)

Introduction to Bioconductor class `ExpressionSet`

Alex Sanchez

(

Contents

1 Bioconductor Classes

Object-oriented design provides a convenient way to represent data and actions that can be performed on them. A *class* can be tought of as a template, a description of what constitutes each instance of the class. An *instance* of a class is a realization of what describes the class. Attributes of a class are data components, and methods of a class are functions, or actions the instance/class is capable of.

The *R* language has an implementation of object concepts through the package *methods*.

The package *Biobase* contains basic strucutures for microarray data.

```
library(Biobase)
```

1.1 class `AnnotatedDataFrame`

Class *AnnotatedDataFrame* is intended to contain covariate information, *i.e.* information relative the hybridization experiments. This is particularily convenient for exploratory analysis, as important covariate are not known.

```
samplenames <- letters[1:10]
dataf <- data.frame(treated=sample(c(TRUE, FALSE), 10, replace=TRUE),
                    sex=sample(c("Male", "Female"), 10, replace=TRUE),
                    mood=sample(c("Happy", "Dontt care", "Grumpy"), 10, replace=TRUE),
                    names=samplenames, row.names="names")
dataDesc = data.frame(c("Treated with dark chocolate", "Sex", "Mood while eating"))

pdata <- new("AnnotatedDataFrame", data=dataf,dataDesc)
```

1.1.1 Exercise

Select a dataset that you understand. Be sure to have information on the covariates in a data frame (e.g. the "targets.txt" created to sotore the groups) Create an `AnnotatedDataFrame`

1.2 class MIAME

Class MIAME was created to adapt Bioconductor data structures to the "Minimum Information About a Microarray Experiment" standard. In practice people tend to skip its use.

```
my.desc <- new("MIAME", name="LPS_Experiment",
            lab="National Cancer Institute",
            contact="Lakshman Chelvaraja",
            title="Molecular basis of age associated cytokine dysregulation in LPS stimula
            url="http://www.jleukbio.org/cgi/content/abstract/79/6/1314")
print(my.desc)

## Experiment data
##   Experimenter name: LPS_Experiment
##   Laboratory: National Cancer Institute
##   Contact information: Lakshman Chelvaraja
##   Title: Molecular basis of age associated cytokine dysregulation in LPS stimulated mac
##   URL: http://www.jleukbio.org/cgi/content/abstract/79/6/1314
##   PMIDs:
##   No abstract available.
```

1.3 class `ExpressionSet`

This class is intended to be a container for high-throughput assays and experimental metadata.

`ExpressionSet` class is derived from the abstract `eSet`, and requires a matrix named `exprs` as assayData member.

Typically, we will use instances of this class to store the results of high throughput experiments.

The only compulsory parameter is the matrix `exprs` but we usually complement it with information and covariates and, perhaps on the experiment or the annotations.

exprs a *matrix* of expression values (one gene per row, one hybridization experiment per column).

phenoData an instance of class *AnnotatedDataFrame*

description an instance of class *MIAME*

annotation an character vector containing the platform name.

```
data(sample.ExpressionSet)
sample.ExpressionSet

## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 500 features, 26 samples
##   element names: exprs, se.exprs
## protocolData: none
## phenoData
##   sampleNames: A B ... Z (26 total)
##   varLabels: sex type score
##   varMetadata: labelDescription
## featureData: none
## experimentData: use experimentData(object)
## Annotation: hgu95av2
```

Slots can be accessed using *accessor methods such as* ***exprs*** *to access the expression matrix (generically called* ***assayData****) or* ***pData*** *used to access the* ***phenoData*** *object*

1.3.1 Exercises

1. *Use a GEO dataset for which you have prepared the expression matrix, the targets file and some additional information and create an ExpressionSet object from scratch to contain all the information.*

2. *Practice extracting information or modifying the expression set.*

2 The GEOquery package

2.1 Overview of GEO

The NCBI Gene Expression Omnibus (GEO) serves as a public repository for a wide range of high-throughput experimental data. These data include single and dual channel microarray-based experiments measuring mRNA, genomic DNA, and protein abundance, as well as non-array techniques such as serial analysis of gene expression (SAGE), mass spectrometry proteomic data, and high-throughput sequencing data.

At the most basic level of organization of GEO, there are four basic entity types. The first three (Sample, Platform, and Series) are supplied by users; the fourth, the dataset, is compiled and curated by GEO staff from the user-submitted data. See the GEO home page for more information.

2.2 Getting data from GEO

Getting data from GEO is really quite easy. There is only one command that is needed, getGEO.

This one function interprets its input to determine how to get the data from GEO and then parse the data into useful R data structures. Usage is quite simple.

```
library(GEOquery)
# gds <- getGEO("GDS507")
gsm <- getGEO(filename=system.file("extdata/GSM11805.txt.gz",package="GEOquery"))
```