Using the GEOquery package

Sean Davis‡*

August 9, 2007

‡Cancer Genetics Branch
National Human Genome Research Institute
National Institutes of Health

Contents

1 Overview of GEO 1
   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 ................................. 3
   3.2 The GSE class ...................................................... 6

4 Converting to BioConductor exprSets and limma MALists 11
   4.1 Converting GDS to an exprSet .................................. 11
   4.2 Converting GDS to an MAList .................................. 12
   4.3 Converting GSE to an exprSet .................................. 15

5 Conclusion 18

1 Overview of GEO

The NCBI Gene Expression Omnibus (GEO) serves as a public repository for a wide range of high-throughput experimental data. These data include single and dual channel microarray-based experiments measuring mRNA, genomic DNA, and protein abundance, as well as

*sdavis2@mail.nih.gov
non-array techniques such as serial analysis of gene expression (SAGE), and mass spectrometry proteomic data. Currently, 65,000 samples and nearly 2000 different platforms are represented in GEO!

At the most basic level of organization of GEO, there are four basic entity types. The first three (Sample, Platform, and Series) are supplied by users; the fourth, the dataset, is compiled and curated by GEO staff from the user-submitted data.

1. Platforms

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

1.2 Samples

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

1.3 Series

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

1.4 Datasets

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

See [http://www.ncbi.nih.gov/geo](http://www.ncbi.nih.gov/geo) for more information
2 Getting Started using GEOquery

Getting data from GEO is really quite easy. There is only one command that is needed, getGEO. This one function interprets its input to determine how to get the data from GEO and then parse the data into useful R data structures. Usage is quite simple:

```r
> library(GEOquery)

This loads the GEOquery library.

> gds <- getGEO("GDS1")

File stored at:
/tmp/RtmpNNp0Kv/GDS1.soft.gz
parsing geodata
parsing subsets
ready to return
```

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

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

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

File stored at:
/tmp/RtmpNNp0Kv/GSM3.soft
```

3 GEOquery Data Structures

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

3.1 The GDS, GSM, and GPL classes

Each of these classes is comprised of a metadata header (taken nearly verbatim from the SOFT format header) and a 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

`contact_zip/postal_code`
[1] "20892"

$data_row_count
[1] "3456"
Testis dissected from adult (12-24 hours post-eclosion) Drosophila melanogaster of the genotype y w[67c1].

Keywords = gonad, male, sex

geo_accession
[1] "GSM3"

last_update_date
[1] "May 27 2005"

molecule_ch1
[1] "total RNA"

organism_ch1
[1] "Drosophila melanogaster"

platform_id
[1] "GPL5"

series_id
[1] "GSE462"

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

status
[1] "Public on Oct 18 2000"

submission_date
[1] "Oct 18 2000"

supplementary_file
[1] "NONE"

title
[1] "testis a"

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>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>138392.6</td>
<td>no</td>
<td>no</td>
<td>101113.8</td>
<td>395070.1</td>
<td>39542</td>
</tr>
</tbody>
</table>
2 2 100973.5 no no 101113.8 395070.1 39542 39401.71
3 3 118994.0 no no 101113.8 395070.1 39542 57422.25
4 4 108126.1 yes no 101113.8 395070.1 39542 46554.27
5 5 293362.1 no no 101113.8 395070.1 39542 231790.33

> Columns(gsm)

<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>gender</th>
<th>tissue</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSM3</td>
<td>male</td>
<td>testis</td>
</tr>
<tr>
<td>GSM4</td>
<td>male</td>
<td>testis</td>
</tr>
<tr>
<td>GSM5</td>
<td>male</td>
<td>gonadectomized male</td>
</tr>
<tr>
<td>GSM6</td>
<td>male</td>
<td>gonadectomized male</td>
</tr>
<tr>
<td>GSM7</td>
<td>female</td>
<td>ovary</td>
</tr>
<tr>
<td>GSM8</td>
<td>female</td>
<td>ovary</td>
</tr>
<tr>
<td>GSM9</td>
<td>female</td>
<td>gonadectomized female</td>
</tr>
<tr>
<td>GSM10</td>
<td>female</td>
<td>gonadectomized female</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>description</th>
<th>Value for GSM3: testis a; src: y w[67c1]/Y testis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value for GSM4: testis b; src: y w[67c1]/Y testis</td>
</tr>
<tr>
<td></td>
<td>Value for GSM5: male a; src: y w[67c1]/Y male</td>
</tr>
<tr>
<td></td>
<td>Value for GSM6: male b; src: y w[67c1]/Y</td>
</tr>
<tr>
<td></td>
<td>Value for GSM7: ovary a; src: y w[67c1] ovary</td>
</tr>
<tr>
<td></td>
<td>Value for GSM8: ovary b; src: y w[67c1] ovary</td>
</tr>
<tr>
<td></td>
<td>Value for GSM9: female a; src: y w[67c1] female</td>
</tr>
<tr>
<td></td>
<td>Value for GSM10: female b; src: y w[67c1] female</td>
</tr>
</tbody>
</table>

### 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:

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

File stored at:
/tmp/RtmpNNp0Kv/GSE462.soft.gz
Parsing....
^PLATFORM = GPL5
^SAMPLE = GSM3
^SAMPLE = GSM4
^SAMPLE = GSM5
^SAMPLE = GSM6
^SAMPLE = GSM7
^SAMPLE = GSM8
^SAMPLE = GSM9

> Meta(gse)

$contact_address
[1] "6 Center Drive"

$contact_city
[1] "Bethesda"

$contact_country
[1] "USA"

$contact_department
[1] "LCDB"

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

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

$contact_institute
[1] "NIDDK, NIH"

$contact_name
[1] "Brian, Oliver"
```
Identification and annotation of all the genes in the sequenced Drosophila genome is a work in progress. Wild-type ... these data suggest that the number of genes in Drosophila will significantly exceed the conservative estimate of 13,601.
$type
[1] "other"

> names(GSMList(gse))

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

> GSMList(gse)[[1]]

An object of class "GSM"
channel_count
[1] "1"
contact_address
[1] "6 Center Drive"
contact_city
[1] "Bethesda"
contact_country
[1] "USA"
contact_department
[1] "LCDB"
contact_email
[1] "oliver@helix.nih.gov"
contact_fax
[1] "301-496-5239"
contact_institute
[1] "NIDDK, NIH"
contact_name
[1] "Brian, Oliver"
contact_phone
[1] "301-496-5495"
contact_state
[1] "MD"
contact_web_link
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]."
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>
<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>

***** Data Table *****

<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>

3451 more rows ...

> names(GPLList(gse))

[1] "GPL5"
4 Converting to BioConductor exprSets and limma MALists

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

4.1 Converting GDS to an exprSet

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

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

File stored at:
/tmp/RtmpNNp0Kv/GPL5.soft

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

```r
> eset
```

ExpressionSet (storageMode: lockedEnvironment)
assayData: 3456 features, 8 samples
   element names: exprs
phenoData
   rowNames: GSM3, GSM4, ..., GSM10 (8 total)
   varLabels and varMetadata:
     sample: NA
     gender: NA
     tissue: NA
     description: NA
featureData
   rowNames: 1, 2, ..., 3456 (3456 total)
   varLabels and varMetadata:
     ID:
       ...: ...
     SPOT_QC: spot quality control
(13 total)
   varMetadata: Column, labelDescription
experimentData: use 'experimentData(object)'
   pubMedIds: 11116097
Annotation character(0)
> pData(eset)

<table>
<thead>
<tr>
<th>sample</th>
<th>gender</th>
<th>tissue</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSM3</td>
<td>GSM3</td>
<td>male testis</td>
</tr>
<tr>
<td>GSM4</td>
<td>GSM4</td>
<td>male testis</td>
</tr>
<tr>
<td>GSM5</td>
<td>GSM5</td>
<td>male gonadectomized male</td>
</tr>
<tr>
<td>GSM6</td>
<td>GSM6</td>
<td>male gonadectomized male</td>
</tr>
<tr>
<td>GSM7</td>
<td>GSM7</td>
<td>female ovary</td>
</tr>
<tr>
<td>GSM8</td>
<td>GSM8</td>
<td>female ovary</td>
</tr>
<tr>
<td>GSM9</td>
<td>GSM9</td>
<td>female gonadectomized female</td>
</tr>
<tr>
<td>GSM10</td>
<td>GSM10</td>
<td>female gonadectomized female</td>
</tr>
</tbody>
</table>

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

4.2 Converting GDS to an MAList

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

> Meta(gds)$platform

[1] "GPL5"

> gpl <- getGEO("GPL5")

File stored at:
/tmp/RtmpNNp0Kv/GPL5.soft

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

> MA <- GDS2MA(gds, GPL = gpl)
> MA
An object of class "MAList"

$M

GSM3  GSM4  GSM5  GSM6  GSM7  GSM8  GSM9  GSM10
[1,] 76820.87 71715.76 51430.49 139715.76 45027.85 69984.33 38569.01 55845.45
[2,]  39401.71  NA  37746.64  91150.39  29691.45  36329.52  30363.85  41058.05
[3,]  57422.25 18338.46  37134.59   75928.14  34181.67  42713.88  32090.47  45916.78
[4,]  46554.27 10928.63  34145.17   74550.27  28498.81  33207.63  37744.81
[5,] 231790.33 341779.05  77703.83   99999.62  61151.98  65974.36  60665.36  69843.97
3451 more rows ... 

$A
NULL

$targets

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

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

$genes

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

1
gi|4505995|ref|NP_002697.1|PPPM1B| protein phosphatase 1B (formerly 2C), magnesium-dependent, beta isoform

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

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

$channel_count
[1] "1"

$description
[1] "Adult testis gene expression profile and gene discovery. Examines testis, whole male minus gonads, ovary and whole female minus gonads from adult, 12-24 hours post-eclosion, genotype y w[67c1]."

$feature_count
[1] "3456"

$order
[1] "none"

$platform
[1] "GPL5"

$platform_organism
[1] "Drosophila melanogaster"

$platform_technology_type
[1] "spotted DNA/cDNA"

$pubmed_id
[1] "11116097"

$reference_series
[1] "GSE462"
4.3 Converting GSE to an exprSet

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

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

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

$GSM10
[1] "GPL5"

$GSM3
[1] "GPL5"
```
Indeed, they all used GPL5 as their platform (which we could have determined by looking at the GPLList for gse, which shows only one GPL for this particular GSE.). So, now we would like to know what column represents the data that we would like to extract. Looking at the first few rows of the Table of a single GSM will likely give us an idea (and by the way, GEO uses a convention that the column that contains the single “measurement” for each array is called the “VALUE” column, which we could use if we don’t know what other column is most relevant).

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

<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>1</td>
<td>4486.49</td>
<td>0</td>
<td>0</td>
<td>3379.579</td>
<td>23337.54</td>
<td>39542</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3482.51</td>
<td>0</td>
<td>0</td>
<td>3379.579</td>
<td>23337.54</td>
<td>39542</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3812.39</td>
<td>0</td>
<td>0</td>
<td>3379.579</td>
<td>23337.54</td>
<td>39542</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3257.56</td>
<td>1</td>
<td>0</td>
<td>3379.579</td>
<td>23337.54</td>
<td>39542</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5436.91</td>
<td>0</td>
<td>0</td>
<td>3379.579</td>
<td>23337.54</td>
<td>39542</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ID_REF</td>
<td></td>
</tr>
<tr>
<td>2 SIGNAL_RAW</td>
<td>raw signal</td>
</tr>
<tr>
<td>3 BKD_FORM</td>
<td></td>
</tr>
<tr>
<td>4 NORM_FORM</td>
<td></td>
</tr>
<tr>
<td>5 BKD_RAW</td>
<td>raw background</td>
</tr>
</tbody>
</table>
We will indeed use the “VALUE” column. We then want to make a matrix of these values like so:

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

GSM10  GSM3  GSM4  GSM5  GSM6  GSM7  GSM8  GSM9
```

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

```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 <- new("phenoData", pData = pdata, varLabels = as.list("samples"))
> eset2 <- new("exprSet", exprs = data.matrix, phenoData = pheno)
> eset2

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

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

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