

Using the GEOquery package

Sean Davis^{‡*}

July 10, 2014

[‡]Genetics Branch
National Cancer Institute
National Institutes of Health

Contents

1	Overview of GEO	2
1.1	Platforms	2
1.2	Samples	2
1.3	Series	2
1.4	Datasets	3
2	Getting Started using GEOquery	3
3	GEOquery Data Structures	3
3.1	The GDS, GSM, and GPL classes	4
3.2	The GSE class	7
4	Converting to BioConductor ExpressionSets and limma MALists	13
4.1	Getting GSE Series Matrix files as an ExpressionSet	13
4.2	Converting GDS to an ExpressionSet	14
4.3	Converting GDS to an MAList	15
4.4	Converting GSE to an ExpressionSet	21
5	Accessing Raw Data from GEO	26
6	Use Cases	27
6.1	Getting all Series Records for a Given Platform	27
7	Conclusion	28

*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).

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

1.4 Datasets

GEO DataSets (GDSxxxx) 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.

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.

```
> # If you have network access, the more typical way to do this
> # would be to use this:
> # gds <- getGEO("GDS507")
> 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.

```
> # If you have network access, the more typical way to do this
> # would be to use this:
> # gds <- getGEO("GSM11805")
> 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:

```
> # Look at gsm metadata:
> 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] "mlenburg@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"
```

```
> # Look at data associated with the GSM:  
> # but restrict to only first 5 rows, for brevity  
> 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

```
> # Look at Column descriptions:  
> Columns(gsm)
```

	Column	Description
1	ID_REF	
2	VALUE	MAS 5.0 Statistical Algorithm (mean scaled to 500)
3	ABS_CALL	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:

```
> # Again, with good network access, one would do:  
> # gse <- getGEO("GSE781",GSEMatrix=FALSE)  
> gse <- getGEO(filename=system.file("extdata/GSE781_family.soft.gz",package="GEOquery
```

```
Parsing....
```

```
> Meta(gse)
```

```
$contact_address
```

```
[1] "715 Albany Street, E613B"
```

```
$contact_city
```

```
[1] "Boston"
```

```
$contact_country
```

```
[1] "USA"
```

```
$contact_department
```

```
[1] "Genetics and Genomics"
```

```
$contact_email
```

```
[1] "mlenburg@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"
```


\$contributor

[1] "Marc,E,Lenburg" "Louis,S,Liou" "Norman,P,Gerry"
[4] "Garrett,M,Frampton" "Herbert,T,Cohen" "Michael,F,Christman"

\$email

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

\$geo_accession

[1] "GSE781"

\$institute

[1] "NCBI NLM NIH"

\$last_update_date

[1] "May 29 2005"

\$name

[1] "Gene Expression Omnibus (GEO)"

\$platform_id

[1] "GPL96" "GPL97"

\$pubmed_id

[1] "14641932"

\$sample_id

[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"

\$status

[1] "Public on Nov 25 2003"

\$submission_date

[1] "Oct 24 2003"

\$summary

[1] "Each total RNA sample is hybridized to two different arrays: Affymetrix U133A (GP

```

[2] ""
[3] "For most of the normal tissue samples there is a renal clear cell carcinoma sample"
[4] ""
[5] "For most of the renal clear cell carcinoma samples there is a corresponding adjace"
[6] "Keywords = kidney"
[7] "Keywords = renal"
[8] "Keywords = RCC"
[9] "Keywords = carcinoma"
[10] "Keywords = cancer"
[11] "Keywords: parallel sample"

```

\$supplementary_file

```
[1] "ftp://ftp.ncbi.nih.gov/pub/geo/DATA/supplementary/series/GSE781/GSE781_RAW.tar"
```

\$title

```
[1] "Normal and Renal Cell Carcinoma Kidney Tissue, Human"
```

\$type

```
[1] "Expression profiling by array"
```

\$web_link

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

> # names of all the GSM objects contained in the GSE

> 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"

```

> # and get the first GSM object on the list

> GSMList(gse)[[1]]

An object of class "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] "mlenburg@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"
An object of class "GEODataTable"
***** Column Descriptions *****
      Column
1  ID_REF
2  VALUE
3 ABS_CALL

                                                    Description
1
2
3 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
22278 more rows ...

> # and the names of the GPLs represented
> names(GPLList(gse))

[1] "GPL96" "GPL97"

```

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.

```
> # Note that GSEMatrix=TRUE is the default
> gse2553 <- getGEO('GSE2553',GSEMatrix=TRUE)
> show(gse2553)

$GSE2553_series_matrix.txt.gz
ExpressionSet (storageMode: lockedEnvironment)
assayData: 12600 features, 181 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: GSM48681 GSM48682 ... GSM48861 (181 total)
  varLabels: title geo_accession ... data_row_count (30 total)
  varMetadata: labelDescription
featureData
  featureNames: 1 2 ... 12600 (12600 total)
  fvarLabels: ID PenAt ... Chimeric_Cluster_IDs (13 total)
  fvarMetadata: Column Description labelDescription
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)
```

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
protocolData: none
phenoData
  sampleNames: GSM11815 GSM11832 ... GSM12448 (17 total)
  varLabels: sample disease.state individual description
  varMetadata: labelDescription
featureData
  featureNames: 200000_s_at 200001_at ... AFFX-TrpnX-M_at (22645 total)
  fvarLabels: ID Gene title ... GO:Component ID (21 total)
  fvarMetadata: Column labelDescription
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.

```
> #get the platform from the GDS metadata
> Meta(gds)$platform
```

```
[1] "GPL97"
```

```
> #So use this information in a call to getGEO
> 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	Gene symbol
1 200000_s_at	pre-mRNA processing factor 8	PRPF8
2 200001_at	calpain, small subunit 1	CAPNS1
3 200002_at	ribosomal protein L35	RPL35
4 200003_s_at	ribosomal protein L28	RPL28
5 200004_at	eukaryotic translation initiation factor 4 gamma, 2	EIF4G2

Gene ID	UniGene title	UniGene symbol	UniGene ID
1 10594	<NA>	<NA>	<NA>
2 826	<NA>	<NA>	<NA>
3 11224	<NA>	<NA>	<NA>
4 6158	<NA>	<NA>	<NA>
5 1982	<NA>	<NA>	<NA>

1 Homo sapiens pre-mRNA processing factor
 2 Homo sapiens calpain, small subunit 1 (CAPNS1), transcript
 3 Homo sapiens ribosomal protein L35 (RPL35), transcript
 4 Homo sapiens ribosomal protein L28 (RPL28), transcript
 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 209915582	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_000017.10 (1553923..1588176, complement)
2	Chromosome 19, NC_000019.9 (36630918..36641255)
3	Chromosome 9, NC_000009.11 (127620158..127624240, complement)
4	Chromosome 19, NC_000019.9 (55897300..55903453)
5	Chromosome 11, NC_000011.9 (10818593..10830582, complement)

```

1
2
3
4
5 DNA binding///protein binding///translation factor activity, nucleic acid binding///tr

```

```

1
2
3 RNA metabolic process///SRP-dependent cotranslational protein targeting to membrane///
4 RNA metabolic process///SRP-dependent cotranslational protein targeting to membrane///
5

```

```

GO:Component
1 U5 snRNP///catalytic step 2 spliceosome///nuclear speck///nucleoplasm
2
3 cytoplasm///cytosol///cytosolic large ribosomal subunit///nucleolus
4 cytoplasm///cytosol///cytosolic large ribosomal subunit
5 cytosol///eukaryotic translation initiation factor 4F complex

```

```

GO:Function ID
1 GO:0030623///GO:0017070///GO:0005515
2 GO:0005509///GO:0004198///GO:0005515
3 GO:0003729///GO:0003735
4 GO:0003723///GO:0005515///GO:0003735
5 GO:0003677///GO:0005515///GO:0008135///GO:0003743

```

```

1
2
3 GO:0016070///GO:0006614///GO:0044267///GO:0010467///GO:0016071///GO:0000184///GO:00064
4 GO:0016070///GO:0006614///GO:0044267///GO:0010467///GO:0016071///GO:0000184///GO:00064
5

```

```

GO:Component ID
1 GO:0005682///GO:0071013///GO:0016607///GO:0005654
2 GO:0005737///GO:0005886
3 GO:0005737///GO:0005829///GO:0022625///GO:0005730
4 GO:0005737///GO:0005829///GO:0022625
5 GO:0005829///GO:0016281

```

22640 more rows ...

\$notes

\$channel_count

[1] "1"

\$dataset_id

[1] "GDS507" "GDS507" "GDS507" "GDS507" "GDS507" "GDS507" "GDS507" "GDS507"
[9] "GDS507" "GDS507" "GDS507" "GDS507"

\$description

[1] "Investigation into mechanisms of renal clear cell carcinogenesis (RCC). Comparison
[2] "RCC"
[3] "normal"
[4] "035"
[5] "023"
[6] "001"
[7] "005"
[8] "011"
[9] "032"
[10] "1"
[11] "2"
[12] "3"
[13] "4"

\$email

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

\$feature_count

[1] "22645"

\$institute

[1] "NCBI NLM NIH"

\$name

[1] "Gene Expression Omnibus (GEO)"

\$order

[1] "none"

\$platform

[1] "GPL97"

\$platform_organism

[1] "Homo sapiens"

\$platform_technology_type

[1] "in situ oligonucleotide"

\$pubmed_id

[1] "14641932"

\$ref

[1] "Nucleic Acids Res. 2005 Jan 1;33 Database Issue:D562-6"

\$reference_series

[1] "GSE781"

\$sample_count

[1] "17"

\$sample_id

[1] "GSM11815,GSM11832,GSM12069,GSM12083,GSM12101,GSM12106,GSM12274,GSM12299,GSM12412"
[2] "GSM11810,GSM11827,GSM12078,GSM12099,GSM12269,GSM12287,GSM12301,GSM12448"
[3] "GSM11810,GSM11815"
[4] "GSM11827,GSM11832"
[5] "GSM12069,GSM12078"
[6] "GSM12083,GSM12099"
[7] "GSM12101"
[8] "GSM12106"
[9] "GSM12269"
[10] "GSM12274,GSM12287"
[11] "GSM12299,GSM12301"
[12] "GSM12412,GSM12448"

\$sample_organism

[1] "Homo sapiens"

\$sample_type

[1] "RNA"

\$title

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

\$type

[1] "gene expression array-based" "disease state"
[3] "disease state" "individual"
[5] "individual" "individual"
[7] "individual" "individual"
[9] "individual" "individual"
[11] "individual" "individual"

```
[13] "individual"

$update_date
[1] "Mar 04 2004"

$value_type
[1] "count"

$web_link
[1] "http://www.ncbi.nlm.nih.gov/projects/geo"
```

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
[1] "GPL96"

$GSM11810
[1] "GPL97"

$GSM11814
[1] "GPL96"

$GSM11815
[1] "GPL97"

$GSM11823
[1] "GPL96"
```

\$GSM11827
[1] "GPL97"

\$GSM11830
[1] "GPL96"

\$GSM11832
[1] "GPL97"

\$GSM12067
[1] "GPL96"

\$GSM12069
[1] "GPL97"

\$GSM12075
[1] "GPL96"

\$GSM12078
[1] "GPL97"

\$GSM12079
[1] "GPL96"

\$GSM12083
[1] "GPL97"

\$GSM12098
[1] "GPL96"

\$GSM12099
[1] "GPL97"

\$GSM12100
[1] "GPL96"

\$GSM12101
[1] "GPL97"

\$GSM12105
[1] "GPL96"

\$GSM12106
[1] "GPL97"

\$GSM12268
[1] "GPL96"

\$GSM12269
[1] "GPL97"

\$GSM12270
[1] "GPL96"

\$GSM12274
[1] "GPL97"

\$GSM12283
[1] "GPL96"

\$GSM12287
[1] "GPL97"

\$GSM12298
[1] "GPL96"

\$GSM12299
[1] "GPL97"

\$GSM12300
[1] "GPL96"

\$GSM12301
[1] "GPL97"

\$GSM12399
[1] "GPL96"

\$GSM12412
[1] "GPL97"

\$GSM12444
[1] "GPL96"

```
$GSM12448
[1] "GPL97"
```

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

```
> # and get the column descriptions
```

```
> 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:

```
> # get the probeset ordering
```

```
> probesets <- Table(GPLList(gse)[[1]])$ID
```

```
> # make the data matrix from the VALUE columns from each GSM
```

```
> # being careful to match the order of the probesets in the platform
```



```

> # with those in the GSMs
> 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)
> # go through the necessary steps to make a compliant ExpressionSet
> 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: 22283 features, 34 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: GSM11805 GSM11810 ... GSM12448 (34 total)
  varLabels: samples
  varMetadata: labelDescription
featureData: 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 Use Cases

GEOquery can be quite powerful for gathering a lot of data quickly. A few examples can be useful to show how this might be done for data mining purposes.

6.1 Getting all Series Records for a Given Platform

For data mining purposes, it is sometimes useful to be able to pull all the GSE records for a given platform. GEOquery makes this very easy, but a little bit of knowledge of the GPL record is necessary to get started. The GPL record contains both the GSE and GSM accessions that reference it. Some code is useful to illustrate the point:

```
> gpl97 <- getGEO('GPL97')
> Meta(gpl97)$title

[1] "[HG-U133B] Affymetrix Human Genome U133B Array"

> head(Meta(gpl97)$series_id)

[1] "GSE362" "GSE473" "GSE620" "GSE674" "GSE781" "GSE907"

> length(Meta(gpl97)$series_id)

[1] 149

> head(Meta(gpl97)$sample_id)

[1] "GSM3922" "GSM3924" "GSM3926" "GSM3928" "GSM3930" "GSM3932"

> length(Meta(gpl97)$sample_id)

[1] 6156
```

The code above loads the GPL97 record into R. The Meta method extracts a list of header information from the GPL record. The “title” gives the human name of the platform. The “series_id” gives a vector of series ids. Note that there are more than 120 series associated with this platform and more than 5100 samples. Code like the following could be used to download all the samples or series. I show only the first 5 samples as an example:

```
> gsmids <- Meta(gpl97)$sample_id
> gsmlist <- sapply(gsmids[1:5], getGEO)
> names(gsmlist)

[1] "GSM3922" "GSM3924" "GSM3926" "GSM3928" "GSM3930"
```

7 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.

8 sessionInfo

- R version 3.1.1 (2014-07-10), x86_64-unknown-linux-gnu
- Locale: LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C, LC_TIME=en_US.UTF-8, LC_COLLATE=C, LC_MONETARY=en_US.UTF-8, LC_MESSAGES=en_US.UTF-8, LC_PAPER=en_US.UTF-8, LC_NAME=C, LC_ADDRESS=C, LC_TELEPHONE=C, LC_MEASUREMENT=en_US.UTF-8, LC_IDENTIFICATION=C
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, utils
- Other packages: Biobase 2.24.0, BiocGenerics 0.10.0, GEOquery 2.30.1, limma 3.20.8
- Loaded via a namespace (and not attached): RCurl 1.95-4.1, XML 3.98-1.1, tools 3.1.1