

Using the GEOquery package

Sean Davis^{‡*}

December 7, 2009

[‡]Genetics Branch
National Cancer 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	7
4	Converting to BioConductor ExpressionSets and limma MALists	9
4.1	Getting GSE Series Matrix files as an ExpressionSet	9
4.2	Converting GDS to an ExpressionSet	10
4.3	Converting GDS to an MAList	12
4.4	Converting GSE to an ExpressionSet	17
5	Accessing Raw Data from GEO	22
6	Conclusion	22
7	sessionInfo	22

*sdavis2@mail.nih.gov

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.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). Series records are available in a couple of formats which are handled by GEOquery independently. The smaller and new GSEMatrix files are quite fast to parse; a simple flag is used by GEOquery to choose to use GSEMatrix files (see below).

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

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

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.

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(filename = system.file("extdata/GDS507.soft.gz",  
+   package = "GEOquery"))
```

Now, `gds` contains the R data structure (of class *GDS*) that represents the GDS507 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(filename = system.file("extdata/GSM11805.txt.gz",  
+   package = "GEOquery"))
```

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 `GEODataTable`. The `GEODataTable` 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"

$comment
[1] "Raw data provided as supplementary file"

$contact_address
[1] "715 Albany Street, E613B"

$contact_city
[1] "Boston"

$contact_country
[1] "USA"

$contact_department
[1] "Genetics and Genomics"

$contact_email
[1] "mleburg@bu.edu"

$contact_fax
[1] "617-414-1646"

$contact_institute
[1] "Boston University School of Medicine"

$contact_name
[1] "Marc,E.,Lenburg"

$contact_phone
[1] "617-414-1375"

$contact_state
[1] "MA"

$contact_web_link
[1] "http://gg.bu.edu"

$`contact_zip/postal_code`
[1] "02130"
```

\$data_row_count

[1] "22283"

\$description

[1] "Age = 70; Gender = Female; Right Kidney; Adjacent Tumor Type = clear cell; Adjacent

[2] "Keywords = kidney"

[3] "Keywords = renal"

[4] "Keywords = RCC"

[5] "Keywords = carcinoma"

[6] "Keywords = cancer"

[7] "Lot batch = 2004638"

\$geo_accession

[1] "GSM11805"

\$last_update_date

[1] "May 28 2005"

\$molecule_ch1

[1] "total RNA"

\$organism_ch1

[1] "Homo sapiens"

\$platform_id

[1] "GPL96"

\$series_id

[1] "GSE781"

\$source_name_ch1

[1] "Trizol isolation of total RNA from normal tissue adjacent to Renal Cell Carcinoma"

\$status

[1] "Public on Nov 25 2003"

\$submission_date

[1] "Oct 20 2003"

\$supplementary_file

[1] "ftp://ftp.ncbi.nih.gov/pub/geo/DATA/supplementary/samples/GSM11nnn/GSM11805/GSM1180

```
$title
[1] "N035 Normal Human Kidney U133A"
```

```
$type
[1] "RNA"
```

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

	ID_REF	VALUE	ABS_CALL
1	AFFX-BioB-5_at	953.9	P
2	AFFX-BioB-M_at	2982.8	P
3	AFFX-BioB-3_at	1657.9	P
4	AFFX-BioC-5_at	2652.7	P
5	AFFX-BioC-3_at	2019.5	P

```
> Columns(gsm)
```

	Column
1	ID_REF
2	VALUE
3	ABS_CALL

	Description
1	
2	MAS 5.0 Statistical Algorithm (mean scaled to 500)
3	MAS 5.0 Absent, Marginal, Present call with Alpha1 = 0.05, Alpha2 = 0.065

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

```
> Columns(gds)
```

	sample	disease.state	individual
1	GSM11815	RCC	035
2	GSM11832	RCC	023
3	GSM12069	RCC	001
4	GSM12083	RCC	005
5	GSM12101	RCC	011
6	GSM12106	RCC	032
7	GSM12274	RCC	2
8	GSM12299	RCC	3
9	GSM12412	RCC	4
10	GSM11810	normal	035
11	GSM11827	normal	023
12	GSM12078	normal	001

13	GSM12099	normal	005
14	GSM12269	normal	1
15	GSM12287	normal	2
16	GSM12301	normal	3
17	GSM12448	normal	4

1	Value for GSM11815: C035 Renal Clear Cell Carcinoma U133B; src: Trizol iso
2	Value for GSM11832: C023 Renal Clear Cell Carcinoma U133B; src: Trizol iso
3	Value for GSM12069: C001 Renal Clear Cell Carcinoma U133B; src: Trizol iso
4	Value for GSM12083: C005 Renal Clear Cell Carcinoma U133B; src: Trizol iso
5	Value for GSM12101: C011 Renal Clear Cell Carcinoma U133B; src: Trizol iso
6	Value for GSM12106: C032 Renal Clear Cell Carcinoma U133B; src: Trizol iso
7	Value for GSM12274: C2 Renal Clear Cell Carcinoma U133B; src: Trizol iso
8	Value for GSM12299: C3 Renal Clear Cell Carcinoma U133B; src: Trizol iso
9	Value for GSM12412: C4 Renal Clear Cell Carcinoma U133B; src: Trizol iso
10	Value for GSM11810: N035 Normal Human Kidney U133B; src: Trizol isolation of tot
11	Value for GSM11827: N023 Normal Human Kidney U133B; src: Trizol isolation of tot
12	Value for GSM12078: N001 Normal Human Kidney U133B; src: Trizol isolation of tot
13	Value for GSM12099: N005 Normal Human Kidney U133B; src: Trizol isolation of tot
14	Value for GSM12269: N1 Normal Human Kidney U133B; src: Trizol isolation of tot
15	Value for GSM12287: N2 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of tot
16	Value for GSM12301: N3 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of tot
17	Value for GSM12448: N4 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of tot

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 *GEODataTable*. 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(filename = system.file("extdata/GSE781_family.soft.gz",
+   package = "GEOquery"))
```

```
Parsing....
```

```
Found 36 entities...
```

```
GPL96 (1 of 36 entities)
```

```
GPL97 (2 of 36 entities)
```

```
GSM11805 (3 of 36 entities)
```

```
GSM11810 (4 of 36 entities)
```

```
GSM11814 (5 of 36 entities)
```

```
GSM11815 (6 of 36 entities)
```

GSM11823 (7 of 36 entities)
GSM11827 (8 of 36 entities)
GSM11830 (9 of 36 entities)
GSM11832 (10 of 36 entities)
GSM12067 (11 of 36 entities)
GSM12069 (12 of 36 entities)
GSM12075 (13 of 36 entities)
GSM12078 (14 of 36 entities)
GSM12079 (15 of 36 entities)
GSM12083 (16 of 36 entities)
GSM12098 (17 of 36 entities)
GSM12099 (18 of 36 entities)
GSM12100 (19 of 36 entities)
GSM12101 (20 of 36 entities)
GSM12105 (21 of 36 entities)
GSM12106 (22 of 36 entities)
GSM12268 (23 of 36 entities)
GSM12269 (24 of 36 entities)
GSM12270 (25 of 36 entities)
GSM12274 (26 of 36 entities)
GSM12283 (27 of 36 entities)
GSM12287 (28 of 36 entities)
GSM12298 (29 of 36 entities)
GSM12299 (30 of 36 entities)
GSM12300 (31 of 36 entities)
GSM12301 (32 of 36 entities)
GSM12399 (33 of 36 entities)
GSM12412 (34 of 36 entities)
GSM12444 (35 of 36 entities)
GSM12448 (36 of 36 entities)

> *Meta(gse)*

\$email

[1] "geo@ncbi.nlm.nih.gov"

\$institute

[1] "NCBI NLM NIH"

\$name

[1] "Gene Expression Omnibus (GEO)"

\$web_link

[1] "<http://www.ncbi.nlm.nih.gov/projects/geo>"


```

> names(GSMList(gse))

 [1] "GSM11805" "GSM11810" "GSM11814" "GSM11815" "GSM11823" "GSM11827"
 [7] "GSM11830" "GSM11832" "GSM12067" "GSM12069" "GSM12075" "GSM12078"
[13] "GSM12079" "GSM12083" "GSM12098" "GSM12099" "GSM12100" "GSM12101"
[19] "GSM12105" "GSM12106" "GSM12268" "GSM12269" "GSM12270" "GSM12274"
[25] "GSM12283" "GSM12287" "GSM12298" "GSM12299" "GSM12300" "GSM12301"
[31] "GSM12399" "GSM12412" "GSM12444" "GSM12448"

> GSMList(gse)[[1]]

An object of class "GSM"

NULL
An object of class "GEODataTable"
***** Column Descriptions *****
      Column
1  ID_REF
2  VALUE
3 ABS_CALL

                                                    Description
1
2                MAS 5.0 Statistical Algorithm (mean scaled to 500)
3 MAS 5.0 Absent, Marginal, Present call with Alpha1 = 0.05, Alpha2 = 0.065
***** Data Table *****
      ID_REF  VALUE  ABS_CALL
1 AFX-BioB-5_at  953.9      P
2 AFX-BioB-M_at 2982.8      P
3 AFX-BioB-3_at 1657.9      P
4 AFX-BioC-5_at 2652.7      P
5 AFX-BioC-3_at 2019.5      P
22272 more rows ...

> names(GPLList(gse))

```

See below for an additional, preferred method of obtaining GSE information.

4 Converting to BioConductor ExpressionSets 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 *ExpressionSet*. Therefore, there are two functions, *GDS2MA* and *GDS2eSet* that accomplish that task.

4.1 Getting GSE Series Matrix files as an ExpressionSet

GEO Series are collections of related experiments. In addition to being available as SOFT format files, which are quite large, NCBI GEO has prepared a simpler format file based on tab-delimited text. The `getGEO` function can handle this format and will parse very large GSEs quite quickly. The data structure returned from this parsing is a list of ExpressionSets. As an example, we download and parse GSE2553.

```
> gse2553 <- getGEO("GSE2553", GSEMatrix = TRUE)

Found 1 file(s)
GSE2553_series_matrix.txt.gz
File stored at:
/var/folders/F+/F+PwkbXqF6WeunvinD8pZk+++TI/-Tmp-//RtmpLfNSbM/GPL1977.soft

> show(gse2553)

$GSE2553_series_matrix.txt.gz
ExpressionSet (storageMode: lockedEnvironment)
assayData: 12600 features, 181 samples
  element names: exprs
phenoData
  sampleNames: GSM48681, GSM48682, ..., GSM48861 (181 total)
  varLabels and varMetadata description:
    title: NA
    geo_accession: NA
    ...: ...
    data_row_count: NA
    (27 total)
featureData
  featureNames: 1, 2, ..., 12600 (12600 total)
  fvarLabels and fvarMetadata description:
    ID: NA
    PenAt: NA
    ...: ...
    Chimeric_Cluster_IDs: NA
    (13 total)
  additional fvarMetadata: Column, Description
experimentData: use 'experimentData(object)'
Annotation: GPL1977

> show(pData(phenoData(gse2553[[1]]))[1:5, c(1, 6, 8)])

                                title type
GSM48681 Patient sample ST18, Dermatofibrosarcoma RNA
```

GSM48682	Patient sample ST410, Ewing Sarcoma	RNA
GSM48683	Patient sample ST130, Sarcoma, NOS	RNA
GSM48684	Patient sample ST293, Malignant Peripheral Nerve Sheath Tumor	RNA
GSM48685	Patient sample ST367, Liposarcoma	RNA
	source_name_ch1	
GSM48681	Dermatofibrosarcoma	
GSM48682	Ewing Sarcoma	
GSM48683	Sarcoma, NOS	
GSM48684	Malignant Peripheral Nerve Sheath Tumor	
GSM48685	Liposarcoma	

4.2 Converting GDS to an ExpressionSet

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

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

File stored at:

```
/var/folders/F+/F+PwkbXqF6WeunvinD8pZk+++TI/-Tmp-//RtmpLfNSbM/GPL97.annot
```

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

```
> eset
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 22645 features, 17 samples
  element names: exprs
phenoData
  sampleNames: GSM11815, GSM11832, ..., GSM12448 (17 total)
  varLabels and varMetadata description:
    sample: NA
    disease.state: NA
    individual: NA
    description: NA
featureData
  featureNames: 200000_s_at, 200001_at, ..., AFFX-TrpnX-M_at (22645 total)
  fvarLabels and fvarMetadata description:
    ID: ID from Platform data table
    Gene.title: Entrez Gene name
    ...: ...
    GO.Component.1: Gene Ontology Component identifier
    (21 total)
  additional fvarMetadata: Column
```

```

experimentData: use 'experimentData(object)'
  pubMedIds: 14641932
Annotation:

```

```
> pData(eset)
```

	sample	disease.state	individual
GSM11815	GSM11815	RCC	035
GSM11832	GSM11832	RCC	023
GSM12069	GSM12069	RCC	001
GSM12083	GSM12083	RCC	005
GSM12101	GSM12101	RCC	011
GSM12106	GSM12106	RCC	032
GSM12274	GSM12274	RCC	2
GSM12299	GSM12299	RCC	3
GSM12412	GSM12412	RCC	4
GSM11810	GSM11810	normal	035
GSM11827	GSM11827	normal	023
GSM12078	GSM12078	normal	001
GSM12099	GSM12099	normal	005
GSM12269	GSM12269	normal	1
GSM12287	GSM12287	normal	2
GSM12301	GSM12301	normal	3
GSM12448	GSM12448	normal	4

```

GSM11815      Value for GSM11815: C035 Renal Clear Cell Carcinoma U133B; src: Triz
GSM11832      Value for GSM11832: C023 Renal Clear Cell Carcinoma U133B; src: Triz
GSM12069      Value for GSM12069: C001 Renal Clear Cell Carcinoma U133B; src: Triz
GSM12083      Value for GSM12083: C005 Renal Clear Cell Carcinoma U133B; src: Triz
GSM12101      Value for GSM12101: C011 Renal Clear Cell Carcinoma U133B; src: Triz
GSM12106      Value for GSM12106: C032 Renal Clear Cell Carcinoma U133B; src: Triz
GSM12274      Value for GSM12274: C2 Renal Clear Cell Carcinoma U133B; src: Triz
GSM12299      Value for GSM12299: C3 Renal Clear Cell Carcinoma U133B; src: Triz
GSM12412      Value for GSM12412: C4 Renal Clear Cell Carcinoma U133B; src: Triz
GSM11810      Value for GSM11810: N035 Normal Human Kidney U133B; src: Trizol isolation
GSM11827      Value for GSM11827: N023 Normal Human Kidney U133B; src: Trizol isolation
GSM12078      Value for GSM12078: N001 Normal Human Kidney U133B; src: Trizol isolation
GSM12099      Value for GSM12099: N005 Normal Human Kidney U133B; src: Trizol isolation
GSM12269      Value for GSM12269: N1 Normal Human Kidney U133B; src: Trizol isolation
GSM12287      Value for GSM12287: N2 Renal Clear Cell Carcinoma U133B; src: Trizol isolation
GSM12301      Value for GSM12301: N3 Renal Clear Cell Carcinoma U133B; src: Trizol isolation
GSM12448      Value for GSM12448: N4 Renal Clear Cell Carcinoma U133B; src: Trizol isolation

```

4.3 Converting GDS to an MAList

No annotation information (called platform information by GEO) was retrieved from because *ExpressionSet* 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] "GPL97"
```

```
> gpl <- getGEO(filename = system.file("extdata/GPL97.annot.gz",  
+   package = "GEOquery"))
```

So, `gpl` now contains the information for GPL5 from GEO. Unlike *ExpressionSet*, 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)  
> MA
```

An object of class "MAList"

\$M

	GSM11815	GSM11832	GSM12069	GSM12083	GSM12101	GSM12106	GSM12274	GSM12299
[1,]	4254.0	5298.2	4026.5	3498.4	3566.4	4903.1	6372.6	4829.1
[2,]	17996.2	12010.7	10283.5	2534.7	11048.4	13354.0	8563.8	17247.6
[3,]	41678.8	39116.9	38758.9	32847.7	39633.9	43511.2	46856.7	47032.4
[4,]	65390.9	34806.2	31257.2	28308.5	67447.5	56989.9	57972.5	57570.5
[5,]	19030.1	15813.6	16355.7	9579.7	14273.5	17217.0	19116.9	17487.6
	GSM12412	GSM11810	GSM11827	GSM12078	GSM12099	GSM12269	GSM12287	GSM12301
[1,]	5205.8	2756.8	3932.0	3729.9	3223.4	3640.5	4886.3	4070.2
[2,]	16018.5	6077.0	15703.8	10138.5	11614.4	8460.5	10282.6	11844.3
[3,]	22152.2	26660.7	26373.6	23809.6	24749.3	21936.8	31462.8	22733.7
[4,]	29062.2	35140.9	23629.3	22100.5	21651.0	18550.7	23496.5	21315.4
[5,]	14671.6	17733.1	18022.4	17957.4	15958.0	15799.8	16685.8	18817.3
	GSM12448							
[1,]	3482.1							
[2,]	9741.6							
[3,]	25395.5							
[4,]	28631.4							
[5,]	17421.1							

22640 more rows ...

\$A

NULL

\$targets

	sample	disease.state	individual
1	GSM11815	RCC	035
2	GSM11832	RCC	023
3	GSM12069	RCC	001
4	GSM12083	RCC	005
5	GSM12101	RCC	011

1 Value for GSM11815: C035 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of to
2 Value for GSM11832: C023 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of to
3 Value for GSM12069: C001 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of to
4 Value for GSM12083: C005 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of to
5 Value for GSM12101: C011 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of to
12 more rows ...

\$genes

	ID	Gene.title
1	200000_s_at PRP8	pre-mRNA processing factor 8 homolog (S. cerevisiae)
2	200001_at	calpain, small subunit 1
3	200002_at	ribosomal protein L35
4	200003_s_at	ribosomal protein L28
5	200004_at	eukaryotic translation initiation factor 4 gamma, 2

	Gene.symbol	Gene.ID	UniGene.title	UniGene.symbol	UniGene.ID
1	PRPF8	10594			
2	CAPNS1	826			
3	RPL35	11224			
4	RPL28	6158			
5	EIF4G2	1982			

1		Homo sapiens PRP8 pre-mRNA processing factor 8 homolog (S. cerevisiae)			
2		Homo sapiens calpain, small subunit 1 (CAPNS1), transcript			
3		Homo sapiens ribosomal protein L35			
4		Homo sapiens ribosomal protein L28			
5		Homo sapiens eukaryotic translation initiation factor 4 gamma, 2 (EIF4G2), transcript			

	GI	GenBank.Accession	Platform_CLONEID	Platform_ORF	Platform_SPOTID
1	91208425	NM_006445	<NA>	<NA>	<NA>
2	51599152	NM_001749	<NA>	<NA>	<NA>
3	78190471	NM_007209	<NA>	<NA>	<NA>
4	34486095	NM_000991	<NA>	<NA>	<NA>
5	111494227	NM_001418	<NA>	<NA>	<NA>

Chromosome.location

1 17p13.3

2 19q13.12

3 9q34.1

4 19q13.4

5 11p15

Chromosome.annotation

1 Chromosome 17, NC_000017.9 (1500673..1534926, complement)

2 Chromosome 19, NC_000019.8 (41322758..41333095)

3 Chromosome 9, NC_000009.10 (126659979..126664061, complement)

4 Chromosome 19, NC_000019.8 (60589112..60595265)

5 Chromosome 11, NC_000011.8 (10775169..10787158, complement)

1 RNA binding///RNA splicing factor activity, transesterificatio

2 calcium ion binding///calcium-dependent cysteine-type endopeptida

3 mRNA binding///protein binding///struc

4 RNA binding///protein binding///structural constituent of ribosome///struc

5 protein binding///protein binding///translation initiation factor activity///translati

1 RNA splicing///nuclear mRNA splicing, via spliceosome///nuclear mRNA splicing, via spl

2

3

4

5 RNA metabolic process///cell cycle arrest///

GO.Component

1 nuclear speck///nucleus///snRNP U5///spliceosome

2 cytoplasm///plasma membrane

3 cytosol///cytosolic large ribosomal subunit///intracellular///nucleolus///ribosome

4 cytosol///cytosolic large ribosomal subunit///intracellular///ribosome

5 eukaryotic translation initiation factor 4F complex

GO.Function.1

1 GO:0003723///GO:0031202///GO:0005515

2 GO:0005509///GO:0004198///GO:0005515

3 GO:0003729///GO:0005515///GO:0003735

4 GO:0003723///GO:0005515///GO:0003735///GO:0003735

5 GO:0005515///GO:0005515///GO:0003743///GO:0003743

GO.Process.1

1 GO:0008380///GO:0000398///GO:0000398///GO:0050896///GO:0007601

2 GO:0008284

3 GO:0006414

4 GO:0006412///GO:0006414

5 GO:0016070///GO:0007050///GO:0008219///GO:0006446

GO.Component.1
1 GO:0016607///GO:0005634///GO:0005682///GO:0005681
2 GO:0005737///GO:0005886
3 GO:0005829///GO:0022625///GO:0005622///GO:0005730///GO:0005840
4 GO:0005829///GO:0022625///GO:0005622///GO:0005840
5 GO:0016281
22640 more rows ...

\$notes

\$

NULL

\$channel_count

[1] "1"

\$description

[1] "Investigation into mechanisms of renal clear cell carcinogenesis (RCC). Comparison

\$feature_count

[1] "22645"

\$order

[1] "none"

\$platform

[1] "GPL97"

\$platform_organism

[1] "Homo sapiens"

\$platform_technology_type

[1] "in situ oligonucleotide"

\$pubmed_id

[1] "14641932"

\$reference_series

[1] "GSE781"

\$sample_count

[1] "17"


```

$sample_organism
[1] "Homo sapiens"

$sample_type
[1] "RNA"

$title
[1] "Renal clear cell carcinoma (HG-U133B)"

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

$update_date
[1] "Mar 04 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 GDS507.

4.4 Converting GSE to an ExpressionSet

First, make sure that using the method described above in the section “Getting GSE Series Matrix files as an ExpressionSet” for using GSE Series Matrix files is not sufficient for the task, as it is much faster and simpler. If it is not (i.e., other columns from each GSM are needed), then this method will be needed.

Converting a *GSE* object to an *ExpressionSet* 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

$GSM11805
NULL

$GSM11810
NULL

```

\$GSM11814
NULL

\$GSM11815
NULL

\$GSM11823
NULL

\$GSM11827
NULL

\$GSM11830
NULL

\$GSM11832
NULL

\$GSM12067
NULL

\$GSM12069
NULL

\$GSM12075
NULL

\$GSM12078
NULL

\$GSM12079
NULL

\$GSM12083
NULL

\$GSM12098
NULL

\$GSM12099
NULL

\$GSM12100
NULL

\$GSM12101
NULL

\$GSM12105
NULL

\$GSM12106
NULL

\$GSM12268
NULL

\$GSM12269
NULL

\$GSM12270
NULL

\$GSM12274
NULL

\$GSM12283
NULL

\$GSM12287
NULL

\$GSM12298
NULL

\$GSM12299
NULL

\$GSM12300
NULL

\$GSM12301
NULL

\$GSM12399

NULL

\$GSM12412

NULL

\$GSM12444

NULL

\$GSM12448

NULL

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	VALUE	ABS_CALL
1	AFFX-BioB-5_at	953.9	P
2	AFFX-BioB-M_at	2982.8	P
3	AFFX-BioB-3_at	1657.9	P
4	AFFX-BioC-5_at	2652.7	P
5	AFFX-BioC-3_at	2019.5	P

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

	Column
1	ID_REF
2	VALUE
3	ABS_CALL
NA	<NA>
NA.1	<NA>

Description

1	
2	MAS 5.0 Statistical Algorithm (mean scaled to 500)
3	MAS 5.0 Absent, Marginal, Present call with Alpha1 = 0.05, Alpha2 = 0.065
NA	<NA>
NA.1	<NA>

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

```

> probesets <- Table(GPLList(gse)[[1]])$ID
> data.matrix <- do.call("cbind", lapply(GSMList(gse), function(x) {
+   tab <- Table(x)
+   mymatch <- match(probesets, tab$ID_REF)
+   return(tab$VALUE[mymatch])
+ })))
> data.matrix <- apply(data.matrix, 2, function(x) {
+   as.numeric(as.character(x))
+ })
> data.matrix <- log2(data.matrix)
> data.matrix[1:5, ]

```

	GSM11805	GSM11810	GSM11814	GSM11815	GSM11823	GSM11827	GSM11830
[1,]	10.926963	NA	11.105254	NA	11.275019	NA	11.438636
[2,]	5.749534	NA	7.908092	NA	7.093814	NA	7.514122
[3,]	7.066089	NA	7.750205	NA	7.244126	NA	7.962896
[4,]	12.660353	NA	12.479755	NA	12.215897	NA	11.458355
[5,]	6.195741	NA	6.061776	NA	6.565293	NA	6.583459
	GSM11832	GSM12067	GSM12069	GSM12075	GSM12078	GSM12079	GSM12083
[1,]	NA	11.424376	NA	11.222795	NA	11.469845	NA
[2,]	NA	7.901470	NA	6.407693	NA	5.165912	NA
[3,]	NA	7.337176	NA	6.569856	NA	7.477354	NA
[4,]	NA	11.397568	NA	12.529870	NA	12.240046	NA
[5,]	NA	6.877744	NA	6.652486	NA	3.981853	NA
	GSM12098	GSM12099	GSM12100	GSM12101	GSM12105	GSM12106	GSM12268
[1,]	10.823367	NA	10.835971	NA	10.810893	NA	11.062653
[2,]	6.556123	NA	8.207014	NA	6.816344	NA	6.563768
[3,]	7.708739	NA	7.428779	NA	7.754888	NA	7.126188
[4,]	12.336534	NA	11.762839	NA	11.237509	NA	12.412490
[5,]	5.501439	NA	6.247928	NA	6.017922	NA	6.525129
	GSM12269	GSM12270	GSM12274	GSM12283	GSM12287	GSM12298	GSM12299
[1,]	NA	10.323055	NA	11.181028	NA	11.566387	NA
[2,]	NA	7.353147	NA	5.770829	NA	6.912889	NA
[3,]	NA	8.742815	NA	7.339850	NA	7.602142	NA
[4,]	NA	11.213408	NA	12.678380	NA	12.232901	NA
[5,]	NA	6.683696	NA	5.918863	NA	5.837943	NA
	GSM12300	GSM12301	GSM12399	GSM12412	GSM12444	GSM12448	
[1,]	11.078151	NA	11.535178	NA	11.105450	NA	
[2,]	4.812498	NA	7.471675	NA	7.488644	NA	
[3,]	7.383704	NA	7.432959	NA	7.381110	NA	

```
[4,] 12.090939      NA 11.421802      NA 12.172834      NA
[5,]  6.281698      NA  5.419539      NA  5.469235      NA
```

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 *ExpressionSet* object:

```
> require(Biobase)
> rownames(data.matrix) <- probesets
> colnames(data.matrix) <- names(GSMList(gse))
> pdata <- data.frame(samples = names(GSMList(gse)))
> rownames(pdata) <- names(GSMList(gse))
> pheno <- as(pdata, "AnnotatedDataFrame")
> eset2 <- new("ExpressionSet", exprs = data.matrix, phenoData = pheno)
> eset2
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 22277 features, 34 samples
  element names: exprs
phenoData
  sampleNames: GSM11805, GSM11810, ..., GSM12448 (34 total)
  varLabels and varMetadata description:
    samples: NA
featureData
  featureNames: 1007_s_at, 1053_at, ..., AFFX-r2-P1-cre-5_at (22277 total)
  fvarLabels and fvarMetadata description: none
experimentData: use 'experimentData(object)'
Annotation:
```

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.

5 Accessing Raw Data from GEO

NCBI GEO accepts (but has not always required) raw data such as .CEL files, .CDF files, images, etc. Sometimes, it is useful to get quick access to such data. A single function, `getGEOSuppFiles`, can take as an argument a GEO accession and will download all the raw data associate with that accession. By default, the function will create a directory in the current working directory to store the raw data for the chosen GEO accession. Combining

a simple `sapply` statement or other loop structure with `getGEOSuppFiles` makes for a very simple way to get gobs of raw data quickly and easily without needing to know the specifics of GEO raw data URLs.

6 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.

7 sessionInfo

- R version 2.11.0 Under development (unstable) (2009-11-13 r50424),
i386-apple-darwin10.2.0
- Locale: en_US/en_US/C/C/en_US/en_US
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: Biobase 2.7.0, bitops 1.0-4.1, GEOquery 2.11.0, limma 3.3.1,
RCurl 1.3-0