Lab 2: Downloading data from GEO

February 2, 2013

1 Gene Expression Omnibus

`http://ncbi.nlm.nih.gov/geo/` The GEO records have several types of format.

1. Platform (GPLxxx)
   Specifies the type of microarray design. One platform can be used in a lot of unrelated experiments, each can include a number of samples.

2. Sample (GSMxxx)
   A sample record is for one particular array hybridization. This may include raw intensity data or processed data.

3. Series (GSExxx)
   A series is a collection of samples done in an experiment or from a set of scientifically related samples.

4. DataSet (GDSxxx) These three types of records are provided by the submitter. The GEO curators may organize the samples to form a different collection called DataSet. "A DataSet represents a curated collection of biologically and statistically comparable GEO Samples and forms the basis of GEO's suite of data display and analysis tools." Samples within a dataset are from the same platform and have gone through similar preprocessing so that their values are considered comparable.

5. Profile data on a gene across samples, derived from DataSets.

2 Search on GEO

Search for datasets by experiment keywords. For example, "lung cancer". You may download an entire dataset or download individual samples. Sometimes you can download "raw" data from a link to a Supplementary File.

Sometimes you learn about a dataset from publications and you may know the record number of the sample number. For these you can directly search by their record ID.

3 Using the getGEO function

Install *GEOquery* from Bioconductor. This is a good chance to introduce "working directory" in R. You can change this by the function "*setwd*" or use the menu File->Change dir. To check your current working directory, do *getwd()*. Once you set the working directory, the downloaded data or any other results you save from the R session will be saved in this directory by default.

```
 setwd("C:/Users/Andrea/Desktop/Bio Labs/Labs")

source("http://www.bioconductor.org/biocLite.R")
biocLite("GEOquery")

library(GEOquery)
#Download data by accession number
sample1 <- getGEO('GSM95009', destdir=".")
```

This will do two things
1. Load the data from GSM95009 and assign it the name sample1
2. Save a dataset in the directory in "*destdir*" You can read in the file that you have downloaded later by using:

```
sample1b <- getGEO(filename="GSM95009.soft")
# You may have to specify the directory if the file is not in your
# working directory.
#getGEO(GEO = NULL, filename = NULL, destdir = tempdir(), GSElimits=NULL,
#GSEMatrix=TRUE,AnnotGPL=FALSE)
```

3.1 Retrieving information from the dataset you downloaded

```
help("GSM-class")
#signature(object = "GEOData"): returns the column descriptions for the current
Columns(sample1)
#signature(object = "GEOData"): returns the metadata for the current object
Meta(sample1)

> str(Meta(sample1))
```

```
List of 36
 $ channel_count          : chr "2"
 $ characteristics_ch1    : chr "ref- t=0"
 $ characteristics_ch2    : chr "t=8min"
 $ contact_address        : chr "300 Pasteur Drive"
 $ contact_city           : chr "Stanford"
 $ contact_country        : chr "USA"
 $ contact_department     : chr "Stanford University, School of Medicine"
 $ contact_email          : chr "array@genome.stanford.edu"
 $ contact_institute      : chr "Stanford Microarray Database (SMD)"
 $ contact_name           : chr ",,Stanford Microarray Database"
 $ contact_phone          : chr "650-498-6012"
 $ contact_state          : chr "CA"
 $ contact_web_link       : chr "http://genome-www5.stanford.edu/"
 $ contact_zip/postal_code: chr "94305"
 $ data_processing        : chr "VALUE is Log (base 2) of the ratio of the median
 $ data_row_count         : chr "10752"
 $ description            : chr [1:2] "Simple annotation: Chemostat, Time course"
 $ geo_accession          : chr "GSM95009"
 $ label_ch1              : chr "Cy3"
 $ label_ch2              : chr "Cy5"
 $ last_update_date       : chr "Jan 19 2007"
 $ molecule_ch1           : chr "total RNA"
 $ molecule_ch2           : chr "total RNA"
 $ organism_ch1           : chr "Saccharomyces cerevisiae"
 $ organism_ch2           : chr "Saccharomyces cerevisiae"
 $ platform_id            : chr "GPL3415"
 $ series_id              : chr "GSE4158"
 $ source_name_ch1        : chr "ref- t=0"
 $ source_name_ch2        : chr "t=8min"
 $ status                 : chr "Public on Feb 03 2006"
 $ submission_date        : chr "Feb 02 2006"
 $ supplementary_file     : chr "NONE"
 $ taxid_ch1              : chr "4932"
 $ taxid_ch2              : chr "4932"
 $ title                  : chr "glucose pulse (0.2g/l) on  galactose chemostat (t
 $ type                   : chr "RNA"
```

```
> Meta(sample1)$platform
```

```
[1] "GPL3415"
```

```
> gpl1=getGEO("GPL3415")
> #signature(object = "GEOData"): returns the "Table" for the current object
> x1=Table(sample1)
> dim(x1)
```

```
[1] 10752    54
```

Suppose I want to get info for just 2 channels. Let's define a function and apply it to the GSM data.

```
> myfun1=function(aGSM){
+   tmp=Table(aGSM)
+   tmp[,c("CH1D_MEAN","CH2D_MEAN")]
+ }
> x1=myfun1(sample1)
```

What type of object is x1?

```
> class(x1)
```

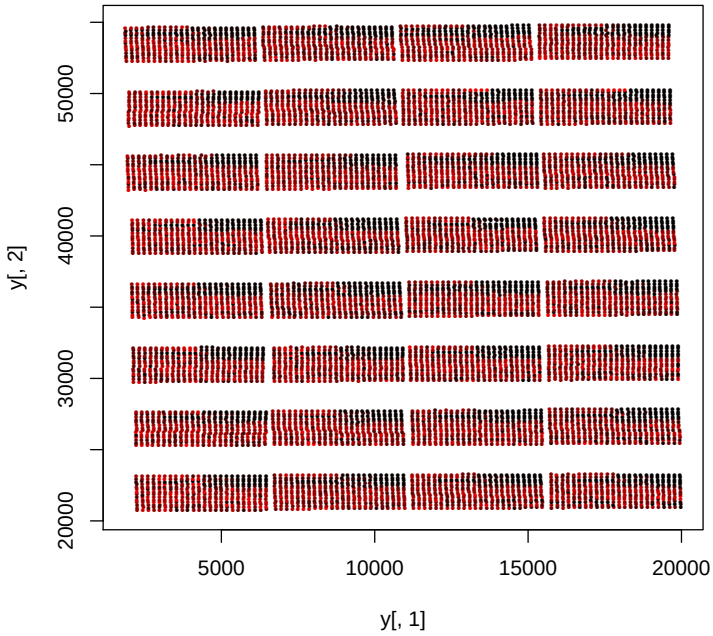
```
[1] "data.frame"
```

3.2 Using the X,Y coordinates

Sometimes you can get raw data with array design information such that you can read these into a dataset like *swirl*, and the platform information is stored with the data. Functions in the *marray* package are convenient. Sometimes you have to use the information in the downloaded data.

Here's an example how you can make an image from the previous example sample to explore spatial pattern.

```
> x1=Table(sample1)
> y=x1[,c("X_COORD","Y_COORD","CH1I_MEAN","CH2I_MEAN")]
> R1=log2(pmax(1,y[,3]))
> R1=(R1-min(R1))/diff(range(R1)) #STANDARDIZE TO (0,1) RANGE
> myRed1=rgb(red=R1,green=0,blue=0) #DEFINE COLOR
> plot(y[,1],y[,2],col=myRed1,pch=16,cex=.4)
```



3.3 Converting GEO data into MAlist or exprSet

You can also use the GDS2MA function to convert a GEO dataset to MAlist. Later we will introduce the class exprSet, and you can use GDS2.

```
> x2=GDS2MA(sample1,GPL=gpl1)

> data1=getGEO("GDS1752",destdir=".")#
> data1=getGEO(filename="GDS1752.soft.gz")
> Meta(sample1)$platform

[1] "GPL3415"

> data2=GDS2MA(data1,GPL=gpl1,do.log2=TRUE)
> #do.log2 Boolean, should the data in the GDS be log2 transformed before insert

> gds505 <- getGEO('GDS505')
> Meta(sample1)$platform

[1] "GPL3415"

> MA <- GDS2MA(gds505)
```

4 Two examples of running loess and spline

Load the built-in R dataset "cars" and read the help file.

- loess: Fit a polynomial surface determined by one or more numerical predictors, using local fitting.
- bs: Generate the B-spline basis matrix for a polynomial spline.

```
> cars.lo <- loess(dist ~ speed, cars)
> plot(cars$dist ~ cars$speed)
> ### show fitted values using loess
> points(cars$speed,predict(cars.lo),col=3,pch=8)
> library(splines)
> basis1=bs(cars$speed,df=3)
> fit2=lm(cars$dist~basis1)
> ### show fitted values using spline
> points(cars$speed,fit2$fitted,col=4,pch=3)
```