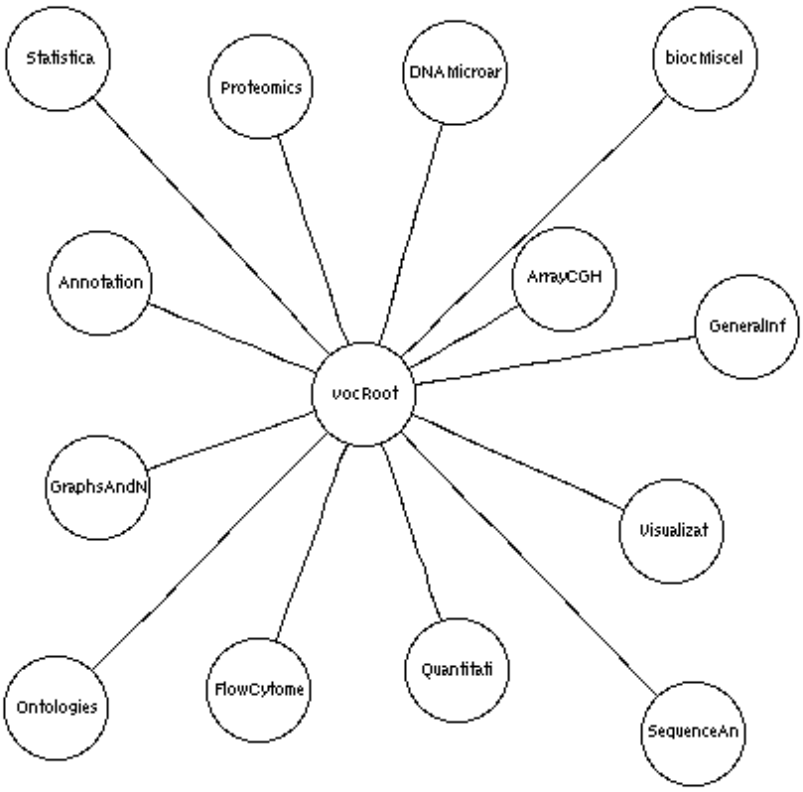

Figure 3: Top level terms.

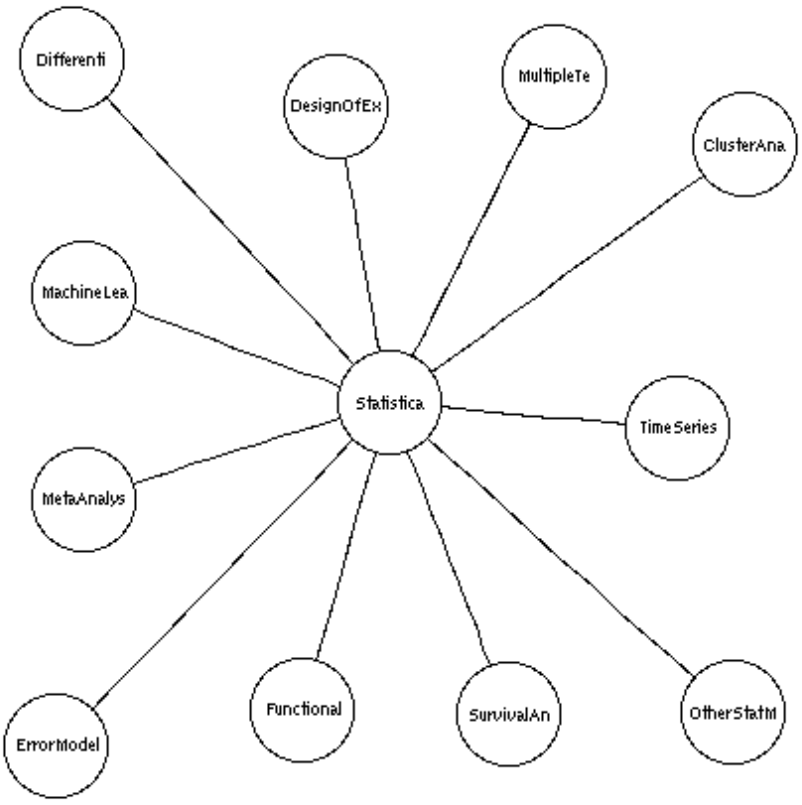

Figure 4: Terms subordinate to statistical modeling for high throughput biology.

4.3 An example ctv document

```
<CRANTaskView>

  <name>MachineLearning</name>
  <topic>Machine Learning &amp;amp; Statistical Learning</topic>
  <maintainer>Torsten Hothorn</maintainer>

  <info>
    Several add-on packages implement ideas and methods developed at the
    borderline between computer science and statistics - this field of research
    is usually referred to as machine learning.

    The packages can be roughly structured into the following topics:
    <ul>
      <li><i>Neural Networks</i>: Single-hidden-layer neural network are
             implemented in package <tt>nnet</tt> as part of the <pkg>VR</pkg>
             bundle (shipped with base R). </li>
      <li><i>Recursive Partitioning</i>: Tree-structured models for
             regression, classification and survival analysis, following the
     ideas in the CART book, are
             implemented in <pkg>rpart</pkg> (shipped with base R) and <pkg>tree</pkg>.
             An adaptation of <pkg>rpart</pkg> for multivariate responses
             is available in package <pkg>mvpart</pkg>. The validity of
             trees can be investigated via permutation approaches with package
             <pkg>rpart.permutation</pkg> and a tree algorithm fitting
             nearest neighbors in each node is implemented in package
             <pkg>knnTree</pkg>. For problems with binary input variables
             the package <pkg>LogicReg</pkg> implements logic regression.
             Graphical tools for the visualization of
             trees are available in packages <pkg>maptree</pkg> and
             <pkg>pinktoe</pkg>.</li>
      <li><i>Regularized and Shrinkage Methods</i>: Regression models with some
             constraint on the parameter estimates can be fitted with the
             <pkg>lasso2</pkg> and <pkg>lars</pkg> packages. The shrunken
             centroids classifier and utilities for gene expression analyses are
             implemented in package <pkg>pamr</pkg>.</li>

      <li><i>Random Forests</i>: The reference implementation of the random
             forest algorithm for regression and classification is available in
             package <pkg>randomForest</pkg>. Package <pkg>ipred</pkg> has bagging
             for regression, classification and survival analysis as well as
             bundling, a combination of multiple models via
```

```
            ensemble learning.</li>
     <li><i>Boosting</i>: Various forms of gradient boosting are
            implemented in packages <pkg>gbm</pkg> and <pkg>boost</pkg>.</li>
     <li><i>Support Vector Machines</i>: The function <tt>svm()</tt> from
            <pkg>e1071</pkg> offers an interface to the LIBSVM library and
            package <pkg>kernlab</pkg> implements a flexible framework
            for kernel learning (including SVMs, RVMs and other kernel
    learning algorithms). An interface to the SVMlight implementation
    (only for one-against-all classification) is provided in package
    <pkg>klaR</pkg>.</li>
     <li><i>Model selection and validation</i>: Package <pkg>e1071</pkg>
            has function <tt>tune()</tt> for hyper parameter tuning and
            function <tt>errorest()</tt> (<pkg>ipred</pkg>) can be used for
            error rate estimation. The cost parameter C for support vector
            machines can be chosen utilizing the functionality of package
            <pkg>svmpath</pkg>.</li>
  </ul>
</info>


<packagelist>
  <pkg>boost</pkg>
  <pkg priority="core">e1071</pkg>
  <pkg priority="core">gbm</pkg>
  <pkg>ipred</pkg>
  <pkg priority="core">kernlab</pkg>
  <pkg>klaR</pkg>
  <pkg>lars</pkg>
  <pkg>lasso2</pkg>
  <pkg>mvpart</pkg>
  <pkg>pamr</pkg>
  <pkg>rpart.permutation</pkg>
  <pkg priority="core">randomForest</pkg>
  <pkg priority="core">rpart</pkg>
  <pkg>svmpath</pkg>
  <pkg>tree</pkg>
  <pkg priority="core">VR</pkg>
</packagelist>

<links>
  <a href="http://www.boosting.org/">Boosting Research Site</a>
</links>
```

```
</CRANTaskView>
```