eSet metadata structures

VJ Carey <stvjc@channing.harvard.edu>

February 10, 2006

1 Introduction

UNDER CONSTRUCTION. SOME CODE MEANINGLESS.

Metadata is data about data. In bioinformatics there are many roles for the metadata concept. This document describes management of metadata on the central data objects of downstream analysis of high-throughput data structures in Bioconductor. Specifically, the *exprSet* and *eSet* classes provide data structure conventions for collections of microarray experiments. These classes can be used to deal with data from other experimental paradigms. For this to occur, metadata conventions will be required so that the different components of the experimental data are identifiable programatically.

As a simple example, consider the recording of patient age. Information on age will be stored in a *phenoData* variable. For some studies, age is recorded in months; for others, age is recorded in years; and in others, age must be coarsened, and is recorded as a categorical response. It is not sufficient to record the age as a numeric variable named "age". One may encode the units of age in the variable name, but a convention for decoding the name may be hard to specify conveniently.

Metadata structures and conventions can help to achieve standards for representation and decoding of data. To continue with the example, we use the exSet example data.

```
> library(Biobase)
> data(exSet)

> exSet

Expression Set (exprSet) with
        500 genes
        26 samples
                 phenoData object with 3 variables and 26 cases
         varLabels
                sex: Female/Male
                type: Case/Control
                score: Testing Score
```

We will extract the *phenoData* component:

```
> pes <- phenoData(exSet)
```

A utility called `convertVarLabels` will upgrade the *phenoData* component so that a general metadata container is present.

```
> pes <- convertVarLabels(pes)
> varMetadata(pes)
```

```
  varName     varLabels
1     sex   Female/Male
2    type  Case/Control
3   score Testing Score
```

Arbitrarily many metadata entries may be added for any variable.

```
> pes <- addVarMetadataEntry(pes, "weight", "units", "pounds")
```

The specific metadata attribute "units" can be ascertained at the *phenoData* level:

```
> getUnits(pes, "weight")
```

```
[1] pounds
Levels: pounds
```

The metadata container is currently a data.frame. For very general programmatic manipulation of metadata, XML or RDF representations may be desirable. Conversion from data.frame to XML may use the StatDataML package or other routines.

2 Details on *phenoData* metadata

At the *phenoData* level, a slot called varMetadata holds a data.frame instance. Currently there is no validation, but it seems reasonable that a varMetadata data.frame should have one row for each *phenoData* variable. Note that the typical reference of the term "variable" is a column of a data frame. For metadata, the variables define rows of the varMetadata container, and variable attributes or properties define the columns of the varMetadata container.

An informally reserved attribute name is `varName`. This is used to name the column that holds the names of variables in the phenoData pData component. The attribute name verb+units+ is recognized by a helper method "getUnits".

Any legacy *exprSet* instance can be converted to one with a varMetadata container using the convertVarLabels helper method. This places the phenoData varLabels content in a data.frame column named `varLabel`. Ultimately the varLabels component of phenoData may be eliminated in favor of varMetadata conventions.

3 Details on `reporterInfo` slot

`reporterInfo` slot is intended to provide a facility for describing the reporters (probes) on a chip. It is symmetric to the `phenoData` slot, except that while `phenoData` describes the samples, `reporterInfo` describes the reporters. The object in the `reporterInfo` slot is a one-column `data.frame` object. Each row in the object represents a probe set in the chip. The values in the `data.frame` object are the predefined types for those probe sets. `reporterInfo` could extract the `data.frame` object from the *exprSet* obect.

```
> reporter <- reporterInfo(exSet)
> head(reporter)
```

```
                probeType
AFFX-MurIL2_at         QC
AFFX-MurIL10_at        QC
AFFX-MurIL4_at         QC
AFFX-MurFAS_at         QC
AFFX-BioB-5_at         QC
AFFX-BioB-M_at         QC
```

```
> levels(reporter[, 1])
```

```
[1] "AntiSenseTarget"      "CommonGroups"         "Incomplete"
[4] "QC"                   "RulesDropped"         "SequenceFamily"
[7] "SimilarityConstraint"
```

We can use the information in `reporterInfo` slot to select probe sets by types.

```
> probeQC <- reporter[, 1] == "QC"
> exSet[probeQC, ]
```

```
Expression Set (exprSet) with
        67 genes
        26 samples
                 phenoData object with 3 variables and 26 cases
         varLabels
                sex: Female/Male
                type: Case/Control
                score: Testing Score
```

4 Details on the eMetadata container for *eSet* objects

The *exprSet* class has proven useful for affymetrix data representation. A new class, *eSet*, allows more complex data representation.

```
> getSlots("eSet")
```

```
         assayData        sampleNames      reporterNames        description
       "listOrEnv"        "character"        "character"            "MIAME"
             notes         annotation            history       reporterInfo
       "character"        "character"        "character" "data.frameOrNULL"
          phenoData
       "phenoData"
```

```
> getClass("listOrEnv")
```

```
Extended class definition ( "ClassUnionRepresentation" )
Virtual Class

No Slots, prototype of class "NULL"

Known Subclasses: "list", "environment"
```

The *exprListNM* class has some legacy (under the name "exprList") in Biobase, and is extended by the metadata-equipped exprList class. The `eMetadata` slot is for a container for metadata about the various components of the exprList. The current container class is *data.frame*. For a standard exprSet like `golubTrain`, this data.frame would have one row. An informally privileged attribute is "name". The `getExpData` method looks for an exprList component named `exprs`, so the "name" attribute should have value `exprs` for this component.

The following shows the basic manipulations currently available; see the eSet class man page for additional methods.

```
> data(exSet)
> md <- data.frame(name = "exprs", etype = "random numbers")
```

Additional conventions should be established for multicolor arrays and other platforms.