Using the GEOquery package

Sean Davis‡*

October 22, 2008

‡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 .............................................. 2

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 ...................................... 8

4 Converting to BioConductor ExpressionSets and limma MALists 12
4.1 Getting GSE Series Matrix files as an ExpressionSet .................. 12
4.2 Converting GDS to an ExpressionSet .................. 13
4.3 Converting GDS to an MAList ........................ 16
4.4 Converting GSE to an ExpressionSet .................. 19

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.\textsuperscript{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

\textsuperscript{1}See \url{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("GDS10")

File stored at:
/tmp/RtmpnFTpz5/GDS10.soft

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/RtmpnFTpz5/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 GEO-query 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:

```r
> 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
```
Testis dissected from adult (12-24 hours post-eclosion) Drosophila melanogaster of the genotype y w[67c1]. Keywords = gonad, male, sex
$type
  [1] "RNA"

> Table(gsm)[1:5, ]

<table>
<thead>
<tr>
<th>ID_REF</th>
<th>SIGNAL_RAW</th>
<th>BKD_FORM</th>
<th>NORM_FORM</th>
<th>BKD_RAW</th>
<th>NORM_VALUE</th>
<th>CONST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>138392.65</td>
<td>no</td>
<td>no</td>
<td>101113.775</td>
<td>395070.1312</td>
<td>39542</td>
</tr>
<tr>
<td>2</td>
<td>100973.49</td>
<td>no</td>
<td>no</td>
<td>101113.775</td>
<td>395070.1312</td>
<td>39542</td>
</tr>
<tr>
<td>3</td>
<td>118994.03</td>
<td>no</td>
<td>no</td>
<td>101113.775</td>
<td>395070.1312</td>
<td>39542</td>
</tr>
<tr>
<td>4</td>
<td>108126.05</td>
<td>yes</td>
<td>no</td>
<td>101113.775</td>
<td>395070.1312</td>
<td>39542</td>
</tr>
<tr>
<td>5</td>
<td>293362.11</td>
<td>no</td>
<td>no</td>
<td>101113.775</td>
<td>395070.1312</td>
<td>39542</td>
</tr>
</tbody>
</table>

VALUE
1 76820.87249
2 39401.7125
3 57422.25249
4 46554.2725
5 231790.3324

> Columns(gsm)

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID_REF</td>
<td></td>
</tr>
<tr>
<td>SIGNAL_RAW</td>
<td>raw signal</td>
</tr>
<tr>
<td>BKD_FORM</td>
<td></td>
</tr>
<tr>
<td>NORM_FORM</td>
<td></td>
</tr>
<tr>
<td>BKD_RAW</td>
<td>raw background as taken in four quarters of microarray</td>
</tr>
<tr>
<td>NORM_VALUE</td>
<td>normalization value</td>
</tr>
<tr>
<td>CONST</td>
<td>constant value</td>
</tr>
<tr>
<td>VALUE</td>
<td></td>
</tr>
</tbody>
</table>

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

> Columns(gds)

<table>
<thead>
<tr>
<th>sample</th>
<th>tissue</th>
<th>strain</th>
<th>disease.state</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSM582</td>
<td>spleen</td>
<td>NOD</td>
<td>diabetic</td>
</tr>
<tr>
<td>GSM589</td>
<td>spleen</td>
<td>NOD</td>
<td>diabetic</td>
</tr>
<tr>
<td>GSM583</td>
<td>spleen</td>
<td>Idd3</td>
<td>diabetic-resistant</td>
</tr>
<tr>
<td>GSM590</td>
<td>spleen</td>
<td>Idd3</td>
<td>diabetic-resistant</td>
</tr>
<tr>
<td>GSM584</td>
<td>spleen</td>
<td>Idd5</td>
<td>diabetic-resistant</td>
</tr>
<tr>
<td>GSM591</td>
<td>spleen</td>
<td>Idd5</td>
<td>diabetic-resistant</td>
</tr>
<tr>
<td>Sample Code</td>
<td>Tissue</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>-------------------------------</td>
<td></td>
</tr>
<tr>
<td>GSM585</td>
<td>spleen</td>
<td>Idd3+Idd5 diabetic-resistant</td>
<td></td>
</tr>
<tr>
<td>GSM592</td>
<td>spleen</td>
<td>Idd3+Idd5 diabetic-resistant</td>
<td></td>
</tr>
<tr>
<td>GSM586</td>
<td>spleen</td>
<td>Idd9 diabetic-resistant</td>
<td></td>
</tr>
<tr>
<td>GSM593</td>
<td>spleen</td>
<td>Idd9 diabetic-resistant</td>
<td></td>
</tr>
<tr>
<td>GSM587</td>
<td>spleen</td>
<td>B10.H2g7 nondiabetic</td>
<td></td>
</tr>
<tr>
<td>GSM594</td>
<td>spleen</td>
<td>B10.H2g7 nondiabetic</td>
<td></td>
</tr>
<tr>
<td>GSM588</td>
<td>spleen</td>
<td>B10.H2g7 Idd3 nondiabetic</td>
<td></td>
</tr>
<tr>
<td>GSM595</td>
<td>spleen</td>
<td>B10.H2g7 Idd3 nondiabetic</td>
<td></td>
</tr>
<tr>
<td>GSM596</td>
<td>thymus</td>
<td>NOD diabetic</td>
<td></td>
</tr>
<tr>
<td>GSM603</td>
<td>thymus</td>
<td>NOD diabetic</td>
<td></td>
</tr>
<tr>
<td>GSM597</td>
<td>thymus</td>
<td>Idd3 diabetic-resistant</td>
<td></td>
</tr>
<tr>
<td>GSM604</td>
<td>thymus</td>
<td>Idd3 diabetic-resistant</td>
<td></td>
</tr>
<tr>
<td>GSM598</td>
<td>thymus</td>
<td>Idd5 diabetic-resistant</td>
<td></td>
</tr>
<tr>
<td>GSM605</td>
<td>thymus</td>
<td>Idd5 diabetic-resistant</td>
<td></td>
</tr>
<tr>
<td>GSM599</td>
<td>thymus</td>
<td>Idd3+Idd5 diabetic-resistant</td>
<td></td>
</tr>
<tr>
<td>GSM606</td>
<td>thymus</td>
<td>Idd3+Idd5 diabetic-resistant</td>
<td></td>
</tr>
<tr>
<td>GSM600</td>
<td>thymus</td>
<td>Idd9 diabetic-resistant</td>
<td></td>
</tr>
<tr>
<td>GSM607</td>
<td>thymus</td>
<td>Idd9 diabetic-resistant</td>
<td></td>
</tr>
<tr>
<td>GSM601</td>
<td>thymus</td>
<td>B10.H2g7 nondiabetic</td>
<td></td>
</tr>
<tr>
<td>GSM608</td>
<td>thymus</td>
<td>B10.H2g7 nondiabetic</td>
<td></td>
</tr>
<tr>
<td>GSM602</td>
<td>thymus</td>
<td>B10.H2g7 Idd3 nondiabetic</td>
<td></td>
</tr>
<tr>
<td>GSM609</td>
<td>thymus</td>
<td>B10.H2g7 Idd3 nondiabetic</td>
<td></td>
</tr>
</tbody>
</table>

Values:

1. Value for GSM582: NOD_S1; src: Spleen
2. Value for GSM589: NOD_S2; src: Spleen
3. Value for GSM583: Idd3_S1; src: Spleen
4. Value for GSM590: Idd3_S2; src: Spleen
5. Value for GSM584: Idd5_S1; src: Spleen
6. Value for GSM591: Idd5_S2; src: Spleen
7. Value for GSM585: Idd3+5_S1; src: Spleen
8. Value for GSM592: Idd3+5_S2; src: Spleen
9. Value for GSM586: Idd9_S1; src: Spleen
10. Value for GSM593: Idd9_S2; src: Spleen
11. Value for GSM587: B10.H2g7_S1; src: Spleen
12. Value for GSM594: B10.H2g7_S2; src: Spleen
13. Value for GSM588: B10.H2g7_Idd3_S1; src: Spleen
14. Value for GSM595: B10.H2g7_Idd3_S2; src: Spleen
15. Value for GSM596: NOD_T1; src: Thymus
16. Value for GSM603: NOD_T2; src: Thymus
17. Value for GSM597: Idd3_T1; src: Thymus
18. Value for GSM604: Idd3_T2; src: Thymus
19. Value for GSM598: Idd5_T1; src: Thymus
Value for GSM605: Idd5_T2; src: Thymus
Value for GSM599: Idd3+5_T1; src: Thymus
Value for GSM606: Idd3+5_T2; src: Thymus
Value for GSM600: Idd9_T1; src: Thymus
Value for GSM607: Idd9_T2; src: Thymus
Value for GSM601: B10.H2g7_T1; src: Thymus
Value for GSM608: B10.H2g7_T2; src: Thymus
Value for GSM602: B10.H2g7 Idd3_T1; src: Thymus
Value for GSM609: B10.H2g7 Idd3_T2; src: Thymus

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("GSE462", GSEMatrix = FALSE)

File stored at: /tmp/RtmpnFTpz5/GSE462.soft
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

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

$contributor
[4] "Jining, Läij" "Kevin, G, Becker" "Brian, Oliver"

$geo_accession
[1] "GSE462"

$last_update_date
[1] "Oct 28 2005"

$platform_id
[1] "GPL5"

$pubmed_id
[1] "11116097"
Identification and annotation of all the genes in the sequenced Drosophila genome is a work in progress. Wild-type flies were collected in a stock colony and frozen on the day of collection. Total RNA was extracted from frozen flies using the TRIzol reagent. mRNA was isolated using the mRNA-quick kit. These data suggest that the number of genes in Drosophila will significantly exceed the conservative estimate of 13,601.

Analysis of transcription in the Drosophila melanogaster testis

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_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 genotype y w[67c1]."
geo_accession [1] "GSM10"
last_update_date [1] "Mar 09 2006"
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] "RNA"

An object of class "GEODataTable"

***** Column Descriptions *****

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID_REF</td>
<td></td>
</tr>
<tr>
<td>SIGNAL_RAW</td>
<td>raw signal</td>
</tr>
<tr>
<td>BKD_FORM</td>
<td></td>
</tr>
<tr>
<td>NORM_FORM</td>
<td></td>
</tr>
<tr>
<td>BKD_RAW</td>
<td>raw background</td>
</tr>
</tbody>
</table>
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.

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

Found 1 file(s)
GSE2553_series_matrix.txt.gz
File stored at:
/tmp/RtmpnFTpz5/GPL1977.soft

> show(gse2553)
```

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

<table>
<thead>
<tr>
<th>title</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSM48681</td>
<td>Patient sample ST18, Dermatofibrosarcoma</td>
</tr>
<tr>
<td>GSM48682</td>
<td>Patient sample ST410, Ewing Sarcoma</td>
</tr>
<tr>
<td>GSM48683</td>
<td>Patient sample ST130, Sarcoma, NOS</td>
</tr>
<tr>
<td>GSM48684</td>
<td>Patient sample ST293, Malignant Peripheral Nerve Sheath Tumor</td>
</tr>
<tr>
<td>GSM48685</td>
<td>Patient sample ST367, Liposarcoma</td>
</tr>
</tbody>
</table>

4.2 Converting GDS to an ExpressionSet

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

> eset <- GDS2eSet(gds, do.log2 = TRUE)
Now, \texttt{eset} is an \textit{ExpressionSet} that contains the same information as in the GEO dataset, including the sample information, which we can see here:

\begin{verbatim}
> eset

ExpressionSet (storageMode: lockedEnvironment)
assayData: 39114 features, 28 samples
  element names: exprs
phenoData
  sampleNames: GSM582, GSM589, ..., GSM609 (28 total)
  varLabels and varMetadata description:
    sample: NA
tissue: NA
...: ...
description: NA
(5 total)
featureData
  featureNames: 1, 2, ..., 39114 (39114 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 \texttt{'experimentData(object)'
  pubMedIds: 11827943
Annotation:

> pData(eset)

  sample  tissue  strain  disease.state
GSM582   GSM582 spleen NOD diabetic
GSM589   GSM589 spleen NOD diabetic
GSM583   GSM583 spleen Idd3 diabetic-resistant
GSM590   GSM590 spleen Idd3 diabetic-resistant
GSM584   GSM584 spleen Idd5 diabetic-resistant
GSM591   GSM591 spleen Idd5 diabetic-resistant
GSM585   GSM585 spleen Idd3+Idd5 diabetic-resistant
GSM592   GSM592 spleen Idd3+Idd5 diabetic-resistant
GSM586   GSM586 spleen Idd9 diabetic-resistant
\end{verbatim}
GSM593 GSM593 spleen Idd9 diabetic-resistant
GSM587 GSM587 spleen B10.H2g7 nondiabetic
GSM594 GSM594 spleen B10.H2g7 nondiabetic
GSM588 GSM588 spleen B10.H2g7 Idd3 nondiabetic
GSM595 GSM595 spleen B10.H2g7 Idd3 nondiabetic
GSM596 GSM596 thymus NOD diabetic
GSM603 GSM603 thymus NOD diabetic
GSM597 GSM597 thymus Idd3 diabetic-resistant
GSM604 GSM604 thymus Idd3 diabetic-resistant
GSM598 GSM598 thymus Idd5 diabetic-resistant
GSM605 GSM605 thymus Idd5 diabetic-resistant
GSM599 GSM599 thymus Idd3+Idd5 diabetic-resistant
GSM606 GSM606 thymus Idd3+Idd5 diabetic-resistant
GSM600 GSM600 thymus Idd9 diabetic-resistant
GSM607 GSM607 thymus Idd9 diabetic-resistant
GSM601 GSM601 thymus B10.H2g7 nondiabetic
GSM608 GSM608 thymus B10.H2g7 nondiabetic
GSM602 GSM602 thymus B10.H2g7 Idd3 nondiabetic
GSM609 GSM609 thymus B10.H2g7 Idd3 nondiabetic

Value for GSM582: NOD_S1; src: Spleen
Value for GSM589: NOD_S2; src: Spleen
Value for GSM583: Idd3_S1; src: Spleen
Value for GSM590: Idd3_S2; src: Spleen
Value for GSM584: Idd5_S1; src: Spleen
Value for GSM591: Idd5_S2; src: Spleen
Value for GSM585: Idd3+5_S1; src: Spleen
Value for GSM592: Idd3+5_S2; src: Spleen
Value for GSM586: Idd9_S1; src: Spleen
Value for GSM593: Idd9_S2; src: Spleen
Value for GSM587: B10.H2g7_S1; src: Spleen
Value for GSM594: B10.H2g7_S2; src: Spleen
Value for GSM588: B10.H2g7_Idd3_S1; src: Spleen
Value for GSM595: B10.H2g7_Idd3_S2; src: Spleen
Value for GSM596: NOD_T1; src: Thymus
Value for GSM603: NOD_T2; src: Thymus
Value for GSM597: Idd3_T1; src: Thymus
Value for GSM604: Idd3_T2; src: Thymus
Value for GSM598: Idd5_T1; src: Thymus
Value for GSM605: Idd5_T2; src: Thymus
Value for GSM599: Idd3+5_T1; src: Thymus
Value for GSM606: Idd3+5_T2; src: Thymus
GSM600 Value for GSM600: Idd9_T1; src: Thymus
GSM607 Value for GSM607: Idd9_T2; src: Thymus
GSM601 Value for GSM601: B10.H2g7_T1; src: Thymus
GSM608 Value for GSM608: B10.H2g7_T2; src: Thymus
GSM602 Value for GSM602: B10.H2g7 Idd3_T1; src: Thymus
GSM609 Value for GSM609: B10.H2g7 Idd3_T2; src: Thymus

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.

```r
> Meta(gds)$platform
[1] "GPL24"
```

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

File stored at:
/tmp/RtmpnFTpz5/GPL5.soft

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:

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

An object of class "MAList"

```
$M
GSM582 GSM589 GSM583 GSM590 GSM584 GSM591 GSM585 GSM592 GSM586 GSM593
[1,] 101 54 111 55 87 30 99 43 105 56
[2,] 26 23 30 27 19 22 32 19 24 25
[3,] NA NA NA NA NA NA NA NA NA NA
[4,] 233 162 252 178 214 144 238 147 250 166
[5,] NA NA NA NA NA NA NA NA NA NA
```

GSM587 GSM594 GSM588 GSM595 GSM596 GSM603 GSM597 GSM604 GSM598 GSM605
[1,] 43 14 112 43 97 36 117 40 125 45
[2,] 14 49 32 29 31 22 26 26 35 26
[3,] NA 7 NA 4 10 22 NA 15 NA 23
[4,] 86 22 236 139 216 112 241 130 270 144
[5,] NA NA NA 3 NA NA NA NA NA NA
```
### $A$

```
NULL
```

### $\text{targets}$

<table>
<thead>
<tr>
<th>sample</th>
<th>tissue</th>
<th>strain</th>
<th>disease.state</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GSM582</td>
<td>spleen</td>
<td>NOD diabetic</td>
</tr>
<tr>
<td>2</td>
<td>GSM589</td>
<td>spleen</td>
<td>NOD diabetic</td>
</tr>
<tr>
<td>3</td>
<td>GSM583</td>
<td>spleen</td>
<td>Idd3 diabetic-resistant</td>
</tr>
<tr>
<td>4</td>
<td>GSM590</td>
<td>spleen</td>
<td>Idd3 diabetic-resistant</td>
</tr>
<tr>
<td>5</td>
<td>GSM584</td>
<td>spleen</td>
<td>Idd5 diabetic-resistant</td>
</tr>
</tbody>
</table>

### description

1 Value for GSM582: NOD_S1; src: Spleen
2 Value for GSM589: NOD_S2; src: Spleen
3 Value for GSM583: Idd3_S1; src: Spleen
4 Value for GSM590: Idd3_S2; src: Spleen
5 Value for GSM584: Idd5_S1; src: Spleen

23 more rows ...

### $\text{genes}$

<table>
<thead>
<tr>
<th>ID</th>
<th>GB_ACC</th>
<th>BSCC_ID</th>
<th>CLONE_ID</th>
<th>SUB.ARRAY</th>
<th>DUPLICATE</th>
<th>ROW</th>
<th>COLUMN</th>
<th>PCR_QC</th>
<th>SPOT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AI944549</td>
<td>bs03g07</td>
<td>FBgn0033989</td>
<td>1</td>
<td>a</td>
<td>1</td>
<td>1</td>
<td>passed</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>AI944695</td>
<td>bs04c11</td>
<td>FBgn0032821</td>
<td>1</td>
<td>a</td>
<td>1</td>
<td>2</td>
<td>passed</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>AI944741</td>
<td>bs04h01</td>
<td>FBgn0034374</td>
<td>1</td>
<td>a</td>
<td>1</td>
<td>3</td>
<td>passed</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>AI944801</td>
<td>bs05f04</td>
<td>FBgn0039421</td>
<td>1</td>
<td>a</td>
<td>1</td>
<td>4</td>
<td>failed</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>AI945043</td>
<td>bs08c11</td>
<td>FBgn0045370</td>
<td>1</td>
<td>a</td>
<td>1</td>
<td>5</td>
<td>passed</td>
<td></td>
</tr>
</tbody>
</table>

1
g 2
3
4 gi|4505995|ref|NP_002697.1|PPPM1B| protein phosphatase 1B (formerly 2C), magnesium-dependent >gi|3378168|emb|CAA06704| (AJ005801) PP2C [Homosapiens]
5

E_VAL SPOT_QC
1 2e-08 44364
2 <NA> 16957

39109 more rows ...
3 <NA>  17896
4 1e-25  16363
5 <NA>  83502
39109 more rows ...

$notes
[[1]]
[1] "able_begin"

$channel_count
[1] "1"

$description
[1] "Examination of spleen and thymus of type 1 diabetes nonobese diabetic (NOD) mouse, four NOD-derived diabetes-resistant congenic strains and two nondiabetic control strains."

$feature_count
[1] "39114"

$order
[1] "none"

$platform
[1] "GPL24"

$platform_organism
[1] "Mus musculus"

$platform_technology_type
[1] "in situ oligonucleotide"

$pubmed_id
[1] "11827943"

$reference_series
[1] "GSE11"

$sample_count
[1] "28"

$sample_organism
[1] "Mus musculus"
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:

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

$GSM10
[1] "GPL5"

$GSM3
[1] "GPL5"

$GSM4
[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).

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

<table>
<thead>
<tr>
<th>ID_REF</th>
<th>SIGNAL_RAW</th>
<th>BKD_FORM</th>
<th>NORM_FORM</th>
<th>BKD_RAW</th>
<th>NORM_VALUE</th>
<th>CONST</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4486.49</td>
<td>0</td>
<td>0</td>
<td>3379.579</td>
<td>23337.54</td>
<td>39542</td>
<td>55845.45</td>
</tr>
<tr>
<td>2</td>
<td>3482.51</td>
<td>0</td>
<td>0</td>
<td>3379.579</td>
<td>23337.54</td>
<td>39542</td>
<td>41058.05</td>
</tr>
<tr>
<td>3</td>
<td>3812.39</td>
<td>0</td>
<td>0</td>
<td>3379.579</td>
<td>23337.54</td>
<td>39542</td>
<td>45916.78</td>
</tr>
<tr>
<td>4</td>
<td>3257.56</td>
<td>1</td>
<td>0</td>
<td>3379.579</td>
<td>23337.54</td>
<td>39542</td>
<td>37744.81</td>
</tr>
<tr>
<td>5</td>
<td>5436.91</td>
<td>0</td>
<td>0</td>
<td>3379.579</td>
<td>23337.54</td>
<td>39542</td>
<td>69843.97</td>
</tr>
</tbody>
</table>
```

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

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ID_REF</td>
</tr>
<tr>
<td>2</td>
<td>SIGNAL_RAW</td>
</tr>
<tr>
<td>3</td>
<td>BKD_FORM</td>
</tr>
<tr>
<td>4</td>
<td>NORM_FORM</td>
</tr>
<tr>
<td>5</td>
<td>BKD_RAW</td>
</tr>
</tbody>
</table>
```

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

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:

```r
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: 3455 features, 8 samples
  element names: exprs
phenoData
  sampleNames: GSM10, GSM3, ..., GSM9 (8 total)
  varLabels and varMetadata description:
    samples: NA
featureData
  featureNames: 1, 2, ..., 3455 (3455 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. Future work with this package will likely focus on better tools for manipulating `GSE` data.

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 2.8.0 (2008-10-20), x86_64-unknown-linux-gnu
- Locale: LC_CTYPE=en_US;LC_NUMERIC=C;LC_TIME=en_US;LC_COLLATE=en_US;LC_MONETARY=C;LC_MESSAGES=en_US;LC_PAPER=en_US;LC_NAME=C;LC_ADDRESS=C;LC_TELEPHONE=C;LC_MEASUREMENT=en_US;LC_IDENTIFICATION=C
- Base packages: base, datasets, graphics, grDevices, methods, stats, tools, utils
- Other packages: Biobase 2.2.0, GEOquery 2.6.0, limma 2.16.0, RCurl 0.91-0