

# SRADB

March 24, 2012

---

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	<i>Collapse tracks in the IGV</i>
-------------	-----------------------------------

---

## 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**

<code>sock</code>	A socket connection to IGV.
<code>files</code>	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: <a href="http://www.broadinstitute.org/igv/ControlIGV">http://www.broadinstitute.org/igv/ControlIGV</a> .

**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](mailto: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 co
  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
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)
```



## 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:	1.5.1
Date:	2011-01-06
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)

## List sra-lite data files associated with each input SRA accessions
## Not run:
listSRAfile (in_acc=c("SRX000089"), sra_con=sra_con, sraType='litesra')

## End(Not run)
## Get file size and date from NCBI ftp site for available sra-lite data files associated with input SRA accessions
## Not run:
getSRAinfo (in_acc=c("SRS012041","SRS000290"), sra_con=sra_con, sraType='litesra')

## End(Not run)
## Download sra-lite data files associated with input SRA accessions from NCBI SRA ftp site
## Not run:
getSRAfile (in_acc=c("SRR000648","SRR000657"), sra_con=sra_con, destdir=getwd(), sraType='litesra')

## 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 SRA accessions
g <- sraGraph('colon cancer', 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")
}

```

---

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
")
}

```

---

entityGraph	<i>Create a new graphNEL object from an input entity matrix or data.frame</i>
-------------	---

---

### 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 te  
  acc <- getSRA (search_terms='colon cancer', out_types=c('sra'), sra_con=sra_con, acc_  
  d <- 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")  
}
```

```
}

```

---

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'))
```

## Arguments

`search_terms` Free text search terms constructed according to SQLite query syntax defined here: [http://www.sqlite.org/fts3.html#section\\_1\\_3](http://www.sqlite.org/fts3.html#section_1_3)

`out_types` Character vector of the following SRA data types: 'sra', 'submission', 'study', 'sample', 'experiment', 'run'. Note: if 'sra' is within out\_types, the out\_types will be set to c('submission', 'study', 'sample', 'experiment', 'run')

`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](mailto: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 file and there could be one
  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 file:
  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
  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
  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 accession
  rs <- getSRA (search_terms = 'study_title: brea* can*', out_types=c('run','study'), sra_con=sra_con)

} else {
  print("use getSRADBFile() to get a copy of the SRAMetadb.sqlite file and then rerun this script")
}

```

---

getSRADBFile	<i>Download and unzip last version of SRAMetadb.sqlite.gz from the server</i>
--------------	---

---

**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 <a href="#">download.file</a> .

**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	<i>Download sra or sra-lite data files associated with input SRA accessions from NCBI SRA ftp site</i>
------------	--

---

**Description**

This function will available download sra or sra-lite data files associated with input SRA accessions from NCBI SRA ftp site.

**Usage**

```
getSRAfile(in_acc, sra_con, destdir, sraType='litesra', makeDirectory = FALSE, m
```

**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
sra_con	Connection to the SRAMetadb SQLite database
destdir	Directory where downloaded fastq files are saved
sraType	types of SRA data files, which should be "sra" or "litesra".
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 <a href="#">download.file</a> .

**Details**

The function will convert all input SRA accessions to SRA experiment and run accessions and then all available sra or sra-lite data files will be downloaded. Currently files are only downloaded from the NCBI SRA ftp site.

**Warning**

The size of sra or sra-lite data files in SRA could be very large and downloading multiple ones could take long time and quite some storage space

**Author(s)**

Jack Zhu <zhujack@mail.nih.gov>

**See Also**

[getSRAfile](#), [listSRAfile](#)

**Examples**

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

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

## Download all available fastq files associated with "SRR000648" and "SRR000657" from NC
getSRAfile (in_acc=c("SRR000648","SRR000657"), sra_con=sra_con, destdir=getwd(), sraType=

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

---

getSRAinfo

*Get file size and date from NCBI ftp site for available sra or sra-lite data files associated with input SRA accessions*

---

**Description**

This function will get sra or sra-lite data file size and date from NCBI ftp site associated with input SRA accessions

**Usage**

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

**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
sra_con	Connection to the SRAMetadb SQLite database
sraType	types of SRA data files, which should be 'sra' or 'litesra'.

**Details**

The function will convert all input SRA accessions to SRA experiment and run accessions and get sra or sra-lite data file size and date from NCBI SRA ftp sites.

**Value**

A data.frame containing NCBI ftp addresses to the SRA data files, and 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 "S
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 t
}
```

---

listSRAfile	<i>List sra/sra-lite data file names associated with input SRA accessions</i>
-------------	---

---

**Description**

This function will list all sra/sra-lite files associated with input SRA accessions

**Usage**

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

**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
sra_con	Connection to the SRAmetadb SQLite database
sraType	types of SRA data files, which should be 'sra' or 'litesra'.

**Details**

The function will convert all input SRA accessions to SRA experiment and run accessions and then construct sra or sra-lite data file ftp addresses from converted experiment and run accessions. No ftp checking or downloading. Currently only NCBI SRA ftp addresses are created.

**Value**

List NCBI ftp links to SRA data files along with input SRA accessions.

**Author(s)**

Jack Zhu <zhujack@mail.nih.gov>

**See Also**

[getSRAfile](#)

**Examples**

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

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

  ## List fastq file ftp addresses associated with "SRA000045"
  listSRAfile (in_acc=c("SRX000122"), sra_con=sra_con, sraType='sra')

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

---

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"))
```

**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', 'run'. 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_
  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](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)  
  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 te  
  g <- sraGraph('colon cancer', 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:  
  library(Rgraphviz)  
  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

## \*Topic **IO**

- IGVclear, 1
- IGVcollapse, 2
- IGVgenome, 3
- IGVgoto, 3
- IGVload, 4
- IGVsnapshot, 6
- IGVsocket, 7
- IGVsort, 8

## \*Topic **NCBI**

- colDescriptions, 11

## \*Topic **SRA**

- colDescriptions, 11
- sraConvert, 18

## \*Topic **database**

- colDescriptions, 11
- sraConvert, 18

## \*Topic **package**

- SRAdb-package, 9

colDescriptions, 11

download.file, 14, 15

entityGraph, 12, 19, 20

getSRA, 12, 13, 19, 20

getSRAdbFile, 14

getSRAfile, 15, 16–18

getSRAinfo, 16, 19

IGVclear, 1

IGVcollapse, 2

IGVgenome, 3, 5, 21

IGVgoto, 1, 3, 4, 5, 7, 21

IGVload, 1, 2, 4, 4, 5, 8, 21

IGVsession, 5

IGVsnapshot, 6

IGVsocket, 7

IGVsort, 8

listSRAfile, 16, 17, 17, 19

plot, 12, 19

sraConvert, 12, 13, 18, 20

SRAdb (*SRAdb-package*), 9

SRAdb-package, 9

sraGraph, 12, 13, 19

startIGV, 1–4, 6–8, 20