DAVIDQuery

April 19, 2010

`AffyProbesetList` *Retrieve Affy probeset IDs from DAVID.*

Description

For a given Affymetrix microarray chip, retrieve Affy probeset IDs from DAVID. Optionally, a menu is used to pick the chip name.

Usage

```
AffyProbesetList(chipname = NULL, menu = TRUE, verbose=FALSE)
```

Arguments

`chipname`	Full name or regular expression.
`menu`	Select chipname from a menu (default=TRUE)
`verbose`	Print a bit of tracing information along the way (default=FALSE) .

Details

First, DAVID's table of chip names is retrieved. When the user selects or specifies one of the names, the associated file of probeset names is retrieved. If `chipname` is a regular expression, then the menu (if requested) is subsetted accordingly.

Value

Character vector of probeset names.

Note

Use with caution. The returned file is not guaranteed to be correct. In the example above, with the chip `"Human Genome U133 Plus 2.0"`, the list returned includes 40907 probeset IDs on the chip (and no others), but appears to be missing 13768 others.

Author(s)

Roger Day

Examples

```
head(AffyProbesetList("Human Genome U133 Plus 2.0", menu=FALSE, verbose=TRUE))
## Not run:
length(AffyProbesetList("133|95"))

## End(Not run)
```

`affyToUniprot` *Obtain Affymetrix probeset IDs for given Uniprot IDs.*

Description

Obtain Affymetrix probeset IDs for given Uniprot IDs, using DAVIDQuery.

Usage

```
affyToUniprot(ids = "88736_AT", ...)
```

Arguments

`ids`	Affymetrix probeset IDs.
`...`	Args to be passed to `DAVIDQuery()`.

Value

The output of DAVIDQuery. If only the DAVIDQueryResult component is desired, include the arg details=FALSE.

Note

There is currently no provision for using `DAVIDQueryLoop` .

Author(s)

Roger Day

See Also

`DAVIDQuery`

`bracketedStrings` *Extract bracketed substrings.*

Description

Extract substrings that are bracketed by specified strings before and after.

Usage

```
bracketedStrings(s, before, after, verbose=FALSE, addNames=FALSE, drop.na=TRUE,
```

Arguments

`s`	Vector of strings to search.
`before`	String to the left of the desired substring within s.
`after`	String to the right of the desired substring within s.
`verbose`	If TRUE, print the starting and ending index (or indices) of the desired substring(s).
`addNames`	If TRUE, and if `s` is a vector, set the `names` attribute of the return value to `s`.
`drop.na`	If TRUE, remove empty strings from the return value.
`warn.if.gt.1`	If TRUE, warn if a string has more than one pair of bracketed target strings.

Value

For a single input string `s`, the return value is the desired substring sandwiched between `before` and `after`. For a vector of inputs, list of outputs.

Author(s)

Roger Day

See Also

`DAVIDQuery`

Examples

```
bracketedStrings("quickbrownfox", "quick", "fox")
bracketedStrings(c("quickbrownfox", "quickredfox"), "quick", "fox", addNames=TRUE)
bracketedStrings("quickbrownfoxANDquickredfox", "quick", "fox")
bracketedStrings("quickbrownfoxANDquickredfox", "quick", "fox", warn.if.gt.1=FALSE)
```

`catalogDAVIDResultsByTool` *Create a catalog of types of DAVID results.*

Description

Loops through values of `tool`, for the specified value of `annot`. Runs each DAVID query, and saves the result to create a catalog of types of DAVID results.

Usage

```
catalogDAVIDResultsByTool(annot = NULL, sleepSeconds = 10, ...)
```

Arguments

`annot`	See `DAVIDQuery`.
`sleepSeconds`	`DAVIDQueryLoop`.
`...`	Extra args passed to DAVIDQuery.

Details

The purpose is to check comprehensively whether there are results that could be better formatted than the default output or the reformatting provided by `formatDAVIDResult`.

Value

A list of outputs from DAVIDQuery. Automatically assigned to the name `catalogOfDAVIDResultsByTool.ANNO` where ANNOT is replaced by the `annot` argument.

Author(s)

Roger Day

`DAVIDQueryLoop` *Access DAVID multiple times.*

Description

Make a query larger than DAVID allows in one go, by looping, respecting the limitations imposed by DAVID policies.

Usage

```
DAVIDQueryLoop(
idList = unlist(strsplit(strsplit("P31946 P62258 P29360 P42655 Q63631\nP01892 O1
" ")[[1]], "\n")),
idLimit = 100,
sleepSeconds = 10,
hitsPerDayLimit = 200,
verbose = FALSE,
testMe = FALSE,
type,
annot,
tool,
graphicMenu = FALSE,
formatEach = FALSE,
formatAll = FALSE,
...)
```

Arguments

idList	IDs of interest for query.
idLimit	Published limit of number of ID's to process in one call.
sleepSeconds	Published minimum time between iterations
hitsPerDayLimit	Published maximum URL calls to the API per day from one address.
verbose	Print out tracking information as the queries are sent.
testMe	Runs DAVIDQueryLoop with arguments set as follows: annot=NULL, tool="geneReportFull", type="UNIPROT_ACCESSION", verbose=TRUE
type	See DAVIDQuery.
annot	See DAVIDQuery.
tool	See DAVIDQuery.
graphicMenu	See DAVIDQuery.
formatEach	Passed to DAVIDQuery as the formatIt argument.
formatAll	Assembled results are sent to formatDAVIDResult.
...	Other args to be passed to DAVIDQuery.

Value

The results of DAVIDQuery bound together with rbind. Not printed (returned invisibly).

Note

For some choice of the tool argument, the result returned may differ if idLimit is changed.

Author(s)

Roger Day

See Also

DAVIDQuery

`DAVIDQuery-package` *Retrieval from DAVID bioinformatics data resource.*

Description

Tools to retrieve data from DAVID, the Database for Annotation, Visualization and Integrated Discovery.

Details

Package:	DAVIDQuery
Type:	Package
Version:	1.0
Date:	2008-10-31
License:	GPL-2
LazyLoad:	yes
Vignette and overview:	vignette("DAVIDQuery")

Author(s)

Roger Day <day@upci.pitt.edu>

References

Home base for DAVID (in this package as DAVIDURLBase): http://david.abcc.ncifcrf.gov

Huang et al, DAVID gene ID conversion: http://www.pubmedcentral.nih.gov/articlerender.fcgi?tool=pubmed&pubmedid=18841237

`DAVIDQuery` *DAVIDQuery*

Description

Launch a query against DAVID, the Database for Annotation, Visualization and Integrated Discovery. Return the results into an R object.

Usage

```
DAVIDQuery(ids = "O00161,O75396", type = "UNIPROT_ACCESSION", annot, tool, URLle
details = TRUE, verbose = FALSE, writeHTML = FALSE, testMe = FALSE, graphicMenu
```

Arguments

ids	IDs for desired objects, as a character vector or as a single string with ids separated by ",". To be passed to the DAVID website, the format has to be the latter.
type	Type of input ids. If missing, a menu is contructed from the R object DAVIDTypeChoices. See http://david.abcc.ncifcrf.gov/content.jsp?file=DAVID_API.html#input_list for up-to-date choices.
annot	Type of annotation requested. If missing, a menu is contructed from the R object DAVIDAnnotChoices. See http://david.abcc.ncifcrf.gov/content.jsp?file=DAVID_API.html#input_list for up-to-date choices.
tool	Type of gene tool to use. If missing, a menu is contructed from the R object DAVIDToolChoices. See http://david.abcc.ncifcrf.gov/content.jsp?file=DAVID_API.html#input_list for up-to-date choices. As of this writing, the tool choices corresponding to the Functional Annotation tools cannot be handled by this package.
URLlengthLimit	Published maximum length of the constructed URL.
details	If TRUE (default), a list of intermediate results is returned; otherwise, just the final query result.
verbose	If TRUE (default is FALSE), more debugging information is printed.
writeHTML	If TRUE (default is FALSE), write the received intermediate HTML to files.
testMe	If TRUE (default is FALSE), assign default valuse and run.
graphicMenu	If TRUE (default is FALSE), use a GUI window for the pick menus.
formatIt	If TRUE (default), try to interpret the returned character table and structure the result. If false, the unadorned character table returned by DAVID.

Details

The API described at http://david.abcc.ncifcrf.gov/content.jsp?file=DAVID_API.html is used. The return is screen-scraped, a new URL is formulated and transmitted, again the return is screen-scraped to find the name of the results file, and finally that file is retrieved into a string matrix.

Obviously this approach is brittle, but it has survived the 2008 DAVID update. A real API would be better, of course.

The return value DAVIDQueryResult is just a character matrix. Its content structure depends on the choices of tool and annotation arguments, so there has been no attempt to manipulate it into, say, a data frame with nice column names.

Value

If detail==FALSE, only DAVIDQueryResult is returned. This a character matrix holding the results of the tab-delimited file returned by DAVID.

If detail==TRUE, a list with contents useful for trouble-shooting:

firstURL
firstStageResult

DAVIDaction

```
secondURL
secondStageResult

hasSessionEnded

downloadFileName


downloadURL
DAVIDQueryResult
```

Author(s)

Roger Day

References

http://david.abcc.ncifcrf.gov http://david.abcc.ncifcrf.gov/content.jsp?file=DAVID_API.html Article: DAVID gene ID conversion tool. Da Wei Huang, Brad T Sherman, Robert Stephens, Michael W Baseler, H Clifford Lane, and Richard A Lempicki http://www.pubmedcentral.nih.gov/articlerender.fcgi?tool=pubmed&pubmedid=18841237

See Also

DAVIDQueryLoop, formatDAVIDResult

Examples

```
result = DAVIDQuery(testMe=TRUE)$DAVIDQueryResult
print(length(result))
print(result$firstURL)
print(result$secondURL)
print(result$O00161$REFSEQ_MRNA)  ### Uses UNIPROT ID's for input.
```

DAVIDToolChoices *Choices for the DAVID query parameters*

Description

DAVIDToolChoices, DAVIDTypeChoices, and DAVIDAnnotChoices are string vectors used to construct pick menus, when the corresponding arguments to DAVIDQuery are not provided.

Details

The source of these lists can be found at http://david.abcc.ncifcrf.gov/content.jsp?file=DAVID_API.html#input_list The lists posted there may change in the future. No mechanism is included in this package to retrieve and parse the page.

Source

http://david.abcc.ncifcrf.gov/content.jsp?file=DAVID_API.html#input_list

See Also

DAVIDQuery

`DAVIDURLBase` *Base URL for the DAVID database.*

Description

Base URL for the DAVID database.

Source

http://david.abcc.ncifcrf.gov

See Also

DAVIDQuery DAVIDToolChoices

`formatDAVIDResult` *Format the character table returned by DAVID.*

Description

These functions attempt to format the character table `result` returned by DAVID.

Usage

```
formatDAVIDResult(result, verbose=FALSE)
formatAnnotationReport(result)
formatGeneReport(result)
formatGeneReportFull(result)
formatList(result)
formatGene2Gene(result)
```

Arguments

`result`	Character table returned by DAVID.
`verbose`	If TRUE, print tool. Warn if tool=="geneReportFull" that the result will be returned invisibly due to its size.

Details

formatDAVIDResult switches out to one of formatGeneReport, formatGeneReportFull, formatGene2Gene, or formatList, depending on the tool argument of DAVIDQuery() used to specify what query report to do. The tool argument is passed as an attribute attached to result.

WARNINGS: Not all values of tool have an associated format.

These format utilities are not guaranteed to work correctly for all combinations of inputs into DAVIDQuery(), or to continue to work correctly if or when the DAVID API changes. If results appear incorrect, one can use the option DAVIDQuery(formatIt=FALSE) to see the unformatted output, and/or paste DAVIDQuery(details=TRUE)[firstURL] into a browser.

In the case of formatGene2Gene, the gene column of the details component might not always contain a single identifier.

Value

For tool=="geneAnnotationReport", a list, one component for each element in the ids arg. Each component has subcomponents

Gene Name	Self-explanatory.
Species	Self-explanatory.
<id>	The identifier(s) in the query. The name is whatever the id type was.
<other items>	Items produced for the input, specified by the annot arg of the query.

For tool=="geneReport" or tool=="list", a character matrix with column names scraped from DAVIDQueryResult, usually:

<gene name>	Using the same ID type as the type argument of DAVIDQuery().
Gene Name	Self-explanatory.
Species	Self-explanatory.

In addition, for tool=="geneReport", the first line of the returned output is saved as an attribute before discarding it.

For tool=="geneReportFull", a list, one component for each element in the ids arg. Each component has subcomponents

Gene Name	Self-explanatory.
Species	Self-explanatory.
<other items>	The union of items produced for the input identifiers (generically called "genes" in DAVID). (The set of attributes is not fixed.)

For tool=="formatGene2Gene", a list with one component for each Functional Group. Each component has components

median	See DAVID documentation.
geo	See DAVID documentation.
diagram	An attempt to parse the fourth column of the Functional Group line of the input. See DAVID documentation and consult the DAVID team.
details	A data frame with columns

gene "gene" as identified in the `ids` arg to `DAVIDQuery`
geneName gene name
url The URL for the Gene Report page at NIAID.

As of this writing, the tool choices corresponding to most Functional Annotation tools cannot be handled by this package.

`idExampleList` *idExampleList*

Description

List of ids used in DAVID examples on page http://david.abcc.ncifcrf.gov/content.jsp?file=DAVID_API.html

See Also

`testGene2Gene`

`testGene2Gene` *Test DAVID gene2gene tool*

Description

This function tests and demonstrates the use of DAVIDQuery to access the gene2gene tool.

Usage

```
testGene2Gene(ids = "33246_AT,32469_AT,1786_AT,32680_AT,1355_G_AT,37968_AT,33530
```

Arguments

`ids`	Arg passed to `DAVIDQuery`.
`type`	Arg passed to `DAVIDQuery`.
`...`	Other args passed to `DAVIDQuery`.

Details

Input Affy IDS are taken from the example on the DAVID web site.

Value

The value returned by DAVIDQuery using tool=gene2gene.

Author(s)

Roger Day

See Also

DAVIDQuery

Examples

```
testGene2Gene(details=FALSE)
### Run example from http://david.abcc.ncifcrf.gov/gene2gene.jsp
testGene2Gene(ids=idExampleList, type="ENTREZ_GENE_ID", details=FALSE)
### Run example from http://david.abcc.ncifcrf.gov/content.jsp?file=DAVID_API.html
```

---

uniprotToAffy *Obtain Affymetrix probeset IDs for given Uniprot IDs.*

Description

Obtain Affymetrix probeset IDs for given Uniprot IDs, using DAVIDQuery.

Usage

```
uniprotToAffy(uid = "O00161", ...)
```

Arguments

uid	Uniprot IDs, either a string with the IDs separated by commas, or else a character vector.
...	Args to be passed to DAVIDQuery().

Value

The output of DAVIDQuery. If only the DAVIDQueryResult component is desired, include the arg details=FALSE. If probesets from a specific chip are desired, then you can intersect these results with the results of AffyProbesetList.

Note

There is currently no provision for using DAVIDQueryLoop .

Author(s)

Roger Day

See Also

DAVIDQuery

Index