Package ‘SRAdb’

September 24, 2012

Type Package

Title A compilation of metadata from NCBI SRA and tools

Version 1.10.0

Date 2012-02-07

Depends RSQLite (>= 0.8-4) , graph, RCurl

Imports GEOquery

Suggests Rgraphviz

Author Jack Zhu and Sean Davis

Maintainer Jack Zhu <zhujack@mail.nih.gov>

biocViews Infrastructure, HighThroughputSequencing, DataImport

Description The Sequence Read Archive (SRA) is the largest public repository of sequencing data from the next generation of sequencing platforms including Roche 454 GS System, Illumina Genome Analyzer,Applied Biosystems SOLiD System, Helicos Heliscope, and others. However, finding data of interest can be challenging using current tools. SRAdb is an attempt to make access to the metadata associated with submission, study, sample, experiment and run much more feasible. This is accomplished by parsing all the NCBI SRA metadata into a SQLite database that can be stored and queried locally. Fulltext search in the package make querying metadata very flexible and powerful. sra or sra-lite files can be downloaded for doing alignment locally. The SQLite database is updated regularly as new data is added to SRA and can be downloaded at will for the most up-to-date metadata.

License Artistic-2.0

LazyLoad yes

URL http://watson.nci.nih.gov/

R topics documented:

`SRAdb-package` *Query NCBI SRA metadata within R or from a local SQLite database*

Description

The Sequence Read Archive (SRA) represents largest public repository of sequencing data from the next generation of sequencing platforms including Roche 454 GS System, Illumina Genome Analyzer, Applied Biosystems SOLiD System, Helicos Heliscope, and others. However, finding data of interest can be challenging using current tools. SRAdb is an attempt to make access to the metadata associated with submission, study, sample, experiment and run much more feasible. This is accomplished by parsing all the NCBI SRA metadata into a SQLite database that can be stored and queried locally. SRAdb is simply a thin wrapper around the SQLite database along with associated tools and documentation. Fulltext search in the package make querying metadata very flexible and powerful. SRA data files (sra or sra-lite) can be downloaded for doing alignment locally. Available BAM files in local or in the Meltzerlab sraDB can be loaded into IGV for visualization easily. The SQLite database is updated regularly as new data is added to SRA and can be downloaded at will for the most up-to-date metadata.

Details

Package:	SRAdb
Type:	Package
Version:	1.9.2
Date:	2012-02-13
License:	What license is it under?
LazyLoad:	yes

Author(s)

Jack Zhu and Sean Davis

Maintainer: Jack Zhu <zhujack@mail.nih.gov>

References

http://watson.nci.nih.gov/~zhujack/SRAmetadb.sqlite.gz

Examples

```
if(file.exists('SRAmetadb.sqlite')) {

  library(SRAdb)
  sra_dbname <- 'SRAmetadb.sqlite'
  sra_con <- dbConnect(dbDriver("SQLite"), sra_dbname)

  ## Get column descriptions
  a <- colDescriptions(sra_con=sra_con)[1:5,]

  ## Convert SRA experiment accessions to other types
  b <- sraConvert( in_acc=c(" SRR000137", "SRR000138 "), out_type=c('sample'), sra_con=sra_con )

  ## Fulltext search SRA meta data using SQLite fts3 module
  rs <- getSRA (search_terms ='breas* NEAR/2 can*', out_types=c('run','study'), sra_con=sra_con)
  rs <- getSRA (search_terms ='breast', out_types=c('run','study'), sra_con=sra_con)
  rs <- getSRA (search_terms ='"breas* can*"', out_types=c('study'), sra_con=sra_con)
  rs <- getSRA (search_terms ='MCF7 OR "MCF-7"', out_types=c('sample'), sra_con=sra_con)
  rs <- getSRA (search_terms ='study_title: brea* can*', out_types=c('run','study'), sra_con=sra_con)
  rs <- getSRA (search_terms ='study_title: brea* can*', out_types=c('run','study'), sra_con=sra_con, acc_c

  ## List fastq file ftp or fasp addresses associated with "SRX000122"
  listSRAfile (in_acc = c("SRX000122"), sra_con = sra_con, fileType = 'sra')
  listSRAfile (in_acc = c("SRX000122"), sra_con = sra_con, fileType = 'sra', srcType='fasp')

  ## Get file size and date from NCBI ftp site for available fastq files associated with "SRS012041","SRS0
  ## Not run:
  getSRAinfo (in_acc=c("SRS012041","SRS000290"), sra_con=sra_con, sraType='litesra')

## End(Not run)

  ## Download litesra files from NCBI SRA using ftp protocol:
  getSRAfile( in_acc = c("SRR000648","SRR000657"), sra_con = sra_con, destDir = getwd(), fileType = 'lites
  ## Download fastq files from EBI using ftp protocol:
  getSRAfile( in_acc, sra_con, destDir = getwd(), fileType = 'fastq', srcType = 'ftp', makeDirectory = FALS

  ## Download fastq files from EBI  ftp siteusing fasp protocol:
  ## Not run:
  ascpCMD <-  'ascp -QT -l 300m -i /usr/local/aspera/connect/etc/asperaweb_id_dsa.putty'
  getSRAfile( in_acc, sra_con,  fileType = 'fastq', srcType = 'fasp',  ascpCMD = ascpCMD )
## End(Not run)

  ## Start IGV from R if no IGV running
  ## Not run: startIGV(memory='mm')

  ## load BAM files to IGV
  ## Not run:
  exampleBams = file.path(system.file('extdata',package='SRAdb'), dir(system.file('extdata',package='SRAdb'
  sock <- IGVsocket()
  IGVload(sock,exampleBams)

## End(Not run)
  ## Change the IGV genome
  ## Not run:
  IGVgenome(sock,genome='hg18')

## End(Not run)
  ## Go to a specified region in IGV
  ## Not run:
  IGVgoto(sock,'chr1:1-10000')
  IGVgoto(sock,'TP53')

## End(Not run)

  ## Make a snapshot of the current IGV window
  ## Not run:
  IGVsnapshot(sock)
  dir()

## End(Not run)

  ## create a graphNEL object from SRA accessions, which are full text search results of terms 'primary thy
  g <- sraGraph('primary thyroid cell line', sra_con)

  ## Not run:
  library(Rgraphviz)
  attrs <- getDefaultAttrs(list(node=list(fillcolor='lightblue', shape='ellipse')))
  plot(g, attrs=attrs)

## End(Not run)

  dbDisconnect(sra_con)

} else {
  print("use getSRAdbFile() to get a copy of the SRAmetadb.sqlite file
and then rerun the example")
}
```

`ascpR` *Fasp file downloading using the ascp command line program*

Description

This function downloads files by fasp protocol using Aspera's ascp command line program, which is include in Aspera Connect software (http://www.asperasoft.com/).

Usage

```
ascpR( ascpCMD, ascpSource, destDir = getwd() )
```

Arguments

`ascpCMD`	ascp main commands, which should be constructed by a user according to the actual installation of Aspera Connect in the system, with proper options to be used. Example commands: "ascp -QT -l 300m -i /usr/local/aspera/connect/etc/asperaweb_id_dsa.putty" (Linux) or "'/Applications/Aspera Connect.app/Contents/Resources/ascp' -QT -l 300m -i '/Applications/Aspera Connect.app/Contents/Resources/asperaweb_id_dsa.putty'" (Mac OS X). More about ascp please see the help ('ascp -h' in a shell).
`ascpSource`	character vector of fasp file sources for the ascp command, e.g. era-fasp@fasp.sra.ebi.ac.uk:vol1/fastq (EBI), anonftp@ftp-trace.ncbi.nlm.nih.gov:/sra/sra-instant/reads/ByExp/litesra/SRX/SRX000/SRX0 (NCBI).
`destDir`	destination directory to save downloaded files.

Details

The function takes advatage of Aspera's fasp transport technology (http://www.asperasoft.com/), which provides high-speed transfering large files over the Internet. Due to complexity with options with ascp and installation difference between different systems, this funciton asks users to supply main ascp comands. Users who are not familiar with ascp command line program should have IT support personnel to install the software and constrct main ascp comands.

Value

A data.frame containing all matched SRA accessions and ftp or fasp addresses.

Author(s)

Jack Zhu <zhujack@mail.nih.gov>

References

http://www.asperasoft.com/

See Also

`ascpSRA`, `getSRAfile`, `getFASTQinfo`, `getSRAinfo`

Examples

```
if( file.exists('SRAmetadb.sqlite') ) {
## Not run:
library( SRAdb )
sra_dbname <- 'SRAmetadb.sqlite'
sra_con <- dbConnect(dbDriver("SQLite"), sra_dbname)
rs <- getFASTQinfo (in_acc=c("SRR000648","SRR000657"), srcType='fasp')

ascpSource <- rs$fasp
ascpCMD <- 'ascp -QT -l 300m -i /usr/local/aspera/connect/etc/asperaweb_id_dsa.putty'
## common ascpCMD in Mac OS X:
#ascpCMD = "'/Applications/Aspera Connect.app/Contents/Resources/ascp' -QT -l 300m -i '/Applications/Aspera

ascpR( ascpCMD, ascpSource, destDir = getwd() )
dbDisconnect( sra_con )

## End(Not run)
} else {
print( "Use getSRAdbFile() to get a copy of the SRAmetadb.sqlite file  and then rerun the example" )
}
```

---

ascpSRA — *Fasp SRA data file downloading using the ascp command line program*

Description

This function downloads SRA data files (fastq, sra or litesra) by fasp protocol using Aspera's ascp command line program, which is included in Aspera Connect software (http://www.asperasoft.com/).

Usage

```
ascpSRA ( in_acc, sra_con, ascpCMD, fileType = 'litesra', destDir = getwd() )
```

Arguments

in_acc	character vector of SRA accessions, which should be in same SRA data type, either submission, study, sample, experiment or run.
sra_con	connection to the SRAmetadb SQLite database.
ascpCMD	ascp main commands, which should be constructed by a user according to the actual installation of Aspera Connect in the system, with proper options to be used. Example commands: "ascp -QT -l 300m -i /usr/local/aspera/connect/etc/asperaweb_id_dsa.putty" (Linux) or "'/Applications/Aspera Connect.app/Contents/Resources/ascp' -QT -l 300m -i '/Applications/Aspera Connect.app/Contents/Resources/asperaweb_id_dsa.putty'" (Mac OS X). More about ascp please see the help ('ascp -h' in a shell).
fileType	type of SRA data files, which should be "sra", "litesra" or "fastq".
destDir	destination directory to save downloaded files.

Details

This function will get fasp file sources first using funciton `listSRAfile` and then download data files using function `ascpR`.

Value

A data.frame of all matched SRA accessions and ftp or fasp file addresses.

Author(s)

Jack Zhu <zhujack@mail.nih.gov>

References

http://www.asperasoft.com/

See Also

ascpR, listSRAfile, getSRAfile, getFASTQinfo, getSRAinfo

Examples

```
if( file.exists('SRAmetadb.sqlite') ) {
## Not run:
library(SRAdb)
sra_dbname <- 'SRAmetadb.sqlite'

sra_con <- dbConnect(dbDriver("SQLite"), sra_dbname)
in_acc <- c("SRR000648","SRR000657")
ascpCMD <- 'ascp -QT -l 300m -i /usr/local/aspera/connect/etc/asperaweb_id_dsa.putty'
## common ascpCMD for a system with Mac OS X:
#ascpCMD <- "'/Applications/Aspera Connect.app/Contents/Resources/ascp' -QT -l 300m -i '/Applications/Aspe

sraFiles <- ascpSRA( in_acc, sra_con, ascpCMD, fileType = 'litesra', destDir=getwd() )
dbDisconnect(sra_con)

## End(Not run)
} else {
  print( "Use ascpSRAdbFile() to get a copy of the SRAmetadb.sqlite file  and then rerun the example" )
}
```

---

colDescriptions *Get column descriptions of SRAmetadb.sqlite*

Description

Get column descriptions of SRAmetadb.sqlite, including table, field, field data type, description and default values

Usage

```
colDescriptions( sra_con )
```

Arguments

sra_con	Connection of the SRAmetadb SQLite database

Value

A seven-column data.frame including table_name, field_name, type, description, value_list.

Author(s)

Jack Zhu<zhujack@mail.nih.gov> and Sean Davis <sdavis2@mail.nih.gov>

Examples

```
if(file.exists('SRAmetadb.sqlite')) {

  library(SRAdb)
  sra_dbname <- 'SRAmetadb.sqlite'
  sra_con <- dbConnect(dbDriver("SQLite"), sra_dbname)

  ## Get column descriptions
  a <- colDescriptions(sra_con=sra_con)[1:5,]

} else {
  print("use getSRAdbFile() to get a copy of the SRAmetadb.sqlite file and then rerun the example")
}
```

---

entityGraph — *Create a new graphNEL object from an input entity matrix or data.frame*

Description

This function will create a new graphNEL object from an input entity matrix or data.frame

Usage

```
entityGraph(df)
```

Arguments

df	A matrix or data.frame

Details

A graphNEL object with edgemode='directed' is created from input data.frame and the `plot` function will draw a graph

Value

A graphNEL object with edgemode='directed'

Author(s)

Jack Zhu <zhujack@mail.nih.gov> and Sean Davis <sdavis2@mail.nih.gov>

See Also

getSRA, sraConvert, sraGraph

Examples

```
if(file.exists('SRAmetadb.sqlite')) {

library(SRAdb)
sra_dbname <- 'SRAmetadb.sqlite'
sra_con <- dbConnect(dbDriver("SQLite"), sra_dbname)

## create a graphNEL object from SRA accessions, which are full text search results of terms 'primary thyro
   acc <- getSRA (search_terms ='primary thyroid cell line', out_types=c('sra'), sra_con=sra_con, acc_only=
   g <- entityGraph(acc)
   ## Not run:
   library(Rgraphviz)
   attrs <- getDefaultAttrs(list(node=list(fillcolor='lightblue', shape='ellipse')))
   plot(g, attrs= attrs)

## End(Not run)

} else {
   print("use getSRAdbFile() to get a copy of the SRAmetadb.sqlite file
and then rerun the example")
}
```

---

getFASTQfile *Download SRA fastq files from EBI ENA through ftp or fasp*

Description

This function downloads SRA fastq data files through ftp or fasp from EBI ENA site for a given list of SRA accessions.

Usage

```
getFASTQfile( in_acc, destDir = getwd(), srcType = 'ftp', makeDirectory = FALSE, method = 'curl'
```

Arguments

in_acc	character vector of SRA accessions that could be be in one or more SRA sata types: study, sample, experiment and/or run.
destDir	destination directory to save downloaded fastq files
srcType	type of transfer protocol, which should be "ftp" or "fasp".
makeDirectory	logical, TRUE or FALSE. If TRUE and baseDir does not exists, storedir will be created to save downloaded files, otherwise downloaded fastq files will be saved to current directory.
method	character vector of length 1, passed to the identically named argument of download.file.

`ascpCMD`	ascp main commands, which should be constructed by a user according to the actual installation of Aspera Connect in the system, with proper options to be used. Example commands: "ascp -QT -l 300m -i /usr/local/aspera/connect/etc/asperaweb_id_dsa.putty" (Linux) or "'/Applications/Aspera Connect.app/Contents/Resources/ascp' -QT -l 300m -i '/Applications/Aspera Connect.app/Contents/Resources/asperaweb_id_dsa.putty'" (Mac OS X). More about ascp please see the help ('ascp -h' in a shell).

Details

The function first gets ftp/fasp addresses of SRA fastq files using funcitn `getFASTQinfo` for a given list of input SRA accessions; then downloads the fastq files through ftp or fasp.

Warning

Downloading SRA fastq files through ftp over long distance could take long time and should consider using using 'fasp'.

Author(s)

Jack Zhu <zhujack@mail.nih.gov>

See Also

`getFASTQinfo`, `getSRAfile`, `ascpR`

Examples

```
if(file.exists('SRAmetadb.sqlite')) {
## Not run:
library(SRAdb)
sra_dbname <- 'SRAmetadb.sqlite'
sra_con <- dbConnect( dbDriver("SQLite"), sra_dbname )

## Download fastq files from EBI ENA  through ftp
getFASTQfile( in_acc = c("SRR000648","SRR000657"), destDir = getwd(), srcType = 'ftp', ascpCMD = NULL )

## Download fastq files from EBI ENA  through fasp
ascpCMD <- 'ascp -QT -l 300m -i /usr/local/aspera/connect/etc/asperaweb_id_dsa.putty'
## common ascpCMD for a system with Mac OS X:
#ascpCMD <- "'/Applications/Aspera Connect.app/Contents/Resources/ascp' -QT -l 300m -i '/Applications/Aspe
getFASTQfile( in_acc = c("SRR000648","SRR000657"), srcType='fasp', ascpCMD=ascpCMD )

dbDisconnect( sra_con )

## End(Not run)
} else {
print("use getSRAdbFile() to get a copy of the SRAmetadb.sqlite file and then rerun the example")
}
```

`getFASTQinfo` *Get SRA fastq file information and associated meta data from EBI ENA*

Description

This function gets SRA fastq file information and essential associated meta data from EBI ENA web site (http://www.ebi.ac.uk/ena/data/view/reports/sra/fastq_files/) for SRA accessions given.

Usage

```
getFASTQinfo( in_acc, srcType = 'ftp' )
```

Arguments

`in_acc`	character vector of SRA accessions that could be be in one or more SRA sata types: study, sample, experiment and/or run.
`srcType`	option for listing either 'ftp' or 'fasp' addresses. The default is 'ftp'.

Details

EBI ENA web site (http://www.ebi.ac.uk/ena/data/view/reports/sra/fastq_files/) is the souce for parsing infromation from, which is updated and verified daily. Ftp or fasp addresses got from this funciton can be used in either `getFASTQfile` or `getSRAfile` to download the files.

Value

A data.frame of ftp/fasp inftomation (addresses, file size, read number, etc) and associated meta data (study, sample, experiment, run, organism, instrument.platform, instrument.model, library.name, library.layout, library.source, library.selection, run.read.count, run.base.count, etc.).

Author(s)

Jack Zhu <zhujack@mail.nih.gov>

See Also

`getFASTQfile`, `listSRAfile`, `getSRAfile`

Examples

```
if(file.exists('SRAmetadb.sqlite')) {
## Not run:
library(SRAdb)
sra_dbname <- 'SRAmetadb.sqlite'
sra_con <- dbConnect(dbDriver("SQLite"), sra_dbname)
getFASTQinfo( in_acc = c("SRR000648","SRR000657"), srcType = 'ftp' )
getFASTQinfo( in_acc = c("SRR000648","SRR000657"), srcType = 'fasp' )

## End(Not run)
} else {
print("Use getSRAdbFile() to get a copy of the SRAmetadb.sqlite file  and then rerun the example")
}
```

getSRA — *Fulltext search SRA meta data using SQLite fts3 module*

Description

This function does Fulltext search on any SRA fields in any SRA data types with Fulltext capacity in the SQLite and returns SRA records

Usage

```
getSRA(search_terms, out_types=c('sra','submission','study','sample','experiment','run'), sra_cc
```

Arguments

`search_terms`	Free text search terms constructed according to SQLite query syntax defined here: http://www.sqlite.org/fts3.html#section_1_3
`out_types`	Character vector of the following SRA data types: 'sra','submission','study','sample','experiment','ru Note: if 'sra' is within out_types, the out_types will be set to c('submission','study','sample','experim
`sra_con`	Connection to the SRAmetadb SQLite database
`acc_only`	logical, if TRUE, the function will return SRA accession for each out_types

Details

Queries performed by this function could be Phrase queries, e.g. '"lin* app*"', or NEAR queries, e.g. '"ACID compliant" NEAR/2 sqlite', or with the Enhanced Query Syntax. Check Full Text Search section on the SQLite site for details. if 'acc_only=TRUE', a data.frame containing only SRA accessions will be returned, which can be used as input for `sraGraph`.

Value

A data.frame containing all returned SRA records with fields defined by out_types.

If acc_only=FALSE, a data.frame of matched accessions of out_types will be returned.

Author(s)

Jack Zhu <zhujack@mail.nih.gov>

References

http://www.sqlite.org/

See Also

`sraConvert`

Examples

```
if(file.exists('SRAmetadb.sqlite')) {

library(SRAdb)
sra_dbname <- 'SRAmetadb.sqlite'
sra_con <- dbConnect(dbDriver("SQLite"), sra_dbname)

## Fulltext search SRA meta data using SQLite fts3 module:
# find all records with words of 'breast' and 'cancer' in a filed and there could be one to many words betw
rs <- getSRA (search_terms ='breast cancer', out_types=c('run','study'), sra_con=sra_con)

# find all records with exact phrase of 'breast cancer' in a filed:
rs <- getSRA (search_terms ='"breast cancer"', out_types=c('run','study'), sra_con=sra_con)

# find records with words beginning with 'braes' and 'can', and the distance between them is equal or less
rs <- getSRA (search_terms ='breas* NEAR/2 can*', out_types=c('run','study'), sra_con=sra_con)

# the same as above except that only one space between the two words
rs <- getSRA (search_terms ='"breas* can*"', out_types=c('study'), sra_con=sra_con)

# find records with 'MCF7' or 'MCF-7' - adding double quote to avoid the SQLite to break down 'MCF-7' to 'M
rs <- getSRA (search_terms ='MCF7 OR "MCF-7"', out_types=c('sample'), sra_con=sra_con)

# the same as above, but only search the field of 'study_title':
rs <- getSRA (search_terms ='study_title: brea* can*', out_types=c('run','study'), sra_con=sra_con)

# the same as above, but only search the field of 'study_title' and return only accessions:
rs <- getSRA (search_terms ='study_title: brea* can*', out_types=c('run','study'), sra_con=sra_con, acc_onl

} else {
   print("use getSRAdbFile() to get a copy of the SRAmetadb.sqlite file  and then rerun the example")
}
```

---

getSRAdbFile — *Download and unzip last version of SRAmetadb.sqlite.gz from the server*

Description

This function is the standard method for downloading and unzipping the most recent SRAmetadb SQLite file from the server.

Usage

```
getSRAdbFile(destdir = getwd(), destfile = "SRAmetadb.sqlite.gz",
               method)
```

Arguments

destdir	The destination directory of the downloaded file
destfile	The filename of the downloaded file. This filename should end in ".gz" as the unzipping assumes that is the case
method	Character vector of length 1, passed to the identically named argument of download.file.

Value

Prints some diagnostic information to the screen.

Returns the local filename for use later.

Author(s)

Jack Zhu <zhujack@mail.nih.gov>, Sean Davis <sdavis2@mail.nih.gov>

Examples

```
## Not run: geometadbfile <- getSRAdbFile()
```

`getSRAfile` *Download SRA data file through ftp or fasp*

Description

This function downloads sra and sra-lite data files associated with input SRA accessions from NCBI SRA or downloads fastq files from EBI ENA through ftp or fasp protocol.

Usage

```
getSRAfile( in_acc, sra_con, destDir = getwd(), fileType = 'litesra', srcType = 'ftp', makeDirec
```

Arguments

`in_acc`	character vector of SRA accessions, which should be in same SRA data type, either submission, study, sample, experiment or run.
`sra_con`	Connection to the SRAmetadb SQLite database
`destDir`	destination directory to save downloaded files.
`fileType`	type of SRA data files, which should be "sra", "litesra" or "fastq".
`srcType`	type of transfer protocol, which should be "ftp" or "fasp".
`makeDirectory`	logical, TRUE or FALSE. If TRUE and baseDir does not exists, storedir will be created to save downloaded files, otherwise downloaded fastq files will be saved to current directory.
`method`	Character vector of length 1, passed to the identically named argument of `download.file`.
`ascpCMD`	ascp main commands, which should be constructed by a user according to the actual installation of Aspera Connect in the system, with proper options to be used. Example commands: "ascp -QT -l 300m -i /usr/local/aspera/connect/etc/asperaweb_id_dsa.putty" (Linux) or "'/Applications/Aspera Connect.app/Contents/Resources/ascp' -QT -l 300m -i '/Applications/Aspera Connect.app/Contents/Resources/asperaweb_id_dsa.putty'" (Mac OS X). More about ascp please see the help ('ascp -h' in a shell).

Details

The function first gets ftp/fasp addresses of SRA data files with funcitn `getSRAinfo` for a given list of input SRA accessions; then downloads the SRA data files through ftp or fasp. The sra or sra-lite data files are downloaded from NCBI SRA and the fastq files are downloaded from EBI ENA.

Warning

Downloading SRA data files through ftp over long distance could take long time and should consider using using 'fasp'.

Author(s)

Jack Zhu <zhujack@mail.nih.gov>

See Also

`listSRAfile`, `getSRAinfo`, `getFASTQinfo`, `getFASTQfile`

Examples

```
if( file.exists( 'SRAmetadb.sqlite' ) ) {

library(SRAdb)
sra_dbname <- 'SRAmetadb.sqlite'
sra_con <- dbConnect( dbDriver("SQLite"), sra_dbname )

## Not run:
## Download litesra files from NCBI SRA using ftp protocol:
getSRAfile( in_acc = c("SRR000648","SRR000657"), sra_con = sra_con, destDir = getwd(), fileType = 'litesra

## Convert NCBI SRA format (.lite.sra or .sra) data to fastq:
## Download SRA Toolkit: http://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?cmd=show&f=software&m=software&s=
## Run fastq-dump to
## system ("fastq-dump SRR000648.lite.sra")

## Download fastq files from EBI using ftp protocol:
getSRAfile( in_acc, sra_con, destDir = getwd(), fileType = 'fastq', srcType = 'ftp', makeDirectory = FALSE

## Download fastq files from EBI  ftp siteusing fasp protocol:
ascpCMD <-  'ascp -QT -l 300m -i /usr/local/aspera/connect/etc/asperaweb_id_dsa.putty'
getSRAfile( in_acc, sra_con,  fileType = 'fastq', srcType = 'fasp',  ascpCMD = ascpCMD )

## End(Not run)

dbDisconnect( sra_con )
} else {
print( "use getSRAdbFile() to get a copy of the SRAmetadb.sqlite file and then rerun the example" )
}
```

`getSRAinfo` *Get SRA data file information from NCBI SRA*

Description

This function gets SRA .sra or .lite.sra file information from NCBI SRA ftp site for a given list SRA accessions.

Usage

```
getSRAinfo( in_acc, sra_con, sraType = 'litesra' )
```

Arguments

`in_acc`	character vector of SRA accessions, which should be in same SRA data type, either submission, study, sample, experiment or run.
`sra_con`	connection to the SRAmetadb SQLite database
`sraType`	type of SRA data files, which should be either 'sra' or 'litesra'.

Details

The function first gets ftp addressed of sra or sra-lite data files with function `listSRAfile` and then get file size and date from NCBI SRA ftp sites.

Value

A data.frame of ftp addresses of SRA data files, and file size and date along with input SRA accessions.

Author(s)

Jack Zhu <zhujack@mail.nih.gov>

See Also

`listSRAfile`, `getSRAfile`

Examples

```
if(file.exists('SRAmetadb.sqlite')) {

library(SRAdb)
sra_dbname <- 'SRAmetadb.sqlite'
sra_con <- dbConnect(dbDriver("SQLite"), sra_dbname)

## Get file size and date from NCBI ftp site for available fastq files associated with "SRS012041","SRS0002
# getSRAinfo (in_acc=c("SRS012041","SRS000290"), sra_con=sra_con, sraType='litesra')

} else {
   print(" use getSRAdbFile() to get a copy of the SRAmetadb.sqlite file  and then rerun the example")
}
```

`IGVclear` *Clear IGV tracks loaded.*

Description

Clear IGV tracks loaded in the current IGV.

Usage

```
IGVclear(sock)
```

Arguments

`sock`	A socket connection to IGV.

Author(s)

Jack Zhu <zhujack@mail.nih.gov>

References

http://www.broadinstitute.org/igv/PortCommands

See Also

startIGV, IGVload, IGVgoto

Examples

```
  ## Not run:
  ## Create a file list from example bam files in the package
  exampleBams = file.path(system.file('extdata',package='SRAdb'),
    dir(system.file('extdata',package='SRAdb'),pattern='bam$'))

  ##Create a socket connection to IGV
  sock <- IGVsocket()
  ## Load the bam files into IGV
  IGVload(sock, exampleBams)

  ## Clear loaded tracks in the current IGV
  IGVclear(sock)

## End(Not run)
```

IGVcollapse *Collapse tracks in the IGV*

Description

Using the remote command port of IGV, this function collapses tracks in the IGV.

Usage

```
IGVcollapse(sock)
```

Arguments

sock	A socket connection to IGV.

Author(s)

Jack Zhu <zhujack@mail.nih.gov>

References

http://www.broadinstitute.org/igv/PortCommands

See Also

`startIGV`, `IGVload`

Examples

```
  ## Not run:
  sock <- IGVsocket()
  IGVcollapse(sock)

## End(Not run)
```

`IGVgenome` *Set the IGV genome.*

Description

Set the IGV genome via the remote command port.

Usage

```
IGVgenome(sock, genome="hg18")
```

Arguments

`sock`	A socket connection to IGV.
`genome`	String representing a genome that IGV knows about.

Author(s)

Sean Davis <sdavis2@mail.nih.gov>

References

http://www.broadinstitute.org/igv/PortCommands

See Also

`startIGV`

Examples

```
## Not run:
sock <- IGVsocket()
IGVgenome(sock, genome='hg18')

## End(Not run)
```

IGVgoto — *Go to a specified region in IGV.*

Description

Using the remote command port of IGV, go to a specified region.

Usage

```
IGVgoto(sock, region)
```

Arguments

`sock`	A socket connection to IGV.
`region`	Scrolls to a locus. Use any text that is valid in the IGV search box.

Author(s)

Sean Davis <sdavis2@mail.nih.gov>

References

http://www.broadinstitute.org/igv/PortCommands

See Also

`startIGV`, `IGVload`

Examples

```
  ## Not run: 
  sock <- IGVsocket()
  IGVgoto(sock, 'chr1:1-10000')
  IGVgoto(sock, 'TP53')

## End(Not run)
```

IGVload — *Load data into IGV via remote port call.*

Description

Loads data via a remote call to IGV.

Usage

```
IGVload(sock, files)
```

Arguments

`sock`	A socket connection to IGV.
`files`	Character vector of one or more filenames with full path or urls to load. Among supported file types are BAM and IGV session file, for other file types please check IGV web site: http://www.broadinstitute.org/igv/ControlIGV.

Author(s)

Sean Davis <sdavis2@mail.nih.gov>

References

http://www.broadinstitute.org/igv/PortCommands

See Also

`startIGV`, `IGVgoto`

Examples

```
  ## Not run: 
  ## Create a file list from example bam files in the package
  exampleBams = file.path(system.file('extdata',package='SRAdb'),
    dir(system.file('extdata',package='SRAdb'),pattern='bam$'))

  ## Create a socket connection to IGV
  sock <- IGVsocket()
  ## Load the bam files into IGV
  IGVload(sock, exampleBams)

## End(Not run)
```

`IGVsession` *Create an IGV session file*

Description

This function will create an IGV session file

Usage

```
IGVsession(files, sessionFile, genome='hg18', VisibleAttribute='', destdir=getwd())
```

Arguments

`files`	Character vector of one or more filenames or urls to load - required.
`sessionFile`	String representing session file name - required
`genome`	String representing a genome that IGV knows about.
`VisibleAttribute`	Character vector of one or more IGV Visible Attributes to annotate data tracks to be loaded - optional.
`destdir`	Path where to save the IGV session file.

Details

While the current state of an IGV session can be saved to a named session file that can be opened to restore the IGV session later on, a IGV session file can be manually or programmatically created to achieve more efficient data loading and better control of IGV. IGVsession function was developed to create such IGV session files. For details please check IGV web site: http://www.broadinstitute.org/igv/ControlIGV

Value

An IGV session file with full file path.

Author(s)

Jack Zhu <zhujack@mail.nih.gov>

See Also

`IGVload`, `IGVgenome`, `IGVgoto`

Examples

```
library(SRAdb)
exampleBams = file.path(system.file('extdata',package='SRAdb'),
    dir(system.file('extdata',package='SRAdb'),pattern='bam$'))
exampleSessionFile <- IGVsession(exampleBams, 'exampleBams.xml');
   ## Not run:
   ## Start IGV within R. You only need one IGV instance with listen port 60151 open.
    startIGV()

## Create a socket connection to IGV
sock <- IGVsocket()
## Wait until IGV fully launched and make sure the listen port for IGV is open (If not configured in IGV, 
   IGVload(sock, exampleSessionFile)

## End(Not run)
```

`IGVsnapshot` *Make a file snapshot of the current IGV screen.*

Description

From the IGV documentation: "Saves a snapshot of the IGV window to an image file. If filename is omitted, writes a .png file with a filename generated based on the locus. If filename is specified, the filename extension determines the image file format, which must be .png or .eps."

Usage

```
IGVsnapshot(sock, fname = "", dirname=getwd())
```

Arguments

`sock`	A socket connection to IGV.
`fname`	The filename to save. Alternatively, if not specified, IGV will create a filename based on the locus being viewed.
`dirname`	The directory name as a string for where to save the snapshot file.

Author(s)

Sean Davis <sdavis2@mail.nih.gov>

References

http://www.broadinstitute.org/igv/PortCommands

See Also

startIGV

Examples

```
   ## Not run:
   ## Create a snapshot of the current IGV window, which is usually the first launched IGV with listen port
   sock <- IGVsocket()
   IGVsnapshot(sock)
   dir()

## End(Not run)
```

IGVsocket *Create a Socket Connection to IGV.*

Description

Create a Socket Connection to IGV by a specified port and host.

Usage

```
IGVsocket(host='localhost', port=60151)
```

Arguments

`host`	The name of remote host where IGV is running.
`port`	The port to connect to/listen on.

Author(s)

Sean Davis <sdavis2@mail.nih.gov>

References

http://www.broadinstitute.org/igv/PortCommands

See Also

startIGV, IGVgoto

Examples

```
    ## Not run:
    ## Create a socket connection to IGV
    sock <- IGVsocket()

## End(Not run)
```

IGVsort — *Sort an alignment track by the specified option.*

Description

Using the remote command port of IGV, Sorts an alignment track by the specified option. Recognized values for the option parameter are: base, position, strand, quality, sample, and readGroup.

Usage

```
IGVsort(sock, option)
```

Arguments

`sock`	A socket connection to IGV.
`option`	Recognized values for the option parameter are: base, position, strand, quality, sample, and readGroup.

Author(s)

Jack Zhu<zhujack@mail.nih.gov>

References

http://www.broadinstitute.org/igv/PortCommands

See Also

`startIGV`, `IGVload`

Examples

```
    ## Not run:
    sock <- IGVsocket()
    IGVsort(sock, 'position')
    IGVsort(sock, 'base')
    IGVsort(sock, 'sample')

## End(Not run)
```

`listSRAfile`	*List sra, sra-lite or fastq data file names associated with input SRA accessions*

Description

This function lists all sra, sra-lite or fastq data files associated with input SRA accessions

Usage

```
listSRAfile( in_acc, sra_con, fileType = 'litesra', srcType = 'ftp' )
```

Arguments

`in_acc`	character vector of SRA accessions, which should be in same SRA data type, either submission, study, sample, experiment or run.
`sra_con`	connection to the SRAmetadb SQLite database
`fileType`	types of SRA data files, which should be 'sra', 'litesra' or 'fastq'.
`srcType`	type of transfer protocol, which should be "ftp" or "fasp".

Details

SRA fastq files are hosted at EBI ftp site (ftp://ftp.sra.ebi.ac.uk/vol1/fastq/) and .sra or .lite.sra files are hosted at NCBI ftp site (ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/).

Value

A data frame of matched SRA accessions and data file names with ftp or fasp addresses.

Author(s)

Jack Zhu <zhujack@mail.nih.gov>

See Also

`getSRAfile`

Examples

```
if( file.exists('SRAmetadb.sqlite') ) {
## Not run:
library(SRAdb)
sra_dbname <- 'SRAmetadb.sqlite'
sra_con <- dbConnect(dbDriver("SQLite"), sra_dbname)

## List ftp or fasp addresses of sra files associated with "SRX000122"
listSRAfile (in_acc = c("SRX000122"), sra_con = sra_con, fileType = 'sra')
listSRAfile (in_acc = c("SRX000122"), sra_con = sra_con, fileType = 'sra', srcType='fasp')

## End(Not run)
} else {
print("use getSRAdbFile() to get a copy of the SRAmetadb.sqlite file  and then rerun the example")
}
```

`sraConvert` *Cross-reference between GEO data types*

Description

A common task is to find all the SRA entities of one type associated with another SRA entity (eg., find all SRA samples associated with SRA study 'SRP001990'). This function provides a very fast mapping between entity types to facilitate queries of this type.

Usage

```
sraConvert(in_acc, out_type = c("sra", "submission", "study", "sample", "experiment", "run"), sr
```

Arguments

`in_acc`	Character vector of SRA accessions and should be of same SRA data type, either one of SRA submission, SRA study, SRA sample, SRA experiment and SRA run'
`out_type`	Character vector of the following SRA data types: 'sra', 'submission','study','sample','experiment','ru if 'sra' is in out_type, out_type will be c("submission", "study", "sample", "experiment", "run")
`sra_con`	Connection to the SRAmetadb SQLite database

Value

A data.frame containing all matched SRA accessions.

Author(s)

Jack Zhu <zhujack@mail.nih.gov>

See Also

`getSRA`, `listSRAfile`, `getSRAinfo`

Examples

```
if(file.exists('SRAmetadb.sqlite')) {

library(SRAdb)
sra_dbname <- 'SRAmetadb.sqlite'
sra_con <- dbConnect(dbDriver("SQLite"), sra_dbname)

## Convert SRA experiment accessions to other types
a <- sraConvert( in_acc=c(" SRR000137", "SRR000138 "), out_type=c('sample'), sra_con=sra_con )
b <- sraConvert (in_acc=c("SRX000089"), sra_con=sra_con)

} else {
   print("use getSRAdbFile() to get a copy of the SRAmetadb.sqlite file
and then rerun the example")
}
```

sraGraph *Create a new graphNEL object of SRA accessios from SRA full text search*

Description

This function will create a new graphNEL object from SRA accessions using function of `entityGraph` and SRA accessions are returned from SRA full text search using function of `getSRA`

Usage

```
sraGraph(search_terms, sra_con)
```

Arguments

`search_terms`	Free text search terms constructed according to SQLite query syntax defined here: http://www.sqlite.org/fts3.html#section_1_3
`sra_con`	Connection to the SRAmetadb SQLite database

Details

This function is a wrapper of two functions: acc <- getSRA(search_terms, out_types='sra', sra_con, acc_only=TRUE) and g <- entityGraph(acc). A graphNEL object with edgemode='directed' is created from input data.frame of SRA accessions and the `plot` function will draw a graph

Value

A graphNEL object with edgemode='directed'

Author(s)

Jack Zhu <zhujack@mail.nih.gov> and Sean Davis <sdavis2@mail.nih.gov>

See Also

`getSRA`, `sraConvert`, `entityGraph`

Examples

```
if(file.exists('SRAmetadb.sqlite')) {

library(SRAdb)
   library(Rgraphviz)

sra_dbname <- 'SRAmetadb.sqlite'
sra_con <- dbConnect(dbDriver("SQLite"), sra_dbname)

## create a graphNEL object from SRA accessions, which are full text search results of terms 'primary thyro
g <- sraGraph('primary thyroid cell line', sra_con)
attrs <- getDefaultAttrs(list(node=list(fillcolor='lightblue', shape='ellipse')))
plot(g, attrs=attrs)
## similiar search as the above, returned much larger data.frame and graph is too clouded
g <- sraGraph('Ewing Sarcoma', sra_con)
## Not run:
plot(g)

## End(Not run)
} else {
print("use getSRAdbFile() to get a copy of the SRAmetadb.sqlite file
and then rerun the example")
}
```

`startIGV` *Start IGV from R with different amount maximum memory support*

Description

This function is to start the Integrative Genomics Viewer (IGV) within R, which is a high-performance visualization tool for interactive exploration of large, integrated datasets. It supports a wide variety of data types including sequence alignments, microarrays, and genomic annotations. In the SRAdb, functions of load2IGV and load2newIGV can be used to load BAM format of sequencing data into IGV conveniently.

Usage

```
startIGV(memory = "mm", devel=FALSE)
```

Arguments

`memory`	Maximum usable memory support for the IGV to be launched, which is defined as the following: 'mm' - 1.2 GB , 'lm' - 2 GB, 'hm' - 10 GB, " - 750 MB
`devel`	Start development version of IGV.

Details

IGV with 1.2 GB maximum usable memory ('mm') is usually for 32-bit Windows; IGV with 2 GB maximum usable memory ('lm') is usually for 32-bit MacOS; IGV with 10 GB maximum usable memory is for large memory 64-bit java machines; IGV with 750 MB (") is sufficient for most applications. The IGV will be launched through Java Web Start. For details about how IGV is launched or have problems to launch it, please refer to this site: http://www.broadinstitute.org/igv/StartIGV . Note: if `IGVload` will be used to load BAM files to the new launched IGV, a connection port needs to be enabled in the IGV. This is how to enable connection port in the IGV: in IGV, go View->Preferences->Advanced->Enable port and check the checkbox.

Author(s)

Jack Zhu

References

http://www.broadinstitute.org/igv/

See Also

IGVload, IGVgoto, IGVgenome

Examples

```
## launch IGV with 1.2 GB maximum usable memory support
## Not run: startIGV("lm"))
```

Index