

Using the GEOquery package

Sean Davis^{†*}

February 15, 2006

[†]Cancer Genetics Branch
National Human Genome Research Institute
National Institutes of Health

Contents

1	Overview of GEO	1
1.1	Platforms	1
1.2	Samples	2
1.3	Series	2
1.4	Datasets	2
2	Getting Started using GEOquery	2
3	GEOquery Data Structures	3
3.1	The GDS, GSM, and GPL classes	3
3.2	The GSE class	6
4	Converting to BioConductor exprSets and limma MALists	10
4.1	Converting GDS to an exprSet	10
4.2	Converting GDS to an MAList	11
4.3	Converting GSE to an exprSet	15
5	Conclusion	17

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

*sdavis2@mail.nih.gov

non-array techniques such as serial analysis of gene expression (SAGE), and mass spectrometry proteomic data. Currently, 65,000 samples and nearly 2000 different platforms are represented in GEO!

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.¹

1.1 Platforms

A Platform record describes the list of elements on the array (e.g., cDNAs, oligonucleotide probesets, ORFs, antibodies) or the list of elements that may be detected and quantified in that experiment (e.g., SAGE tags, peptides). Each Platform record is assigned a unique and stable GEO accession number (GPLxxx). A Platform may reference many Samples that have been submitted by multiple submitters.

1.2 Samples

A Sample record describes the conditions under which an individual Sample was handled, the manipulations it underwent, and the abundance measurement of each element derived from it. Each Sample record is assigned a unique and stable GEO accession number (GSMxxx). A Sample entity must reference only one Platform and may be included in multiple Series.

1.3 Series

A Series record defines a set of related Samples considered to be part of a group, how the Samples are related, and if and how they are ordered. A Series provides a focal point and description of the experiment as a whole. Series records may also contain tables describing extracted data, summary conclusions, or analyses. Each Series record is assigned a unique and stable GEO accession number (GSExxx).

1.4 Datasets

GEO DataSets (GDSxxx) are curated sets of GEO Sample data. A GDS record represents a 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 GDS refer to the same Platform, that is, they share a common set of probe elements. Value measurements for each Sample within a GDS are assumed to be calculated in an equivalent manner, that is, considerations such as background processing and normalization are consistent across the dataset. Information reflecting experimental design is provided through GDS subsets.

¹See <http://www.ncbi.nih.gov/geo> for more information

2 Getting Started using GEOquery

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)
```

This loads the GEOquery library.

```
> gds <- getGEO("GDS1")
```

File stored at:

```
/tmp/Rtmp1ZIBNz/GDS1.soft.gz
```

```
parsing geodata
```

```
parsing subsets
```

```
ready to return
```

Now, `gds` contains the R data structure (of class *GDS*) that represents the GDS1 entry from GEO. You'll note that the filename used to store the download was output to the screen (but not saved anywhere) for later use to a call to `getGEO(filename=...)`.

We can do the same with any other GEO accession, such as GSM3, a GEO sample.

```
> gsm <- getGEO("GSM3")
```

File stored at:

```
/tmp/Rtmp1ZIBNz/GSM3.soft
```

3 GEOquery Data Structures

The GEOquery data structures really come in two forms. The first, comprising *GDS*, *GPL*, and *GSM* all behave similarly and accessors have similar effects on each. The fourth GEOquery data structure, *GSE* is a composite data type made up of a combination of *GSM* and *GPL* objects. I will explain the first three together first.

3.1 The GDS, GSM, and GPL classes

Each of these classes is comprised of a metadata header (taken nearly verbatim from the SOFT format header) and a `GEODDataTable`. The `GEODDataTable` has two simple parts, a `Columns` part which describes the column headers on the `Table` part. There is also a `show` method for each class. For example, using the `gsm` from above:

```
> Meta(gsm)
```

\$channel_count
[1] "1"

\$contact_address
[1] "6 Center Drive"

\$contact_city
[1] "Bethesda"

\$contact_country
[1] "USA"

\$contact_department
[1] "LCDB"

\$contact_email
[1] "oliver@helix.nih.gov"

\$contact_fax
[1] "301-496-5239"

\$contact_institute
[1] "NIDDK, NIH"

\$contact_name
[1] "Brian,,Oliver"

\$contact_phone
[1] "301-496-5495"

\$contact_state
[1] "MD"

\$contact_web_link
[1] "<http://www.niddk.nih.gov/intram/people/boliver.htm>"

\$contact_zip/postal_code
[1] "20892"

\$data_row_count
[1] "3456"

```

$description
[1] "Testis dissected from adult (12-24 hours post-eclosion) Drosophila melanogaster of
[2] "Keywords = gonad, male, sex"

$geo_accession
[1] "GSM3"

$molecule_ch1
[1] "total RNA"

$organism_ch1
[1] "Drosophila melanogaster"

$platform_id
[1] "GPL5"

$series_id
[1] "GSE462"

$source_name_ch1
[1] "y w[67c1]/Y testis"

$status
[1] "Public on Oct 18 2000"

$submission_date
[1] "Oct 18 2000"

$title
[1] "testis a"

$type
[1] "RNA"

```

```
> Table(gsm)[1:5, ]
```

	ID_REF	SIGNAL_RAW	BKD_FORM	NORM_FORM	BKD_RAW	NORM_VALUE	CONST	VALUE
1	1	138392.6	no	no	101113.8	395070.1	39542	76820.87
2	2	100973.5	no	no	101113.8	395070.1	39542	39401.71
3	3	118994.0	no	no	101113.8	395070.1	39542	57422.25
4	4	108126.1	yes	no	101113.8	395070.1	39542	46554.27
5	5	293362.1	no	no	101113.8	395070.1	39542	231790.33

```
> Columns(gsm)
```

	Column	Description
1	ID_REF	
2	SIGNAL_RAW	raw signal
3	BKD_FORM	
4	NORM_FORM	
5	BKD_RAW	raw background as taken in four quarters of microarray
6	NORM_VALUE	normalization value
7	CONST	constant value
8	VALUE	

The *GPL* behaves exactly as the *GSM* class. However, the GDS has a bit more information associated with the *Columns* method:

```
> Columns(gds)
```

	sample	gender	tissue	
1	GSM3	male	testis	
2	GSM4	male	testis	
3	GSM5	male	gonadectomized male	
4	GSM6	male	gonadectomized male	
5	GSM7	female	ovary	
6	GSM8	female	ovary	
7	GSM9	female	gonadectomized female	
8	GSM10	female	gonadectomized female	

	description
1	Value for GSM3: testis a; src: y w[67c1]/Y testis
2	Value for GSM4: testis b; src: y w[67c1]/Y testis
3	Value for GSM5: male a; src: y w[67c1]/Y male
4	Value for GSM6: male b; src: y w[67c1]/Y
5	Value for GSM7: ovary a; src: y w[67c1] ovary
6	Value for GSM8: ovary b; src: y w[67c1] ovary
7	Value for GSM9: female a; src: y w[67c1] female
8	Value for GSM10: female b; src: y w[67c1] female

3.2 The GSE class

The *GSE* is the most confusing of the GEO entities. A GSE entry can represent an arbitrary number of samples run on an arbitrary number of platforms. The *GSE* has a metadata section, just like the other classes. However, it doesn't have a *GEODDataTable*. Instead, it contains two lists, accessible using *GPLList* and *GSMList*, that are each lists of *GPL* and *GSM* objects. To show an example:

```
> gse <- getGEO("GSE462")
```

```
File stored at:
/tmp/Rtmp1ZIbNz/GSE462.soft.gz
Parsing....
^PLATFORM = GPL5
^SAMPLE = GSM3
^SAMPLE = GSM4
^SAMPLE = GSM5
^SAMPLE = GSM6
^SAMPLE = GSM7
^SAMPLE = GSM8
^SAMPLE = GSM9
```

```
> Meta(gse)
```

```
$contact_address
[1] "6 Center Drive"
```

```
$contact_city
[1] "Bethesda"
```

```
$contact_country
[1] "USA"
```

```
$contact_department
[1] "LCDB"
```

```
$contact_email
[1] "oliver@helix.nih.gov"
```

```
$contact_fax
[1] "301-496-5239"
```

```
$contact_institute
[1] "NIDDK, NIH"
```

```
$contact_name
[1] "Brian,,Oliver"
```

```
$contact_phone
[1] "301-496-5495"
```

```
$contact_state
[1] "MD"
```

```

$contact_web_link
[1] "http://www.niddk.nih.gov/intram/people/boliver.htm"

$`contact_zip/postal_code`
[1] "20892"

$contributor
[1] "Justen,,Andrews"    "Gerard,G,Bouffard" "Chris,,Cheadle"
[4] "Jining,,L\303\274" "Kevin,G,Becker"    "Brian,,Oliver"

$geo_accession
[1] "GSE462"

$platform_id
[1] "GPL5"

$pubmed_id
[1] "11116097"

$sample_id
[1] "GSM10" "GSM3"  "GSM4"  "GSM5"  "GSM6"  "GSM7"  "GSM8"  "GSM9"

$status
[1] "Public on Jul 16 2003"

$submission_date
[1] "Jun 25 2003"

$summary
[1] "Identification and annotation of all the genes in the sequenced Drosophila genome i

$title
[1] "Analysis of transcription in the Drosophila melanogaster testis"

$type
[1] "other"

> names(GSMList(gse))

[1] "GSM10" "GSM3"  "GSM4"  "GSM5"  "GSM6"  "GSM7"  "GSM8"  "GSM9"

> GSMList(gse)[[1]]

```



```

An object of class "GSM"
channel_count
[1] "1"
contact_address
[1] "6 Center Drive"
contact_city
[1] "Bethesda"
contact_country
[1] "USA"
contact_department
[1] "LCDB"
contact_email
[1] "oliver@helix.nih.gov"
contact_fax
[1] "301-496-5239"
contact_institute
[1] "NIDDK, NIH"
contact_name
[1] "Brian,,Oliver"
contact_phone
[1] "301-496-5495"
contact_state
[1] "MD"
contact_web_link
[1] "http://www.niddk.nih.gov/intram/people/boliver.htm"
contact_zip/postal_code
[1] "20892"
data_row_count
[1] "3456"
description
[1] "Whole adult male minus (12-24 hours post-eclosion) Drosophila melanogaster of the g
geo_accession
[1] "GSM10"
molecule_ch1
[1] "total RNA"
organism_ch1
[1] "Drosophila melanogaster"
platform_id
[1] "GPL5"
series_id
[1] "GSE462"
source_name_ch1

```

```

[1] "y w[67c1] female"
status
[1] "Public on Oct 18 2000"
submission_date
[1] "Oct 18 2000"
title
[1] "female b"
type
[1] "cDNA"
An object of class "GEODataTable"
***** Column Descriptions *****
      Column      Description
1      ID_REF
2 SIGNAL_RAW      raw signal
3      BKD_FORM
4  NORM_FORM
5      BKD_RAW      raw background
6 NORM_VALUE normalization value
7      CONST      constant value
8      VALUE
***** Data Table *****
  ID_REF SIGNAL_RAW BKD_FORM NORM_FORM  BKD_RAW NORM_VALUE CONST  VALUE
1      1    4486.49        0          0 3379.579   23337.54 39542 55845.45
2      2    3482.51        0          0 3379.579   23337.54 39542 41058.05
3      3    3812.39        0          0 3379.579   23337.54 39542 45916.78
4      4    3257.56         1          0 3379.579   23337.54 39542 37744.81
5      5    5436.91        0          0 3379.579   23337.54 39542 69843.97
3451 more rows ...

> names(GPLList(gse))
[1] "GPL5"

```

4 Converting to BioConductor *exprSets* and *limma* *MALists*

GEO datasets are (unlike some of the other GEO entities), quite similar to the *limma* data structure *MAList* and to the *Biobase* data structure *exprSet*. Therefore, there are two functions, `GDS2MA` and `GDS2eSet` that accomplish that task.

4.1 Converting GDS to an *exprSet*

Taking our `gds` object from above, we can simply do:

```
> eset <- GDS2eSet(gds, do.log2 = TRUE)
```

Loading required package: Biobase

Welcome to Bioconductor

Vignettes contain introductory material.
To view, simply type 'openVignette()' or start with 'help(Biobase)'.
For details on reading vignettes, see the openVignette help page.

Now, `eset` is an *exprSet* that contains the same information as in the GEO dataset, including the sample information, which we can see here:

```
> eset
```

Expression Set (exprSet) with
3456 genes
8 samples
phenodata object with 4 variables and 8 cases
varLabels
: sample
: gender
: tissue
: description

```
> pData(eset)
```

	sample	gender	tissue	
GSM3	GSM3	male	testis	
GSM4	GSM4	male	testis	
GSM5	GSM5	male	gonadectomized male	
GSM6	GSM6	male	gonadectomized male	
GSM7	GSM7	female	ovary	
GSM8	GSM8	female	ovary	
GSM9	GSM9	female	gonadectomized female	
GSM10	GSM10	female	gonadectomized female	

	description
GSM3	Value for GSM3: testis a; src: y w[67c1]/Y testis
GSM4	Value for GSM4: testis b; src: y w[67c1]/Y testis
GSM5	Value for GSM5: male a; src: y w[67c1]/Y male
GSM6	Value for GSM6: male b; src: y w[67c1]/Y
GSM7	Value for GSM7: ovary a; src: y w[67c1] ovary
GSM8	Value for GSM8: ovary b; src: y w[67c1] ovary
GSM9	Value for GSM9: female a; src: y w[67c1] female
GSM10	Value for GSM10: female b; src: y w[67c1] female

4.2 Converting GDS to an MAlist

No annotation information (called platform information by GEO) was retrieved from because *exprSet* does not contain slots for gene information, typically. However, it is easy to obtain this information. First, we need to know what platform this GDS used. Then, another call to `getGEO` will get us what we need.

```
> Meta(gds)$platform
```

```
[1] "GPL5"
```

```
> gpl <- getGEO("GPL5")
```

File stored at:

```
/tmp/Rtmp1Z1bNz/GPL5.soft
```

So, `gpl` now contains the information for GPL5 from GEO. Unlike *exprSet*, the *limma* *MAlist* does store gene annotation information, so we can use our newly created `gpl` of class *GPL* in a call to `GDS2MA` like so:

```
> MA <- GDS2MA(gds, GPL = gpl)
```

Loading required package: limma

```
> MA
```

An object of class "MAlist"

\$M

	GSM3	GSM4	GSM5	GSM6	GSM7	GSM8	GSM9	GSM10
1	76820.9	71715.8	51430.5	139716.0	45027.9	69984.3	38569.0	55845.4
2	39401.7	NA	37746.6	91150.4	29691.5	36329.5	30363.8	41058.1
3	57422.3	18338.5	37134.6	75928.1	34181.7	42713.9	32090.5	45916.8
4	46554.3	10928.6	34145.2	74550.3	28498.8	28617.4	33207.6	37744.8
5	231790.0	341779.0	77703.8	99999.6	61152.0	65974.4	60665.4	69844.0

3451 more rows ...

\$A

NULL

\$targets

	sample	gender	tissue
1	GSM3	male	testis
2	GSM4	male	testis
3	GSM5	male	gonadectomized male
4	GSM6	male	gonadectomized male

```

5 GSM7 female          ovary
6 GSM8 female          ovary
7 GSM9 female gonadectomized female
8 GSM10 female gonadectomized female

```

```

description
1 Value for GSM3: testis a; src: y w[67c1]/Y testis
2 Value for GSM4: testis b; src: y w[67c1]/Y testis
3 Value for GSM5: male a; src: y w[67c1]/Y male
4 Value for GSM6: male b; src: y w[67c1]/Y
5 Value for GSM7: ovary a; src: y w[67c1] ovary
6 Value for GSM8: ovary b; src: y w[67c1] ovary
7 Value for GSM9: female a; src: y w[67c1] female
8 Value for GSM10: female b; src: y w[67c1] female

```

``` $genes ```

	ID	GB_ACC	BSCC_ID	CLONE_ID	SUB.ARRAY	DUPLICATE	ROW	COLUMN	PCR_QC	SPOT_ID
1	1	AI944549	bs03g07	FBgn0033989		1	a	1	1	passed
2	2	AI944695	bs04c11	FBgn0032821		1	a	1	2	passed
3	3	AI944741	bs04h01	FBgn0034374		1	a	1	3	passed
4	4	AI944801	bs05f04	FBgn0039421		1	a	1	4	failed
5	5	AI945043	bs08c11	FBgn0045370		1	a	1	5	passed

```

1
2
3
4 gi|4505995|ref|NP_002697.1|PPPM1B| protein phosphatase 1B (formerly 2C), magnesium-dep
5

```

```

E_VAL SPOT_QC
1 2e-08 44364
2 NA 16957
3 NA 17896
4 1e-25 16363
5 NA 83502
3451 more rows ...

```

``` $notes ```

```

[[1]]
[1] "able_begin"

```

``` $channel_count ```

```

[1] "1"

```

\$description
[1] "Adult testis gene expression profile and gene discovery. Examines testis, whole mal

\$feature_count
[1] "3456"

\$order
[1] "none"

\$platform
[1] "GPL5"

\$platform_organism
[1] "Drosophila melanogaster"

\$platform_technology_type
[1] "spotted DNA/cDNA"

\$pubmed_id
[1] "11116097"

\$reference_series
[1] "GSE462"

\$sample_count
[1] "8"

\$sample_organism
[1] "Drosophila melanogaster"

\$sample_type
[1] "cDNA"

\$title
[1] "Testis gene expression profile"

\$type
[1] "gene expression array-based"

\$update_date
[1] "07/03/2004"

```
$value_type  
[1] "count"
```

Now, `MA` is of class *MAList* and contains not only the data, but the sample information and gene information associated with GDS1.

4.3 Converting GSE to an `exprSet`

Converting a *GSE* object to an *exprSet* object currently takes a bit of R data manipulation due to the varied data that can be stored in a *GSE* and the underlying *GSM* and *GPL* objects. However, using a simple example will hopefully be illustrative of the technique.

First, we need to make sure that all of the *GSMs* are from the same platform:

```
> gsmplatforms <- lapply(GSMList(gse), function(x) {  
+   Meta(x)$platform  
+ })  
> gsmplatforms
```

```
$GSM10  
[1] "GPL5"
```

```
$GSM3  
[1] "GPL5"
```

```
$GSM4  
[1] "GPL5"
```

```
$GSM5  
[1] "GPL5"
```

```
$GSM6  
[1] "GPL5"
```

```
$GSM7  
[1] "GPL5"
```

```
$GSM8  
[1] "GPL5"
```

```
$GSM9  
[1] "GPL5"
```

Indeed, they all used GPL5 as their platform (which we could have determined by looking at the `GPLList` for `gse`, which shows only one GPL for this particular GSE.). So, now we

would like to know what column represents the data that we would like to extract. Looking at the first few rows of the Table of a single GSM will likely give us an idea (and by the way, GEO uses a convention that the column that contains the single “measurement” for each array is called the “VALUE” column, which we could use if we don’t know what other column is most relevant).

```
> Table(GSMList(gse)[[1]])[1:5, ]
```

	ID_REF	SIGNAL_RAW	BKD_FORM	NORM_FORM	BKD_RAW	NORM_VALUE	CONST	VALUE
1	1	4486.49	0	0	3379.579	23337.54	39542	55845.45
2	2	3482.51	0	0	3379.579	23337.54	39542	41058.05
3	3	3812.39	0	0	3379.579	23337.54	39542	45916.78
4	4	3257.56	1	0	3379.579	23337.54	39542	37744.81
5	5	5436.91	0	0	3379.579	23337.54	39542	69843.97

```
> Columns(GSMList(gse)[[1]])[1:5, ]
```

	Column	Description
1	ID_REF	
2	SIGNAL_RAW	raw signal
3	BKD_FORM	
4	NORM_FORM	
5	BKD_RAW	raw background

We will indeed use the “VALUE” column. We then want to make a matrix of these values like so:

```
> probesets <- Table(GPLList(gse)[[1]])$ID
> data.matrix <- log2(do.call("cbind", lapply(GSMList(gse), function(x) {
+   tab <- Table(x)
+   mymatch <- match(probesets, tab$ID_REF)
+   return(tab$VALUE[mymatch])
+ })))
> data.matrix[1:5, ]
```

	GSM10	GSM3	GSM4	GSM5	GSM6	GSM7	GSM8	GSM9
[1,]	15.76915	16.22921	16.13000	15.65034	17.09214	15.45853	16.09474	15.23515
[2,]	15.32538	15.26597	NaN	15.20406	16.47596	14.85776	15.14885	14.89007
[3,]	15.48673	15.80932	14.16259	15.18048	16.21235	15.06094	15.38242	14.96986
[4,]	15.20399	15.50663	13.41582	15.05939	16.18593	14.79861	14.80460	15.01923
[5,]	16.09185	17.82246	18.38270	16.24570	16.60964	15.90011	16.00962	15.88859

Note that we do a “match” to make sure that the values and the platform information are in the same order. Finally, to make the *exprSet* object:


```

> require(Biobase)

[1] TRUE

> rownames(data.matrix) <- probesets
> colnames(data.matrix) <- names(GSMList(gse))
> pdata <- data.frame(samples = names(GSMList(gse)))
> rownames(pdata) <- names(GSMList(gse))
> pheno <- new("phenoData", pData = pdata, varLabels = as.list("samples"))
> eset2 <- new("exprSet", exprs = data.matrix, phenoData = pheno)
> eset2

```

```

Expression Set (exprSet) with
  3456 genes
  8 samples
      phenoData object with 1 variables and 8 cases
varLabels
: samples

```

So, using a combination of `lapply` on the `GSMList`, one can extract as many columns of interest as necessary to build the data structure of choice. Because the GSM data from the GEO website are fully downloaded and included in the *GSE* object, one can extract foreground and background as well as quality for two-channel arrays, for example. Getting array annotation is also a bit more complicated, but by replacing “platform” in the `lapply` call to get platform information for each array, one can get other information associated with each array. Future work with this package will likely focus on better tools for manipulating *GSE* data.

5 Conclusion

The `GEOquery` package provides a bridge to the vast array resources contained in the NCBI GEO repositories. By maintaining the full richness of the GEO data rather than focusing on getting only the “numbers”, it is possible to integrate GEO data into current Bioconductor data structures and to perform analyses on that data quite quickly and easily. These tools will hopefully open GEO data more fully to the array community at large.