Using the SRAdb Package to Query the Sequence Read Archive

Jack Zhu*and Sean Davis†

Genetics Branch, Center for Cancer Research,
National Cancer Institute,
National Institutes of Health

September 8, 2010

1 Introduction

High throughput sequencing technologies have very rapidly become standard tools in biology. The data that these machines generate are large, extremely rich. As such, the Sequence Read Archives (SRA) have been set up at NCBI in the United States, EMBL in Europe, and DDBJ in Japan to capture these data in public repositories in much the same spirit as MIAME-compliant microarray databases like NCBI GEO and EBI ArrayExpress.

Accessing data in SRA requires finding it first. This R package provides a convenient and powerful framework to do just that. In addition, SRAdb features functionality to determine availability of sequence files and to download files of interest.

SRA does not currently store aligned reads or any other processed data that might rely on alignment to a reference genome. However, NCBI GEO does often contain aligned reads for sequencing experiments and the SRAdb package can help to provide links to these data as well. In combination with the GEOmetadb and GEOquery packages, these data are also, then, accessible.

2 Getting Started

Since SRA is a continuously growing repository, the SRAdb SQLite file is updated regularly. The first step, then, is to get the SRAdb SQLite file from the online location. The download and uncompress steps are done automatically with a single command, getSRAdbFile.

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

```
> library(SRAdb)
> sqlfile <- getSRAdbFile()
```

The default storage location is in the current working directory and the default filename is "SRAmetadb.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 SRAdb, it is perhaps of some interest to know some details about the file itself.

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

```
                      size isdir mode
SRAmetadb.sqlite 567287808 FALSE  644
                                 mtime
SRAmetadb.sqlite 2010-09-08 17:50:17
                                 ctime
SRAmetadb.sqlite 2010-09-08 17:50:17
                                 atime uid gid
SRAmetadb.sqlite 2010-09-08 17:50:17 502  20
                     uname grname
SRAmetadb.sqlite biocbuild  staff
```

Then, create a connection for later queries. The standard DBI functionality as implemented in RSQLite function `dbConnect` makes the connection to the database. The `dbDisconnect` function disconnects the connection.

```
> sra_con <- dbConnect(SQLite(), sqlfile)
```

For further details, at this time see `help('SRAdb-package')`.

3 Using the SRAdb package

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.

```
> sra_con <- dbConnect(SQLite(), "SRAmetadb.sqlite")
```

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

```
> sra_tables <- dbListTables(sra_con)
> sra_tables
```

```
 [1] "col_desc"         "data_block"
 [3] "experiment"       "metaInfo"
 [5] "run"              "sample"
 [7] "sra"              "sra_ft"
 [9] "sra_ft_content"   "sra_ft_segdir"
[11] "sra_ft_segments" "study"
[13] "submission"
```

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

```
> dbListFields(sra_con, "study")
```

```
 [1] "study_ID"             "study_alias"
 [3] "study_accession"      "study_title"
 [5] "study_type"           "study_abstract"
 [7] "center_name"          "center_project_name"
 [9] "project_id"           "study_description"
[11] "study_url_link"       "study_entrez_link"
[13] "study_attribute"      "submission_accession"
[15] "sradb_updated"
```

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 *study* table.

```
> sqliteQuickSQL(sra_con, "PRAGMA TABLE_INFO(study)")
```

```
   cid                 name    type notnull
1    0             study_ID    REAL       0
2    1           study_alias    TEXT       0
3    2      study_accession    TEXT       0
4    3           study_title    TEXT       0
5    4            study_type    TEXT       0
6    5       study_abstract    TEXT       0
7    6           center_name    TEXT       0
8    7  center_project_name    TEXT       0
9    8            project_id INTEGER       0
10   9    study_description    TEXT       0
11  10       study_url_link    TEXT       0
12  11    study_entrez_link    TEXT       0
13  12      study_attribute    TEXT       0
14  13 submission_accession    TEXT       0
15  14        sradb_updated    TEXT       0
```

```
   dflt_value pk
1        <NA>  0
2        <NA>  0
3        <NA>  0
4        <NA>  0
5        <NA>  0
6        <NA>  0
7        <NA>  0
8        <NA>  0
9        <NA>  0
10       <NA>  0
11       <NA>  0
12       <NA>  0
13       <NA>  0
14       <NA>  0
15       <NA>  0
```

3.2 Writing SQL queries and getting results

Select 3 records from the *study* table and show the first 5 columns:

```
> rs <- dbGetQuery(sra_con, "select * from study limit 3")
> rs[, 1:5]

  study_ID                study_alias
1        1              Natto BEST195
2        2 Resequence B. subtilis 168
3        3           KU_MeDIPseq_2009
  study_accession
1       DRP000001
2       DRP000002
3       DRP000030
                                                                          study_titl
1                      Whole genome sequencing of Bacillus subtilis subsp. natto BEST19
2     Whole genome resequencing of Bacillus subtilis subsp. subtilis str.             16
3 Whole-genome DNA methylation analysis in human breast cancer cell lines using MeDIP-se
              study_type
1 Whole Genome Sequencing
2 Whole Genome Sequencing
3            Epigenetics
```

Get the SRA study accessions and titles from SRA study that study_type contains "Transcriptome". The "%" sign is used in combination with the "like" operator to do a "wildcard" search for the term "Transcriptome" with any number of characters after it.

```
> rs <- dbGetQuery(sra_con, paste("select study_accession,study_title from study where
+     "study_description like 'Transcriptome%'",
+     sep = " "))
> rs[1:3, ]
```

```
  study_accession
1       SRP000568
2       SRP000714
3       SRP001122

                                                        study_title
1                                                 Highly integrated epigenome maps in Ar
2 A Global View of Gene Activity and Alternative Splicing by Deep Sequencing of the Huma
3  A Global View of Gene Activity and Alternative Splicing by Deep Sequencing of the Hum
```

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(sra_tables, getTableCounts,
+     sra_con, simplify = FALSE))
```

```
                 [,1]
col_desc          102
data_block       5606
experiment      25027
metaInfo            2
run             65225
sample          75988
sra             66284
sra_ft          66284
sra_ft_content  66284
sra_ft_segdir      15
sra_ft_segments 28694
study            2730
submission      20044
```

3.3 Conversion of SRA entity types

Large-scale consumers of SRA data might want to convert SRA entity type from one to others, e.g. finding all experiment accessions (SRX, ERX or DRX) and run accessions (SRR,

ERR or DRR) associated with 'SRP001007'. Function sraConvert does the conversion with a very fast mapping between entity types.

Covert 'SRP001007' to other possible types in the SRAmetadb.sqlite.

```
> conversion <- sraConvert(c("SRP001007",
+     "SRP000931"), sra_con = sra_con)
> conversion[1:3, ]

      study submission    sample experiment
1 SRP000931  SRA009053 SRS003453  SRX006122
2 SRP000931  SRA009053 SRS003453  SRX006129
3 SRP000931  SRA009053 SRS003453  SRX006130
        run
1 SRR018256
2 SRR018263
3 SRR018264
```

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

```
> apply(conversion, 2, unique)

$study
[1] "SRP000931" "SRP001007"

$submission
[1] "SRA009053" "SRA009276"

$sample
 [1] "SRS003453" "SRS003454" "SRS003455"
 [4] "SRS003456" "SRS003457" "SRS003458"
 [7] "SRS003459" "SRS003460" "SRS003461"
[10] "SRS003462" "SRS003463" "SRS003464"
[13] "SRS004650"

$experiment
 [1] "SRX006122" "SRX006129" "SRX006130"
 [4] "SRX006123" "SRX006124" "SRX006125"
 [7] "SRX006126" "SRX006127" "SRX006128"
[10] "SRX006131" "SRX006132" "SRX006133"
[13] "SRX006134" "SRX006135" "SRX007396"

$run
 [1] "SRR018256" "SRR018263" "SRR018264"
 [4] "SRR018257" "SRR018258" "SRR018259"
```

```
 [7] "SRR018260" "SRR018261" "SRR018262"
[10] "SRR018265" "SRR018266" "SRR018267"
[13] "SRR018268" "SRR018269" "SRR020739"
[16] "SRR020740"
```

3.4 Full text search

Searching by regular table and field specific SQL commands can be very powerful and if you are familiar with SQL language and the table structure. If not, SQLite has a very handy module called Full text search (fts3), which allow users to do Google like search with terms and operators. The function getSRA does Full text search against all fields in a fts3 table with terms constructed with the Standard Query Syntax and Enhanced Query Syntax. Please see http://www.sqlite.org/fts3.html for detail.

Find all run and study combined records in which any given fields has 'breast' and 'cancer' words, including 'breast' and 'cancer' are not next to each other:

```
> rs <- getSRA(search_terms = "breast cancer",
+     out_types = c("run", "study"), sra_con = sra_con)
> dim(rs)

[1] 138  22
```

If you only wants records containing exact phrase of 'breast cancer', in which 'breast' and 'cancer' have other characters between other than a space:

```
> rs <- getSRA(search_terms = "\"breast cancer\"",
+     out_types = c("run", "study"), sra_con = sra_con)
> dim(rs)

[1] 95 22
```

Find all sample records containing words of either 'MCF7' or 'MCF-7':

```
> rs <- getSRA(search_terms = "MCF7 OR \"MCF-7\"",
+     out_types = c("sample"), sra_con = sra_con)
> dim(rs)

[1] 92 10
```

Find all submissions by GEO:

```
> rs <- getSRA(search_terms = "submission_center: GEO",
+     out_types = c("submission"), sra_con = sra_con)
> dim(rs)
```

```
[1] 402   7
```

Find study records containing a word beginning with 'Carcino':

```
> rs <- getSRA(search_terms = "Carcino*",
+     out_types = c("study"), sra_con = sra_con)
> dim(rs)
```

```
[1] 21 12
```

3.5 Get fastq file information

List fastq file names including ftp addresses associated with "SRA000045":

```
> listFastq("SRA011804", sra_con = sra_con)
```

The above function does not check file availability, size and date of the fastq files on the server, but the function getFastqInfo does this, which is good to know if you are preparing to download them:

```
> rs <- getFastqInfo(in_acc = c("SRA011804"),
+     sra_con = sra_con)
> rs[1:3, ]
```

Next you might want to download fastq files from the ftp site. The getFastq function will download all available fastq files associated with "SRR000648" and "SRR000657" from NCBI SRA ftp site to a new folder in current directory:

```
> getFastq(in_acc = c("SRR000648", "SRR000657"),
+     sra_con = sra_con, destdir = getwd())
```

4 Interactive views of sequence data

Working with sequence data is often best done interactively in a genome browser, a task not easily done from R itself. We have found the Integrative Genomics Viewer (IGV) a high-performance visualization tool for interactive exploration of large, integrated datasets, increasing usefully for visualizing sequence alignments. In SRAdb, functions `startIGV`, `load2IGV` and `load2newIGV` provide convenient functionality for R to interact with IGV. Note that for some OS, these functions might not work or work well.

Launch IGV with 2 GB maximum usable memory support:

```
> startIGV("mm")
```

IGV offers a remort control port that allows R to communicate with IGV. The current command set is fairly limited, but it does allow for some IGV operations to be performed in the R console. To utilize this functionality, be sure that IGV is set to allow communication via the "enable port" option in IGV preferences. To load BAM files to IGV and then manipulate the window:

```
> exampleBams = file.path(system.file("extdata",
+     package = "SRAdb"), dir(system.file("extdata",
+     package = "SRAdb"), pattern = "bam$"))
> IGVgenome("hg18")
> IGVload(exampleBams)
> IGVgoto("chr1:1-1000")
> IGVsnapshot()
```

5 Graphic view of SRA entities

Due to the nature of SRA data and its design, sometimes it is hard to get a whole picture of the relationship between a set of SRA entities. Functions of `entityGraph` and `sraGraph` in this package generate graphNEL objects with edgemode='directed' from input data.frame or directly from search terms, and then the `plot` function can easily draw a graph.

Create a graphNEL object from SRA accessions, which are full text search results of terms 'colon cancer'

```
> acc <- getSRA(search_terms = "colon cancer",
+     out_types = c("sra"), sra_con = sra_con,
+     acc_only = TRUE)
> g <- entityGraph(acc)
> attrs <- getDefaultAttrs(list(node = list(fillcolor = "lightblue",
+     shape = "ellipse")))
> plot(g, attrs = attrs)
```

Create a graphNEL object directly from full text search results of terms 'colon cancer'

```
> g <- sraGraph("colon cancer", sra_con)
> library(Rgraphviz)
> attrs <- getDefaultAttrs(list(node = list(fillcolor = "lightblue",
+     shape = "ellipse")))
> plot(g, attrs = attrs)
```

6 sessionInfo

```
> toLatex(sessionInfo())
```

- R version 2.11.1 Patched (2010-05-31 r52167), `i386-apple-darwin9.8.0`

- Locale: `C/en_US.UTF-8/C/C/C/C`
- Base packages: base, datasets, grDevices, graphics, methods, stats, tools, utils
- Other packages: DBI 0.2-5, RSQLite 0.9-2, SRAdb 1.2.1, graph 1.26.0
- Loaded via a namespace (and not attached): Biobase 2.8.0, GEOquery 2.13.5, RCurl 1.4-3, XML 3.1-1