Using the *GEOmetadb* package

Jack Zhu*and Sean Davis†
Genetics Branch, National Cancer Institute, National Institutes of Health

October 28, 2009

Contents

1 Overview of *GEOmetadb*

The NCBI Gene Expression Omnibus (GEO) represents the largest repository of microarray data in existence. One difficulty in dealing with GEO is finding the microarray data that is of interest. As part of the NCBI Entrez search system, GEO can be searched online via web pages or using NCBI Eutils. However, the web search is not as full-featured as it could be, particularly for programmatic access. NCBI Eutils offers another option for finding data within the vast stores of GEO, but it is cumbersome to use, often requiring multiple complicated Eutils calls to get at the relevant information. We have found it **absolutely**

*zhujack@mail.nih.gov
†sdavis2@mail.nih.gov

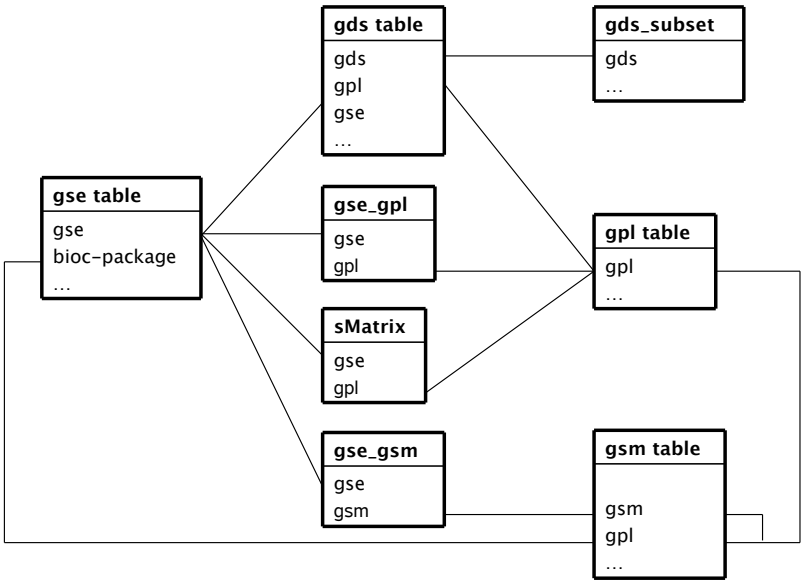

Figure 1: A graphical representation (sometimes called an *Entity-Relationship Diagram*) of the relationships between the tables in the GEOmetadb SQLite database

critical to have ready access not just to the microarray data, but to the metadata describing the microarray experiments. To this end we have created *GEOmetadb*.

1.1 What is *GEOmetadb*?

The *GEOmetadb* is an attempt to make querying the metadata describing microarray experiments, platforms, and datasets both easier and more powerful. At the heart of *GEOmetadb* is a SQLite database that stores nearly all the metadata associated with all GEO data types including GEO samples (GSM), GEO platforms (GPL), GEO data series (GSE), and curated GEO datasets (GDS), as well as the relationships between these data types. This database is generated by our server by parsing all the records in GEO and needs to be downloaded via a simple helper function to the user's local machine before *GEOmetadb* is useful. Once this is done, the entire GEO database is accessible with simple SQL-based queries. With the GEOmetadb database, queries that are simply not possible using NCBI tools or web pages are often quite simple. The relationships between the tables in the GEOmetadb SQLite database can be seen in figure 1.

1.2 Conversion capabilities

A very typical problem for large-scale consumers of GEO data is to determine the relationships between various GEO accession types. As examples, consider the following questions:

- What samples are associated with GEO platform "GPL96", which represents the Affymetrix hgu133a array?
- What GEO Series were performed using "GPL96"?
- What samples are in my favorite three GEO Series records?
- How many samples are associated with the ten most popular GEO platforms?

Because these types of questions are common, *GEOmetadb* contains the function `geoConvert` that addresses these questions directly and efficiently.

1.3 What *GEOmetadb* is not

We have faithfully parsed and maintained in GEO when creating *GEOmetadb*. This means that limitations inherent to GEO are also inherent in *GEOmetadb*. We have made no attempt to curate, semantically recode, or otherwise "clean up" GEO; to do so would require significant resources, which we do not have.

GEOmetadb does not contain any microarray data. For access to microarray data from within R/Bioconductor, please look at the *GEOquery* package. In fact, we would expect that many users will find that the combination of *GEOmetadb* and *GEOquery* is quite powerful.

2 Getting Started

Once *GEOmetadb* is installed (see the Bioconductor website for full installation instructions), we are ready to begin.

2.1 Getting the *GEOmetadb* database

This package does not come with a pre-installed version of the database. This has the advantage that the user will get the most up-to-date version of the database to start; the database can be re-downloaded using the same command as often as desired. First, load the library.

```
> library(GEOmetadb)
```

The download and uncompress steps are done automatically with a single command, `getSQLiteFile`.

```
> getSQLiteFile()
```

```
Unzipping...
Metadata associate with downloaded file:
                name               value
1     schema version                 1.0
2 creation timestamp 2009-10-24 07:56:26
[1] "/Users/biocbuild/bbs-2.5-bioc/meat/GEOmetadb/inst/doc/GEOmetadb.sqlite"
```

The default storage location is in the current working directory and the default filename is "GEOmetadb.sqlite"; it is best to leave the name unchanged unless there is a pressing reason to change it.

Since this SQLite file is of key importance in *GEOmetadb*, it is perhaps of some interest to know some details about the file itself.

```
> file.info("GEOmetadb.sqlite")
```

```
                      size isdir mode
GEOmetadb.sqlite 987377664 FALSE  644
                                mtime
GEOmetadb.sqlite 2009-10-28 03:23:06
                                ctime
GEOmetadb.sqlite 2009-10-28 03:23:06
                                atime uid gid     uname
GEOmetadb.sqlite 2009-10-28 03:23:06 502  20 biocbuild
                 grname
GEOmetadb.sqlite  staff
```

Now, the SQLite file is available for connection. The standard *DBI* functionality as implemented in *RSQLite* function `dbConnect` makes the connection to the database. The `dbDisconnect` function disconnects the connection.

```
> con <- dbConnect(SQLite(), "GEOmetadb.sqlite")
> dbDisconnect(con)
```

```
[1] TRUE
```

The variable `con` is an *RSQLite* connection object.

2.2 A word about SQL

The Structured Query Language, or SQL, is a very powerful and standard way of working with relational data. GEO is composed of several data types, all of which are related to each other; in fact, NCBI uses a relational SQL database for metadata storage and querying. SQL databases and SQL itself are designed specifically to work efficiently with just such data. While the goal of many programming projects and programmers is to hide the details of SQL

from the user, we are of the opinion that such efforts may be counterproductive, particularly with complex data and the need for *ad hoc* queries, both of which are characteristics with GEO metadata. We have taken the view that exposing the power of SQL will enable users to maximally utilize the vast data repository that is GEO. We understand that many users are not accustomed to working with SQL and, therefore, have devoted a large section of the vignette to working examples. Our goal is not to teach SQL, so a quick tutorial of SQL is likely to be beneficial to those who have not used it before. Many such tutorials are available online and can be completed in 30 minutes or less.

3 Examples

3.1 Interacting with the database

The functionality covered in this section is covered in much more detail in the *DBI* and *RSQLite* package documentation. We cover enough here only to be useful.

Again, we connect to the database.

```
> con <- dbConnect(SQLite(), "GEOmetadb.sqlite")
```

The `dbListTables` function lists all the tables in the SQLite database handled by the connection object `con`.

```
> geo_tables <- dbListTables(con)
> geo_tables
```

```
 [1] "gds"               "gds_subset"
 [3] "geoConvert"        "geodb_column_desc"
 [5] "gpl"               "gse"
 [7] "gse_gpl"           "gse_gsm"
 [9] "gsm"               "metaInfo"
[11] "sMatrix"
```

There is also the `dbListFields` function that can list database fields associated with a table.

```
> dbListFields(con, "gse")
```

```
 [1] "ID"                    "title"
 [3] "gse"                   "status"
 [5] "submission_date"       "last_update_date"
 [7] "pubmed_id"             "summary"
 [9] "type"                  "contributor"
[11] "web_link"              "overall_design"
[13] "repeats"               "repeats_sample_list"
[15] "variable"              "variable_description"
[17] "contact"               "supplementary_file"
```

Sometimes it is useful to get the actual SQL schema associated with a table. As an example of doing this and using an *RSQLite* shortcut function, `sqliteQuickSQL`, we can get the table schema for the *gpl* table.

```
> sqliteQuickSQL(con, "PRAGMA TABLE_INFO(gpl)")
```

```
   cid                 name type notnull dflt_value pk
1    0                   ID REAL       0       <NA>  0
2    1                title TEXT       0       <NA>  0
3    2                  gpl TEXT       0       <NA>  0
4    3               status TEXT       0       <NA>  0
5    4      submission_date TEXT       0       <NA>  0
6    5     last_update_date TEXT       0       <NA>  0
7    6           technology TEXT       0       <NA>  0
8    7         distribution TEXT       0       <NA>  0
9    8             organism TEXT       0       <NA>  0
10   9         manufacturer TEXT       0       <NA>  0
11  10 manufacture_protocol TEXT       0       <NA>  0
12  11              coating TEXT       0       <NA>  0
13  12       catalog_number TEXT       0       <NA>  0
14  13              support TEXT       0       <NA>  0
15  14          description TEXT       0       <NA>  0
16  15             web_link TEXT       0       <NA>  0
17  16              contact TEXT       0       <NA>  0
18  17       data_row_count REAL       0       <NA>  0
19  18   supplementary_file TEXT       0       <NA>  0
20  19         bioc_package TEXT       0       <NA>  0
```

3.2 Writing SQL queries and getting results

Select 5 records from the *gse* table and show the first 7 columns.

```
> rs <- dbGetQuery(con, "select * from gse limit 5")
> rs[, 1:7]
```

```
  ID
1  1
2  2
3  3
4  4
5  5
                                                 title
1                                 NHGRI_Melanoma_class
2                               Cerebellar development
```

```
3             Renal Cell Carcinoma Differential Expression
4     Diurnal and Circadian-Regulated Genes in Arabidopsis
5 Global profile of germline gene expression in C. elegans
   gse                 status submission_date
1 GSE1 Public on Jan 22 2001      2001-01-22
2 GSE2 Public on Apr 26 2001      2001-04-19
3 GSE3 Public on Jul 19 2001      2001-07-19
4 GSE4 Public on Jul 20 2001      2001-07-20
5 GSE5 Public on Jul 24 2001      2001-07-24
  last_update_date pubmed_id
1       2005-05-29  10952317
2       2005-05-29        NA
3       2005-05-29  11691851
4       2005-05-29  11158533
5       2005-07-18  11030340
```

Get the GEO series accession and title from GEO series that were submitted by "Sean Davis". The "

```
> rs <- dbGetQuery(con, paste("select gse,title from gse where",
+     "contributor like '%Sean%Davis%'", sep = " "))
> rs
```

```
        gse
1   GSE2553
2   GSE4406
3   GSE5357
4   GSE7376
5   GSE8486
6   GSE9328
7   GSE7882
8  GSE14543
9   GSE5481
10 GSE18544
```

```
1
2                                        Gene expression profiling of CD4+ T-cells and GM
3
4                                                        Detection of novel amplific
5                                                        Whole genome DNAse hypersen
6
7                       Gene Expression and Comparative Genomic Hybridization of Ductal C
8                                                                    A molecular f
```

```
9
10 Expression Profiling of a Mouse Xenograft Model of âĂIJTriple-NegativeâĂ■ Breast Canc
```

As another example, GEOmetadb can find all samples on GPL96 (Affymetrix hgu133a) that have .CEL files available for download.

```
> rs <- dbGetQuery(con, paste("select gsm,supplementary_file",
+     "from gsm where gpl='GPL96'", "and supplementary_file like '%CEL.gz'"))
> dim(rs)
```

```
[1] 13892     2
```

But why limit to only GPL96? Why not look for all Affymetrix arrays that have .CEL files? And list those with their associated GPL information, as well as the Bioconductor annotation package name?

```
> rs <- dbGetQuery(con, paste("select gpl.bioc_package,gsm.gpl,",
+     "gsm,gsm.supplementary_file", "from gsm join gpl on gsm.gpl=gpl.gpl",
+     "where gpl.manufacturer='Affymetrix'",
+     "and gsm.supplementary_file like '%CEL.gz' "))
> rs[1:5, ]
```

```
  bioc_package   gpl    gsm
1       hu6800 GPL80 GSM575
2       hu6800 GPL80 GSM576
3       hu6800 GPL80 GSM577
4       hu6800 GPL80 GSM578
5       hu6800 GPL80 GSM579
                                                                    supplementary_file
1 ftp://ftp.ncbi.nih.gov/pub/geo/DATA/supplementary/samples/GSMnnn/GSM575/GSM575.cel.gz
2 ftp://ftp.ncbi.nih.gov/pub/geo/DATA/supplementary/samples/GSMnnn/GSM576/GSM576.cel.gz
3 ftp://ftp.ncbi.nih.gov/pub/geo/DATA/supplementary/samples/GSMnnn/GSM577/GSM577.cel.gz
4 ftp://ftp.ncbi.nih.gov/pub/geo/DATA/supplementary/samples/GSMnnn/GSM578/GSM578.cel.gz
5 ftp://ftp.ncbi.nih.gov/pub/geo/DATA/supplementary/samples/GSMnnn/GSM579/GSM579.cel.gz
```

Of course, we can combine programming and data access. A simple `sapply` example shows how to query each of the tables for number of records.

```
> getTableCounts <- function(tableName, conn) {
+     sql <- sprintf("select count(*) from %s",
+         tableName)
+     return(dbGetQuery(conn, sql)[1, 1])
+ }
> do.call(rbind, sapply(geo_tables, getTableCounts,
+     con, simplify = FALSE))
```

```
                     [,1]
gds                  2089
gds_subset          11858
geoConvert        1667634
geodb_column_desc     104
gpl                  6608
gse                 14136
gse_gpl             18414
gse_gsm            412699
gsm                375737
metaInfo                2
sMatrix             17624
```

3.3 Conversion of GEO entity types

Large-scale consumers of GEO data might want to convert GEO entity type from one to others, e.g. finding all GSM and GSE associated with 'GPL96'. Function goeConvert does the conversion with a very fast mapping between entity types.

Covert 'GPL96' to other possible types in the GEOmetadb.sqlite.

```
> conversion <- geoConvert("GPL96")
```

Check what GEO types and how many entities in each type in the conversion.

```
> lapply(conversion, dim)

$gse
[1] 712   2

$gsm
[1] 22352     2

$gds
[1] 254   2

$sMatrix
[1] 747   2
```

```
> conversion$gse[1:5, ]

  from_acc   to_acc
1    GPL96  GSE1000
2    GPL96 GSE10024
3    GPL96 GSE10043
4    GPL96 GSE10072
5    GPL96 GSE10089
```

```
> conversion$gsm[1:5, ]

  from_acc    to_acc
1    GPL96 GSM100454
2    GPL96 GSM100455
3    GPL96 GSM100456
4    GPL96 GSM100457
5    GPL96 GSM100458

> conversion$gds[1:5, ]

  from_acc  to_acc
1    GPL96 GDS1023
2    GPL96 GDS1036
3    GPL96 GDS1050
4    GPL96 GDS1062
5    GPL96 GDS1063

> conversion$sMatrix[1:5, ]

  from_acc                         to_acc
1    GPL96  GSE1000_series_matrix.txt.gz
2    GPL96 GSE10024_series_matrix.txt.gz
3    GPL96 GSE10043_series_matrix.txt.gz
4    GPL96 GSE10072_series_matrix.txt.gz
5    GPL96 GSE10089_series_matrix.txt.gz
```

3.4 More advanced queries

Now, for something a bit more complicated, we would like to find all the human breast cancer-related Affymetrix gene expression GEO series.

```
> sql <- paste("SELECT DISTINCT gse.title,gse.gse",
+     "FROM", "  gsm JOIN gse_gsm ON gsm.gsm=gse_gsm.gsm",
+     "  JOIN gse ON gse_gsm.gse=gse.gse", "  JOIN gse_gpl ON gse_gpl.gse=gse.gse",
+     "  JOIN gpl ON gse_gpl.gpl=gpl.gpl", "WHERE",
+     "  gsm.molecule_ch1 like '%total RNA%' AND",
+     "  gse.title LIKE '%breast cancer%' AND",
+     "  gpl.organism LIKE '%Homo sapiens%'",
+     sep = " ")
> rs <- dbGetQuery(con, sql)
> dim(rs)

[1] 158   2
```

```
> print(rs[1:5, ], right = FALSE)

  title
1 A Modular Analysis of Breast Cancer Reveals a Novel Low-Grade Molecular Signature in E
2 A Phase II Study of Neoadjuvant Gemcitabine Plus Doxorubicin Followed by Gemcitabine P
3 A Supervised Risk Predictor of Breast Cancer Based on Biological Subtypes
4 A functional and regulatory network associated with PIP expression in human breast can
5 A gene expression signature predicting the recurrence of tamoxifen-treated primary bre
  gse
1 GSE2294
2 GSE8465
3 GSE10886
4 GSE11627
5 GSE9893
```

Finally, it is probably a good idea to close the connection, please see *DBI* for detail.

```
> dbDisconnect(con)

[1] TRUE
```

If you want to remove old GEOmetadb.sqlite file before retrieve a new version from the server, execute the following codes:

```
> file.remove("GEOmetadb.sqlite")

[1] TRUE
```