Package 'rPlant'

October 21, 2015

Version 2.12

Date 2015-09-18

Title Interface to the Agave API

Author Barb Banbury <darwinthesun@gmail.com> and Kurt Michels <kamichels@math.arizona.edu>, Jeremy M. Beaulieu <jeremy.be

Maintainer Kurt Michels <kamichels@math.arizona.edu>

Depends R (>= 2.13.0), rjson, RCurl, seqinr

LazyData yes

Repository/R-Forge/Project rPlant

BugReports https://r-forge.r-project.org/tracker/?group_id=1328

Description

Provides an interface to the the many computational resources iPlant offers through their RESTful application programming interface, see <http://www.iplantcollaborative.org/> for more information about iPlant, and <http://www.iplantcollaborative.org/ci/iplant-science-apis> for its APIs. Currently, 'rPlant' functions interact with the iPlant Agave API, the Taxonomic Name Resolution Service API, and the Phylotastic Taxosaurus API. Before using 'rPlant', users will have to register with the iPlant Collaborative. See <http://agaveapi.co/> for more information on the Agave API and <http://user.iplantcollaborative.org/> to register with iPlant.

URL http://www.iplantcollaborative.org/discover/discovery-environment

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2015-10-21 10:02:03

R topics documented:

`rplant-package` *rplant*

Description

R interface to iPlant's Discovery Enviornment and the Taxonomic Name Resolution Service (TNRS)

Details

The iPlant Collaborative has developed many resources to deal with the emerging computational challenges facing biology. Users have access to many different applications for data analysis, including clustering/network analyses, QTL mapping, sequence alignments, phylogenetic tree building, and comparative methods. The 'rPlant' package provides a direct link between iPlant's API and the R environment. 'rPlant' users must be registered with iPlant and have a valid user name and password. If you do not have an account yet, visit `http://user.iplantcollaborative.org/` and register.

Package:	rplant
Type:	Package
Version:	3.0.0
Date:	2014-6-5
License:	GPL (>= 2)

See Also

`Validate`, `UploadFile`, `Muscle`

Examples

```
## Not run: data(PROTEIN.fasta)
## Not run: write.fasta(sequences = PROTEIN.fasta, names = names(PROTEIN.fasta),
```

```
                    file.out = "PROTEIN.fasta")
## End(Not run)
## Not run: Validate("username","password")
## Not run: UploadFile("PROTEIN.fasta", filetype="FASTA-0")
## Not run: SubmitJob(application="Muscle-3.8.32u4", file.list=list("PROTEIN.fasta"),
                  input.list=list("stdin"), args.list=list(c("arguments", "-phyiout")),
                  job.name="MuscleSJAA")
## End(Not run)

### Or, one can use one of the wrapper functions. ###
## Not run: Muscle("PROTEIN.fasta", aln.filetype="PHYLIP_INT", job.name="muscleAA")
```

`ClustalW` *ClustalW alignment*

Description

An approach for performing multiple alignments of large numbers of amino acid or nucleotide sequences is described. The method is based on first deriving a phylogenetic tree from a matrix of all pairwise sequence similarity scores, obtained using a fast pairwise alignment algorithm. See details on http://www.clustal.org/clustal2/.

Usage

```
ClustalW(file.name, file.path="", type="DNA", aln.filetype="CLUSTALW",
         args=NULL, out.name=NULL, job.name=NULL, print.curl=FALSE,
         shared.username=NULL, suppress.Warnings=FALSE, email=TRUE)
```

Arguments

`file.name`	Name of file to be evaluated on the Discovery Environment (DE), see details for supported input formats.
`file.path`	Optional path to a user's subdirectory on the DE; default path is empty, which leads to the home directory
`type`	Two options "PROTEIN" or "DNA". This defines the type of sequences in the file
`aln.filetype`	ClustalW does alignment of sequences, this option selects the file type of that result file. There are seven options `CLUSTALW`, `FASTA`, `PHYLIP_INT`, `NEXUS`, `GCG`, `GDE`, and `PIR`
`args`	Optional for arguments (i.e. flags). The ClustalW model has much additional functionality that is not fit into this wrapper function (http://www.clustal.org/download/clustalw_help.txt), see details. This option allows users to add anything that is not included (.i.e. args="-ITERATION=TREE"), to iterate at each step, see details.
`out.name`	The name given to the output filename
`job.name`	The name to give the job being submitted

`print.curl` Prints the curl statement that can be used in the terminal, if curl is installed on your computer

`shared.username` With iPlant you have the ability to share folders with other users. If someone has shared a folder with you and you want to run a job with them, enter their username for this input.

`suppress.Warnings` This will turn off the warnings, will speed up run time. Use with caution, if the inputs are incorrect they will not be caught.

`email` By default an email will be sent to the user when the job finishes.

Details

The supported input file format is the fasta format http://en.wikipedia.org/wiki/FASTA_format.

Additional arguments, `args`, can be found at http://www.clustal.org/download/clustalw_help.txt. The args input is text with the flags and inputs for those flags in a string like on the command line.

There are seven options for output files: `CLUSTALW` http://meme.nbcr.net/meme/doc/clustalw-format.html, `FASTA` http://www.bioperl.org/wiki/PHYLIP_multiple_alignment_format, `PHYLIP_INT` http://www.bioperl.org/wiki/PHYLIP_multiple_alignment_format, `NEXUS` http://en.wikipedia.org/wiki/Nexus_file, `GCG` http://www.genomatix.de/online_help/help/sequence_formats.html#GCG, `GDE` http://www.cse.unsw.edu.au/~binftools/birch/GDE/overview/GDE.file_formats.html, and `PIR` http://www.bioinformatics.nl/tools/crab_pir.html.

The result file is ALWAYS 'clustalw2.fa'.

Value

A list containing the job id and the job name is provided for jobs submitted. If an error, then a message stating the error should also be reported.

See Also

`SubmitJob`, `Validate`, `UploadFile`

Examples

```
## Not run: data(DNA.fasta)
## Not run: write.fasta(sequences = DNA.fasta, names = names(DNA.fasta), file.out = "DNA.fasta")
## Not run: Validate("username","password")
## Not run: UploadFile("DNA.fasta", filetype="FASTA-0")
## Not run: ClustalW("DNA.fasta", job.name="ClustalWPHY", aln.filetype="PHYLIP_INT")
```

`Data Sets` *Data Sets*

Description

The *.fasta data sets are sample data sets with only 10 sequences. One is protein and one is DNA. The .tfam/.tped files are transformed PLINK format. The PLINK format is used mostly for Genome Wide Association Studies.

Format

fasta

`Fasttree` *FastTree Dispatcher*

Description

FastTree infers approximately-maximum-likelihood phylogenetic trees from alignments of nucleotide or protein sequences. See http://meta.microbesonline.org/fasttree/

Usage

```
Fasttree(file.name, file.path="", job.name=NULL, out.name=NULL, args=NULL,
         type="DNA", model=NULL, gamma=FALSE, stat=FALSE, print.curl=FALSE,
         shared.username=NULL, email=TRUE, suppress.Warnings=FALSE)
```

Arguments

`file.name`	name of file to be evaluated on the Discovery Environment (DE), see details for supported input formats.
`file.path`	optional path to a user's subdirectory on the DE; default path is empty, which leads to the home directory
`out.name`	the name given to the output tree (default FastTree_OutTree.nwk)
`type`	Two options "PROTEIN" or "DNA". This defines the type of sequences in the file, either proteins or nucleotides.
`model`	Substitution model. For DNA the choices are `GTRCAT`, and `JCCAT` (default), the Jukes-Cantor + CAT model. For protein the choices are `JTTCAT` (default), and `WAGCAT`.
`gamma`	(-gamma) Use this option (about 5% slower) if you want to rescale the branch lengths and compute a Gamma20-based likelihood. Gamma likelihoods are more comparable across runs (default is FALSE).
`stat`	(-log) This allows for statistical comparisons (when gamma=TRUE) of the likelihood of different topologies. The result file is 'logfile'.

job.name	the name to give the job being submitted
print.curl	Prints the curl statement that can be used in the terminal, if curl is installed on your computer
shared.username	With iPlant you have the ability to share folders with other users. If someone has shared a folder with you and you want to run a job with them, enter their username for this input.
suppress.Warnings	This will turn off the warnings, will speed up run time. Use with caution, if the inputs are incorrect they will not be caught.
email	By default an email will be sent to the user when the job finishes.
args	Optional for arguments (i.e. flags). The Fasttree model has much additional functionality that is not fit into this wrapper function (http://meta.microbesonline.org/fasttree/#Usage), see details. This option allows users to add anything that is not included, see details.

Details

The input file formats that are supported are fasta format http://en.wikipedia.org/wiki/FASTA_format or interleaved phylip format http://www.bioperl.org/wiki/PHYLIP_multiple_alignment_format.

Additional arguments, args, can be found at http://meta.microbesonline.org/fasttree/. The args input is text with the flags and inputs for those flags in a string like on the command line.

Fasttree outputs trees in Newick format http://en.wikipedia.org/wiki/Newick_format. The placement of the root is not biologically meaningful. The local support values are given as names for the internal nodes, and range from 0 to 1, not from 0 to 100 or 0 to 1,000. If all sequences are unique, then the tree will be fully resolved (the root will have three children and other internal nodes will have two children). If there are multiple sequences that are identical to each other, then there will be a multifurcation. Also, there are no support values for the parent nodes of redundant sequences.

Value

A list containing the job id and the job name is provided for jobs submitted. If an error, then a message stating the error should also be reported.

See Also

SubmitJob, Validate, UploadFile

Examples

```
## Not run: data(fasta_aa.aln)
## Not run: write.table(fasta_aa.aln, "fasta_aa.aln", quote=FALSE, row.names=FALSE, col.names=FALSE)
## Not run: Validate("username","password")
## Not run: UploadFile("fasta_aa.aln")
## Not run: myJobFaP <- Fasttree("phylip_interleaved_aa.aln", job.name="fasttreeAAphy")
```

FaST_LMM *FaST-LMM-2.07*

Description

FaST-LMM (Factored Spectrally Transformed Linear Mixed Models) is a program for performing genome-wide association studies (GWAS) on large data sets. FaST-LMM is described more fully at http://www.nature.com/nmeth/journal/v8/n10/abs/nmeth.1681.html, and also at http://fastlmm.codeplex.com/

Usage

```
FaST_LMM(input.file.list="", ALL.file.path="", print.curl=FALSE,
         sim.file.list=NULL, pheno.file.name=NULL, mpheno=1,
         args=NULL, covar.file.name=NULL, job.name=NULL,
         shared.username=NULL, suppress.Warnings=FALSE,
         out.basename=NULL, email=TRUE)
```

Arguments

`input.file.list`
: (required) Names of files, in a list format, to be evaluated on the iPlant servers. There are only three possible input groups for file.list, regular fileset (.map/.ped), transposed fileset (.tfam/.tped), and binary fileset (.bed/.bim/.fam).

`ALL.file.path`
: Optional path to a user's subdirectory on the iPlant servers; default path is empty, which leads to the home directory. All files specified including optional files must be in this directory.

`job.name`
: The name to give the job being submitted

`out.basename`
: The base name for the output files (not including extension).

`sim.file.list`
: (optional) A list containing the names of genetic similarity files. These are used to determine the genetic similarities between individuals. The may be different from the input.file.list but MUST have the same formats as the input.file.list.

`pheno.file.name`
: (optional) The full file name of the PLINK alternative phenotype file. This includes at least three columns, family ID, individual ID, and phenotype value, and may have any number of phenotype columns. Missing value as default is -9 but it can be changed.

`mpheno`
: (optional) This value MUST be entered if a phenotype file is provided. This value is the index for phenotype in -pheno file to process, starting at 1 for the first phenotype column. Default is 1

`covar.file.name`
: (optional) Must have at least three tab-delimited columns: family ID, individual ID, and covariate value, and may have any number of covariate values. The same missing value signifier from the phenotype file must be used. Note: the file should not have a header row.

`args` Optional for arguments (i.e. flags). The FaST-LMM model has much additional functionality that is not fit into this wrapper function (http://fastlmm.codeplex.com/), see manual. This option allows users to add anything that is not included (.i.e. args="-Ftest"), to automatically invoke the F-test, see details.

`print.curl` Prints the curl statement that can be used in the terminal, if curl is installed on your computer

`shared.username` With iPlant you have the ability to share folders with other users. If someone has shared a folder with you and you want to run a job with them, enter their username for this input.

`suppress.Warnings` This will turn off the warnings, will speed up run time. Use with caution, if the inputs are incorrect they will not be caught.

`email` By default an email will be sent to the user when the job finishes.

Details

The inputs for `input.file.list` are to be used only in three very strict groups. Group 1: TFAM, TPED, these are the PLINK transposed filesets. Group 2: MAP, PED, these are the PLINK regular filesets. Group 3: BED, BIM, FAM, these are the PLINK binary filesets.

Additional arguments, `args`, can be found at http://fastlmm.codeplex.com/, see the manual. The args input is text with the flags and inputs for those flags in a string like on the command line.

Not all information on the FaST-LMM model is here, see the FaST-LMM website http://fastlmm.codeplex.com/, or the FaST-LMM manual for more information.

Value

A list containing the job id and the job name is provided for jobs submitted. If an error, then a message stating the error should also be reported.

See Also

SubmitJob, Validate, UploadFile

Examples

```
## Not run: data(geno_test.tfam)
## Not run: data(geno_test.tped)
## Not run: write.table(geno_test.tfam, file = "geno_test.tfam", row.names=FALSE,
                    col.names=FALSE, quote=FALSE, sep="\t")
## End(Not run)
## Not run: write.table(geno_test.tped, file = "geno_test.tped", row.names=FALSE,
                    col.names=FALSE, quote=FALSE, sep="\t")
## End(Not run)
## Not run: Validate("username","password")
## Not run: UploadFile("geno_test.tfam")
## Not run: UploadFile("geno_test.tped")
## Not run: FaST_LMM(input.file.list=list("geno_test.tfam","geno_test.tped"))
```

ListApps *Deployed applications*

Description

Functions for listing applications deployed in the iPlant software infrastructure and information about them.

Usage

```
ListApps(description=FALSE, print.curl=FALSE, suppress.Warnings = FALSE)
GetAppInfo(application, return.json=FALSE, print.curl=FALSE)
```

Arguments

application	name of DE application
description	when the description=TRUE, then the ListApps function will include a brief description of the app, default = FALSE
return.json	optional screen output that displays all of the results from the api, default = FALSE
print.curl	Prints the curl statement that can be used in the terminal, if curl is installed on your computer
suppress.Warnings	This will turn off the warnings, will speed up run time. Use with caution, if the inputs are incorrect they will not be caught.

Value

The function ListApps returns a list of sorted applications and a short description. The applications are all verified to be public applications and they are the newest version. All of the public applications in that list can be used in the SubmitJob function. The GetAppInfo function gives critical information on the application that is needed in the SubmitJob function. A list of information is outputted. The first element gives a short description of the application. The second element in the list gives basic information on the application including is it a public application and if it is the newest version. Both are important information. If the application is a private application it can only be run by the person who submitted the application to the Agave API, and clearly you want to run the newest version of the public application.

The third element in the list the matrix outputted gives four columns of information. The first column, labeled 'kind, tells the "input", sometimes the "output" and "parameters" from the application. The second column, labeled id, give the name of the "input", etc. For example, GetAppInfo("velveth-1.2.07u1")$Information, the 'kind' column states there are six inputs for this app, and the 'id' column the names of those inputs are "reads5","reads3", etc. There are also eight parameters for the app, the paramters are format2, etc. The third column in the matrix is 'fileType/value. For the input this tells the file type which is important because if the wrong fileType is inputted into the function, then the function will not work. For the parameters the third column contains the type of input necessary for the parameters, common ones are string, boolean, etc. The last column gives brief details on each input.

See Also

SubmitJob

Examples

```
## Not run: ListApps()
## Not run: GetAppInfo("Muscle-3.8.32u4")
## Not run: GetAppInfo("velveth-1.2.07u1")
```

ListDir *Directory management*

Description

Functions for listing the contents of a directory, making new directories, or deleting entire directories in the iPlant infrastructure

Usage

```
ListDir(dir.name, dir.path="", print.curl=FALSE, shared.username=NULL,
        suppress.Warnings=FALSE, show.hidden=FALSE)
ShareDir(dir.name, dir.path="", shared.username, read=TRUE, execute=TRUE,
         write=TRUE, print.curl=FALSE, suppress.Warnings=FALSE)
PermissionsDir(dir.name, dir.path="", print.curl=FALSE, suppress.Warnings=FALSE)
RenameDir(dir.name, new.dir.name, dir.path="", print.curl=FALSE, suppress.Warnings=FALSE)
CopyDir(dir.name, dir.path="", end.path="", print.curl=FALSE, suppress.Warnings=FALSE)
MoveDir(dir.name, dir.path="", end.path="", print.curl=FALSE, suppress.Warnings=FALSE)
MakeDir(dir.name, dir.path="", print.curl=FALSE, suppress.Warnings=FALSE)
DeleteDir(dir.name, dir.path="", print.curl=FALSE, suppress.Warnings=FALSE)
```

Arguments

dir.path	optional path to a user's sub directory on the iPlant servers; default path is empty, which leads to the home directory
dir.name	name of subdirectory to be modified on the iPlant server in the dir.path
new.dir.name	the new name of the directory on the iPlant server to be renamed
end.path	Path to destination sub directory where directory on the iPlant server is moved to; default path is empty which leads to a users home directory
print.curl	Prints the curl statement that can be used in the terminal, if curl is installed on your computer
shared.username	With iPlant you have the ability to share files with other users, their username is the value for the shared.username, see details.
suppress.Warnings	This will turn off the warnings, will speed up run time. Use with caution, if the inputs are incorrect they will not be caught.

read	Gives read permissions for shared directory
write	Gives write permissions for shared directory
execute	Gives execute permissions for shared directory
show.hidden	Option to display hidden files (ie, ".svn", ".Rdata")

Details

The ListDir function lists the files in the given directory, dir.path. If listing shared folders then the dir.path becomes the path to the SHARED users shared folder. iPlant offers sharing of files between two (or more) iPlant users, the ShareDir makes it so. Once sharing is done a user can use PermissionsDir on any directory to find who the folder is shared with. The RenameDir function renames the dir.name in dir.path (on the iPlant servers), to the new.dir.name. The CopyDir function moves a dir.name in the dir.path to the end.path. The MoveDir function moves a dir.name in the dir.path to the end.path .The MakeDir function makes a directory dir.name in the directory dir.path. The DeleteDir function deletes the directory dir.name in the directory dir.path.

Value

If an error, then a message stating the error should also be reported.

See Also

UploadFile

Examples

```
# Makes the subdirectory named "new" in the users 'rPlant' directory
## Not run: MakeDir(dir.name="new", dir.path="data")

# Shares the subdirectory named "new" with the iPlant user "dude"
## Not run: ShareDir(dir.name="new", dir.path="data", shared.username="dude")

# Checks permissions folder "new"
## Not run: PermissionsDir(dir.name="new", dir.path="data")

# Lists the contents of a user's subdirectory "new"
## Not run: ListDir(dir.name="new". dir.path="data")

# Rename the directory
## Not run: RenameDir(dir.name="new", dir.path="data", new.dir.name="newest")

 # Copy a subdirectory `newest' from the directory `data' to the home directory
## Not run: CopyDir(dir.name="newest", dir.path="data", end.path="")

# Move a subdirectory `newest' from the directory `data' to the home directory
## Not run: MoveDir(dir.name="newest", dir.path="data", end.path="")

# Deletes the subdirectory "newest"
## Not run: DeleteDir(dir.name="newest", dir.path="")
```

Mafft *MAFFT Multiple Sequence Alignment*

Description

MAFFT is a multiple sequence alignment program for unix-like operating systems. It offers a range of multiple alignment methods, L-INS-i (accurate; for alignment of about 200 sequences), FFT-NS-2 (fast; for alignment of about 10,000 sequences), etc. See http://mafft.cbrc.jp/alignment/software/. The manual is also available here: http://mafft.cbrc.jp/alignment/software/manual/manual.html

Usage

```
Mafft(file.name, file.path="", type="DNA", aln.filetype="FASTA", args=NULL,
      out.name=NULL, print.curl=FALSE, job.name=NULL, email=TRUE,
      shared.username=NULL, suppress.Warnings=FALSE)
```

Arguments

file.name	name of file to be evaluated on the Discovery Environment (DE), see details for supported input formats.
file.path	optional path to a user's subdirectory on the DE; default path is empty, which leads to the home directory
type	Two options "PROTEIN" or "DNA". This defines the type of sequences in the file, either proteins or nucleotides.
aln.filetype	Mafft does alignment of sequences, this option selects the file type of that result file. There are two options FASTA and CLUSTALW
args	Optional for arguments (i.e. flags). The Mafft model has much additional functionality that is not in this wrapper function (http://mafft.cbrc.jp/alignment/software/manual/manual.html#lbAI), see details. This option allows users to add anything that is not included (.i.e. args="–auto"), which automatically selects an appropriate strategy according to data size, see details.
out.name	the name to given to the output file (default mafft.aln)
job.name	the name to give the job being submitted
print.curl	Prints the curl statement that can be used in the terminal, if curl is installed on your computer
shared.username	With iPlant you have the ability to share folders with other users. If someone has shared a folder with you and you want to run a job with them, enter their username for this input.
suppress.Warnings	This will turn off the warnings, will speed up run time. Use with caution, if the inputs are incorrect they will not be caught.
email	By default an email will be sent to the user when the job finishes.

Details

The supported input file format is the fasta format http://en.wikipedia.org/wiki/FASTA_format.

Additional arguments, args, can be found at http://mafft.cbrc.jp/alignment/software/manual/manual.html#lbAI. The args input is text with the flags and inputs for those flags in a string like on the command line.

There are two options for output files: FASTA http://en.wikipedia.org/wiki/FASTA_format and CLUSTALW http://meme.nbcr.net/meme/doc/clustalw-format.html.

The result file is ALWAYS 'mafft.aln'.

Value

A list containing the job id and the job name is provided for jobs submitted. If an error, then a message stating the error should also be reported.

See Also

ListApps, Validate, UploadFile

Examples

```
## Not run: data(DNA.fasta)
## Not run: write.fasta(sequences = DNA.fasta, names = names(DNA.fasta), file.out = "DNA.fasta")
## Not run: Validate("username","password")
## Not run: UploadFile("DNA.fasta", filetype="FASTA-0")
## Not run: Mafft("DNA.fasta", job.name="MafftFASTA")
```

Muscle *MUSCLE alignment*

Description

MUSCLE is a program for creating multiple alignments of amino acid or nucleotide sequences. A range of options is provided that give you the choice of optimizing accuracy, speed, or some compromise between the two. The manual is also available here: http://www.drive5.com/muscle/manual/

Usage

```
Muscle(file.name, file.path="", job.name=NULL, args=NULL,
       aln.filetype="PHYLIP_INT", shared.username=NULL, out.name=NULL,
       suppress.Warnings=FALSE, email=TRUE, print.curl=FALSE)
```

Arguments

file.name	name of file to be evaluated on the Discovery Environment (DE), see details for supported input formats.
file.path	optional path to a user's subdirectory on the DE; default path is empty, which leads to the home directory
job.name	the name to give the job being submitted
out.name	name of the output file
aln.filetype	Muscle does alignment of sequences, this option selects the file type of that result file. There are six options PHYLIP_INT, PHYLIP_SEQ, HTML, FASTA, CLUSTALW, MSF, see details
print.curl	Prints the curl statement that can be used in the terminal, if curl is installed on your computer
shared.username	With iPlant you have the ability to share folders with other users. If someone has shared a folder with you and you want to run a job with them, enter their username for this input.
suppress.Warnings	This will turn off the warnings, will speed up run time. Use with caution, if the inputs are incorrect they will not be caught.
email	By default an email will be sent to the user when the job finishes.
args	Optional for arguments (i.e. flags). The Muscle model has much additional functionality that is not fit into this wrapper function (http://www.drive5.com/muscle/muscle_userguide3.8.html#_Toc260497051), see details. This option allows users to add anything that is not included (.i.e. args="-spscore"), which computes SP objective score, see details.

Details

The supported input file format is the fasta format http://en.wikipedia.org/wiki/FASTA_format.

Additional arguments, args, can be found at http://www.drive5.com/muscle/muscle_userguide3.8.html#_Toc260497051. The args input is text with the flags and inputs for those flags in a string like on the command line.

There are six possible alignment output files, named appropriately: phylip_interleaved.aln http://www.bioperl.org/wiki/PHYLIP_multiple_alignment_format, phylip_sequential.aln http://www.bioperl.org/wiki/PHYLIP_multiple_alignment_format, html.aln http://sonnhammer.sbc.su.se/Belvu.html,fasta.aln http://en.wikipedia.org/wiki/FASTA_format, clustalw.aln http://meme.nbcr.net/meme/doc/clustalw-format.html, msf.aln http://en.wikipedia.org/wiki/MSF.

Value

A list containing the job id and the job name is provided for jobs submitted. If an error, then a message stating the error should also be reported.

See Also

SubmitJob, Validate, UploadFile

Examples

```
## Not run: data(PROTEIN.fasta)
## Not run: write.fasta(sequences = PROTEIN.fasta, names = names(PROTEIN.fasta),
                    file.out = "PROTEIN.fasta")
## End(Not run)
## Not run: Validate("username","password")
## Not run: UploadFile("PROTEIN.fasta", filetype="FASTA-0")
## Not run: Muscle("PROTEIN.fasta", aln.filetype="FASTA", job.name="muscleAAfasta")
```

PLINK *PLINK-1.07*

Description

PLINK is an open-source whole genome association analysis toolset, designed to perform a range of basic, large-scale analyses in a computationally efficient manner, check http://pngu.mgh.harvard.edu/~purcell/plink/ for details.

Usage

```
PLINK(file.list="", file.path="", job.name=NULL, out.basename=NULL,
      association.method="--assoc", no.sex=TRUE, args=NULL,
      print.curl=FALSE, multi.adjust=TRUE, email=TRUE,
      shared.username=NULL, suppress.Warnings=FALSE)
```

Arguments

file.list	Names of files, in a list format, to be evaluated on the Discovery Environment (DE). There are only three possible input groups for file.list, regular fileset (.map/.ped), transposed fileset (.tfam/.tped), and binary fileset (.bed/.bim/.fam).
file.path	Optional path to a user's subdirectory on the DE; default path is empty, which leads to the home directory
job.name	The name to give the job being submitted
out.basename	The base name for the output files (not including extension).
association.method	PLINKs association methods. Choices are outlined on the PLINK webpage, http://pngu.mgh.harvard.edu/~purcell/plink/anal.shtml
no.sex	The sex column (5) is all zeroes (No sex field) (default is TRUE)
multi.adjust	Adjustment for multiple testing (recommended). A file of adjust significance values that correct for all tests performed and other metrics will be created (default is TRUE).

`args` Optional for arguments (i.e. flags). The PLINK model has so much additional functionality that it cannot all be fit into this wrapper function (http://pngu.mgh.harvard.edu/~purcell/plink/reference.shtml#options). This option allows users to add anything that is not included (.i.e. args="–silent"), to suppress output to console, see details.

`print.curl` Prints the curl statement that can be used in the terminal, if curl is installed on your computer

`shared.username` With iPlant you have the ability to share folders with other users. If someone has shared a folder with you and you want to run a job with them, enter their username for this input.

`suppress.Warnings` This will turn off the warnings, will speed up run time. Use with caution, if the inputs are incorrect they will not be caught.

`email` By default an email will be sent to the user when the job finishes.

Details

The inputs for `file.list` are to be used only in three very strict groups. Group 1: TFAM, TPED, these are the PLINK transposed filesets. Group 2: MAP, PED, these are the PLINK regular filesets. Group 3: BED, BIM, FAM, these are the PLINK binary filesets.

See the PLINK website for more information. http://pngu.mgh.harvard.edu/~purcell/plink

Additional arguments, `args`, can be found at http://pngu.mgh.harvard.edu/~purcell/plink/reference.shtml#options. The args input is text with the flags and inputs for those flags in a string like on the command line.

There are many output files possible, http://pngu.mgh.harvard.edu/~purcell/plink/reference.shtml#output

Value

A list containing the job id and the job name is provided for jobs submitted. If an error, then a message stating the error should also be reported.

See Also

SubmitJob, Validate, UploadFile

Examples

```
## Not run: data(geno_test.tfam)
## Not run: data(geno_test.tped)
## Not run: write.table(geno_test.tfam, file = "geno_test.tfam", row.names=FALSE,
                    col.names=FALSE, quote=FALSE, sep="\t")
## End(Not run)
## Not run: write.table(geno_test.tped, file = "geno_test.tped", row.names=FALSE,
                    col.names=FALSE, quote=FALSE, sep="\t")
## End(Not run)
## Not run: Validate("username","password")
```

```
## Not run: UploadFile("geno_test.tfam")
## Not run: UploadFile("geno_test.tped")
## Not run: PLINK(file.list=list("geno_test.tfam","geno_test.tped"),
               association.method="--assoc", print.curl=TRUE)
## End(Not run)
```

PLINKConversion *PLINKConversion*

Description

PLINK is an open-source whole genome association analysis toolset, designed to perform a range of basic, large-scale analyses in a computationally efficient manner, check http://pngu.mgh.harvard.edu/~purcell/plink/ for details. This function converts the standard PLINK file formats (Regular (ped/map), Transposed (tped/tfam), and Binary (bed/bim/fam)) to various other PLINK file formats.

Usage

```
PLINKConversion(file.list="", file.path="", output.type="--recode",
                job.name=NULL, shared.username=NULL, print.curl=FALSE,
                 suppress.Warnings=FALSE, out.basename=NULL, email=TRUE)
```

Arguments

file.list	Names of files, in a list format, to be evaluated on the Discovery Environment (DE). There are only three possible input groups for file.list, regular fileset (.map/.ped), transposed fileset (.tfam/.tped), and binary fileset (.bed/.bim/.fam).
file.path	Optional path to a user's subdirectory on the DE; default path is empty, which leads to the home directory
job.name	The name to give the job being submitted
output.type	PLINKs conversion methods. Choices are outlined on the PLINK webpage, see details for more information.
out.basename	The base name for the output files (not including extension).
print.curl	Prints the curl statement that can be used in the terminal, if curl is installed on your computer
shared.username	With iPlant you have the ability to share folders with other users. If someone has shared a folder with you and you want to run a job with them, enter their username for this input.
suppress.Warnings	This will turn off the warnings, will speed up run time. Use with caution, if the inputs are incorrect they will not be caught.
email	By default an email will be sent to the user when the job finishes.

Details

The inputs for `file.list` are to be used only in three very strict groups. Group 1: TFAM, TPED, these are the PLINK transposed filesets. Group 2: MAP, PED, these are the PLINK regular filesets. Group 3: BED, BIM, FAM, these are the PLINK binary filesets.

'output.type' gives the user a lot of flexibility by allowing you to type in the proper argument. All possible arguments for 'output.type' are outlined below, if a different argument is entered the application will fail.

	"output.type"	"explanation"
[,1]	–make-bed	Make .bed, .fam and .bim
[,2]	–recode	Output new .ped and .map files
[,3]	–recode12	As above, with 1/2 allele coding
[,4]	–recode-rlist	List individuals with minor allele genotypes
[,5]	–recode-lgen	Output data in long LGEN format
[,6]	–recodeHV	As above, with Haploview .info file
[,7]	–recode-fastphase	Ouput fastphase format file
[,8]	–recode-bimbam	Ouput bimbam format file
[,9]	–recode-structure	Ouput structure format file
[,10]	–recodeA	Raw data file with additive coding
[,11]	–recodeAD	Raw data file with additive/dominance coding
[,12]	–recode –transpose	Transposed Filests (.tfam/.tped)

See the PLINK website for more information. http://pngu.mgh.harvard.edu/~purcell/plink

Value

A list containing the job id and the job name is provided for jobs submitted. If an error, then a message stating the error should also be reported.

See Also

SubmitJob, Validate, UploadFile

Examples

```
## Not run: data(geno_test.tfam)
## Not run: data(geno_test.tped)
## Not run: write.table(geno_test.tfam, file = "geno_test.tfam", row.names=FALSE,
                    col.names=FALSE, quote=FALSE, sep="\t")
## End(Not run)
## Not run: write.table(geno_test.tped, file = "geno_test.tped", row.names=FALSE,
                    col.names=FALSE, quote=FALSE, sep="\t")
## End(Not run)
## Not run: Validate("username","password")
## Not run: UploadFile("geno_test.tfam")
## Not run: UploadFile("geno_test.tped")
## Not run: PLINKConversion(file.list=list(geno_test.tfam, geno_test.tped), output.type="--recode")
```

RAxML *RAxML*

Description

RAxML (Randomized Accelerated Maximum Likelihood) is a program for sequential and parallel Maximum Likelihood based inference of large phylogenetic tress. It has originall been derived from from fastDNAml which in turn was derived from Joe Felsentein's dnaml which is part of the PHYLIP package. See http://bioinformatics.oxfordjournals.org/content/suppl/2014/01/18/btu033.DC1/NewManual.pdf for details.

Usage

```
RAxML(file.name, file.path="", type="DNA", out.name=NULL,
      model=NULL, bootstrap=NULL, algorithm="d", rseed=NULL,
      multipleModelFileName=NULL, args=NULL, numcat=25, nprocs=12,
      job.name=NULL, print.curl=FALSE, shared.username=NULL,
      substitution.matrix=NULL, empirical.frequencies=FALSE,
      suppress.Warnings=FALSE, email=TRUE)
```

Arguments

`file.name`	Name of file to be evaluated on the Discovery Environment (DE), see details for supported input formats.
`file.path`	Optional path to a user's subdirectory on the DE; default path is empty, which leads to the home directory
`out.name`	the name to give the output files
`type`	Two options "PROTEIN" or "DNA". This defines the type of sequences in the file, either proteins or nucleotides.
`rseed`	(-p) required seed number, to replicate set this integer, otherwise it will randomly chosen
`model`	(-m) Substitution model. For DNA the choices are `GTRCAT`, `GTRGAMMA`, `GTRCATI` and `GTRGAMMAI`. For protein the choices are `PROTCAT`, `PROTGAMMA`, `PROTCATI` and `PROTGAMMAI`. The details p. 10-11 of manual.
`substitution.matrix`	This is only necessary for the protein alignments. The choices are: `DAYHOFF`, `DCMUT`, `JTT`, `MTREV`, `WAG`, `RTREV`, `CPREV`, `VT`, `BLOSUM62`, `MTMAM`, `LG`, `MTART`, `MTZOA`, `PMB`, `HIVB`, `HIVW`, `JTTDCMUT`, `FLU` and `GTR`, the default is `BLOSUM62`
`empirical.frequencies`	(F) Only used for protein alighnments, with appendix you can specify if you want to use empirical base frequencies.
`bootstrap`	(-b) Random Seed Number for non-parametric bootstrapping, details bottom of p. 7 of manual
`algorithm`	(-f) Select the type of algorithm/function you want. "d" is default, details bottom of p. 8 of manual

`multipleModelFileName`
: (-q) substitution model file, details bottom of p. 7 of manual

`numcat`
: (-c) Specify the number of distinct rate categories, details top of p. 13 of manual

`nprocs`
: The number of processors to be allocated to the job, default = 12

`job.name`
: the name to give the job being submitted

`print.curl`
: Prints the curl statement that can be used in the terminal, if curl is installed on your computer

`shared.username`
: With iPlant you have the ability to share folders with other users. If someone has shared a folder with you and you want to run a job with them, enter their username for this input. Then the dir.path becomes the path to the SHARED user's shared folder.

`suppress.Warnings`
: This will turn off the warnings, will speed up run time. Use with caution, if the inputs are incorrect they will not be caught.

`email`
: By default an email will be sent to the user when the job finishes.

`args`
: Optional for arguments (i.e. flags). The RAxML model has much additional functionality that is not fit into this wrapper function (http://bioinformatics.oxfordjournals.org/content/suppl/2014/01/18/btu033.DC1/NewManual.pdf), see details. This option allows users to add anything that is not included (.i.e. args="-d"), which starts the RAxML search with a random starting tree, see detals.

Details

The input file format that is supported is the interleaved phylip format http://www.bioperl.org/wiki/PHYLIP_multiple_alignment_format.

Additional arguments, `args`, can be found at http://bioinformatics.oxfordjournals.org/content/suppl/2014/01/18/btu033.DC1/NewManual.pdf. The args input is text with the flags and inputs for those flags in a string like on the command line.

For this application there are numerous output files. See pg 16-17 of the manual for complete details. RAxML outputs trees in Newick format http://en.wikipedia.org/wiki/Newick_format.

Value

A list containing the job id and the job name is provided for jobs submitted. If an error, then a message stating the error should also be reported.

See Also

`SubmitJob`, `Validate`, `UploadFile`

Examples

```
## Not run: data(phylip_interleaved_dna.aln)
## Not run: write.table(phylip_interleaved_dna.aln, "phylip_interleaved_dna.aln",
                    quote=FALSE, row.names=FALSE, col.names=FALSE)
```

```
## End(Not run)
## Not run: Validate("username","password")
## Not run: UploadFile("phylip_interleaved_dna.aln")
## Not run: myJobRaxD <- RAxML("phylip_interleaved_dna.aln", job.name="raxmlDNAphy")
```

SubmitJob *Executing analytical applications*

Description

Functions for executing and managing analytical applications deployed in the iPlant infrastructure

Usage

```
SubmitJob(application, file.path="", file.list=NULL, input.list,
          args.list=NULL, job.name, nprocs=1, private.APP=FALSE,
          suppress.Warnings=FALSE,  shared.username=NULL,
          print.curl=FALSE, email=TRUE)
Wait(job.id, minWaitsec, maxWaitsec, print=FALSE)
CheckJobStatus(job.id, history = FALSE, print.curl = FALSE)
KillJob(job.id, print.curl=FALSE)
ListJobOutput(job.id, print.curl=FALSE, print.total=TRUE)
RetrieveJob(job.id, file.vec=NULL, print.curl=FALSE, verbose=FALSE)
GetJobHistory(return.json=FALSE, print.curl=FALSE)
DeleteJob(job.id, print.curl=FALSE, ALL=FALSE)
```

Arguments

application	Name of DE application. Use the ListApps() function for a list of eligible applications. To run your own private application use private.APP =TRUE and suppress.Warnings=TRUE.
file.path	Optional path to a user's subdirectory on the DE; the default path is empty, which leads to the home directory.
file.list	A list of input files, many functions only have one input file, but some have multiple input files. These should be organized as a list. The file.list and input.list should correspond. See details for more information.
job.name	The name to give the job being submitted.
nprocs	The number of processors to be allocated to the job, default = 1.
private.APP	Optional argument for submitting a job on your own private application, default is FALSE
job.id	The unique ID number given to a submitted job.
input.list	A list of type of input that is specific to the application. See details for more information.
args.list	A list of input options available for the application. These are usually the flagging options in command line invocations. See details for more information.

`return.json` Optional screen output that displays all of the results from the api, default = FALSE.

`file.vec` Names of output files to download, can be one or many. If left NULL, all the files in the job output will download.

`minWaitsec` A range of times (in seconds) must be entered for the `Wait` function. This entry is the minimum time (in seconds) of that range.

`maxWaitsec` A range of times (in seconds) must be entered for the `Wait` function. This entry is the maximum time (in seconds) of that range.

`print.curl` Prints the curl statement that can be used in the terminal, if curl is installed on your computer.

`print.total` Option only for the `ListJobOuput` function this option will print the total number of files in the folder.

`print` Only for the `Wait` function, when `print=TRUE`, it simply prints the status when the job is complete.

`verbose` For the `RetrieveJob` function this option will print the names of the files as they are downloaded.

`shared.username` With iPlant you have the ability to share folders with other users. If someone has shared a folder with you and you want to run a job with them, enter their username for this input.

`suppress.Warnings` This will turn off the warnings, will speed up run time. Use with caution, if the inputs are incorrect they will not be caught. If the application you are running is a private application have suppress.Warnings=TRUE.

`email` This option is only on the `SubmitJob` function. By default an email will be sent to the user when the job finishes.

`ALL` This option is only on the `DeleteJob` function. If ALL=TRUE then all jobs in the job history will be deleted.

`history` This option is only on the `CheckJobStatus` function. If TRUE, then will show entire history of job.

Details

The function `SubmitJob`, takes inputs and arguments and submits a job on the Agave API. The `SubmitJob` function will run the `application` with the file inputs `file.list` that are in the directory `file.path`. The files within `file.list` need to match the expected file types for the application (defined in `input.list` argument). The appropriate options for the `application` need to be outlined in `input.list` and potentially `args.list`. The `SubmitJob` function outputs the `job.id` and the `job name`. With that `job.id` you can run `CheckJobStatus(job.id)` to check the status of your job, and the `job name` can be used in workflows. The stages for `CheckJobStatus` are:

```
PENDING
 STAGING_INPUTS
 CLEANING_UP
 ARCHIVING
```

```
STAGING_JOB
FINISHED
KILLED
FAILED
STOPPED
RUNNING
PAUSED
QUEUED
SUBMITTING
STAGED
PROCESSING_INPUTS
ARCHIVING_FINISHED
ARCHIVING_FAILED
```

When it is finished it will read either ARCHIVING_FINISHED or FINISHED, unless it failed. Use the KillJob function to terminate a running job. Use the Wait function to wait until job is finished. Be cautious using the Wait function, because it will lock up the workspace until the job is finished. When the job is finished then use the ListJobOutput function to see all of the files in your job. The number of output files varies by application. The RetrieveJob function takes the job.id and the file.vec as input, and downloads the specified files in the file.vec. The files will be downloaded to your current working directory (getwd()). The file.vec contains the file names that you want to download. This vector is a subset of the output from ListJobOutput. The DeleteJob function then deletes the job and the correponding output folder that was generated from running the job. Using the option DeleteJob(ALL=TRUE) will delete all jobs in a user's job history. The GetJobHistory function displays all jobs in your history that have not been deleted.

For the SubmitJob function the application must match an application name that is in the output from the ListApps function. For the input.list use the GetAppInfo function, the 'kind' column verifies if "input" or "output". What goes in the input.list is only the name in the 'id' column when the 'kind' column is "input". For example, when the application is "muscle-lonestar-3.8.31u2", we can use GetAppInfo("muscle-lonestar-3.8.31u2")$Information to determine that the application is expecting "stdin" as its first input file (input.list=list("stdin")). For the application "velveth-1.2.07u1", GetAppInfo("velveth-1.2.07u1")$Information, tells us that the application will expect six input files, which should be in the order: input.list=list("reads1", "reads2", "reads3"

A few things to note: 1) depending on the application, the input.list can be shorter than the the number of inputs, for example, using the "velveth-1.2.07u1" application, the input list could be input.list=list("reads1", "reads2", "reads3"); 2) the file.list should always be the same length as input.list; 3) for args.list use GetAppInfo function, when the 'kind' column is 'parameters', those are the inputs for args.list. For velveth-1.2.07u1 the args.list is as follows, list(c("format1", value), c("kmer", value), c("Output", value)). The list can be as long as the number of options.

Value

A list containing the job id and the job name is provided for jobs submitted. If an error, then a message stating the error should also be reported.

See Also

ListApps, Validate, UploadFile

Examples

```
## Not run: data(DNA.fasta)
## Not run: write.fasta(sequences = DNA.fasta, names = names(DNA.fasta), file.out = "DNA.fasta")
## Not run: Validate("username","password")
## Not run: UploadFile("DNA.fasta", filetype="FASTA-0")

# Submit a MUSCLE job using the provided data in the package.  The job will return
# a job id and job name
## Not run: myJob <- SubmitJob(application="Muscle-3.8.32u4", file.list=list("DNA.fasta"),
                            input.list=list("stdin"), args.list=list(c("arguments",
                            "-phyiout")), job.name="muscleDNA")
## End(Not run)

# Check the status of any job
## Not run: CheckJobStatus(myJob$id)

# Lists and output files a job has created
## Not run: ListJobOutput(myJob$id)

# Might want to kill job if incorrect running
## Not run: KillJob(myJob$id)
# Need to wait for job to be done
## Not run: Wait(myJob$id, 5, 1800, print=TRUE)

# Download output files
## Not run: RetrieveJob(myJob$id, ListJobOutput(myJob$id, print.total=FALSE))

# View job history
## Not run: GetJobHistory()

# Delete Job
## Not run: DeleteJob(myJob$id)
## Not run: DeleteJob(ALL=TRUE)
```

TNRS *Resolving Names via TNRS*

Description

The function ResolveNames interacts with the TNRS server at iPlant that uses fuzzy name matching to find standardized taxonomic plant names. GetPhylotasticToken is similar, but can be used for animal taxa as well, since it also utilizes the NCBI and ITIS databases. GetPhylotasticToken returns a unique token that can be checked online or using the RetrieveTNRSNames function. CompareTNRS will compare the original list of names to the returned TNRS names to see which names changed.

Usage

```
ResolveNames(names, max.per.call=100, verbose=TRUE)
GetPhylotasticToken(names, max.per.call=100, verbose=TRUE)
RetrieveTNRSNames(names, token, source=c("iPlant_TNRS", "NCBI"),
                  match.threshold=0.5, verbose=FALSE)
CompareNames(old.names, new.names, verbose=TRUE)
```

Arguments

names	Vector of names to be resolved via TNRS
max.per.call	The number of names to submit at a time, default is 100
verbose	Optional screen output that displays all of the results from the api
token	Unique identifier from the GetPhylotasticToken function used to retrieve names
source	Which source to utilize to check names
match.threshold	Threshold to accept new name
old.names	Original names
new.names	Returned TNRS names

Value

Vector of taxonomic names for ResolveNames and RetrieveTNRSNames. Unique identifier (=token) for GetPhylotasticToken

Examples

```
data(DNA.fasta)
speciesNames <- names(DNA.fasta)

#Taxonomic name checking by iPlant TNRS
## Not run: TNRSspeciesNames <- ResolveNames(speciesNames, 100, verbose=F)

#Taxonomic name checking by Phylotastic Taxosaurus
token <- GetPhylotasticToken(speciesNames)
TNRSSpeciesNames <- RetrieveTNRSNames(speciesNames, token, "NCBI")
CompareNames(speciesNames, TNRSSpeciesNames, verbose=TRUE)
```

UploadFile *File management*

Description

Functions for uploading and manipulating files in the iPlant infrastructure

Usage

```
UploadFile(local.file.name, local.file.path="", filetype=NULL,
           print.curl=FALSE, suppress.Warnings=FALSE)
ShareFile(file.name, file.path="", shared.username, read=TRUE,
          execute=TRUE, write=TRUE, print.curl=FALSE, suppress.Warnings=FALSE)
PermissionsFile(file.name, file.path="", print.curl=FALSE, suppress.Warnings=FALSE)
RenameFile(file.name, new.file.name, file.path="", print.curl=FALSE,
           suppress.Warnings=FALSE)
CopyFile(file.name, file.path="", end.path="", print.curl=FALSE,
         suppress.Warnings=FALSE)
MoveFile(file.name, file.path="", end.path="", print.curl=FALSE,
         suppress.Warnings=FALSE)
DeleteFile(file.name, file.path="", print.curl=FALSE, suppress.Warnings=FALSE)
SupportFile(print.curl=FALSE, suppress.Warnings=FALSE)
```

Arguments

`local.file.name`
: Name of local file on user's computer. This file cannot be an object in the R workspace.

`local.file.path`
: optional path to a sub directory where local file on user's computer is located (include ENTIRE path); default path is empty which leads to a users home directory

`filetype`
: format of file that is to be uploaded. Currently 34 file formats supported. A filetype is not required to upload the file, but some applications require that your data set have a file type assigned to it.

`file.name`
: Name of file to be modified on the iPlant servers.

`new.file.name`
: the new name of the file on the iPlant server to be renamed

`file.path`
: optional path to a sub directory where file on the iPlant server is located; default path is empty which leads to a users home directory

`end.path`
: Path to destination sub directory where file on DE is moved to; default path is empty which leads to a users home directory

`read`
: Gives read permissions for shared file

`write`
: Gives write permissions for shared file

`execute`
: Gives execute permissions for shared file

`shared.username`
: With iPlant you have the ability to share files with other users, their username is the value for the shared.username

`print.curl`
: Prints the curl statement that can be used in the terminal, if curl is installed on your computer

`suppress.Warnings`
: This will turn off the warnings, will speed up run time. Use with caution, if the inputs are incorrect they will not be caught.

Details

The SupportFile function provides a list of the supported file types that can be uploaded onto the iPlant servers. The UploadFile function uploads a file, local.file.name, that is on your computer (i.e. local directory), in the directory local.file.path. If the local.file.path is not specified your working directory is the default path (use getwd() to find that). The filetype is NOT required, but if you do use it then use the SupportFile function to view the 34 file formats that are supported (i.e. for a fasta file use FASTA-0). iPlant offers sharing of files between two (or more) iPlant users, the ShareFile makes it so. Once sharing is done a user can use PermissionsFile on any file to find who the file is shared with. The RenameFile function renames the file.name in file.path (on the iPlant servers), to the new.file.name. The CopyFile function copies a file.name in the file.path to the end.path. The MoveFile function moves a file.name in the file.path to the end.path. The DeleteFile function deletes the file.name in the file.path.

Value

If an error, then a message stating the error should also be reported.

See Also

ListDir

Examples

```
# Write .fasta file to home directory
## Not run: data(DNA.fasta)
## Not run: write.fasta(sequences = DNA.fasta, names = names(DNA.fasta),
                        file.out = "DNA.fasta")
## End(Not run)
## Not run: UploadFile(local.file.name="DNA.fasta", local.file.path="path/to/dir",
                       filetype="FASTA-0")
## End(Not run)

# Upload a file to the DE
## Not run: data(geno_test.tfam)
## Not run: UploadFile(local.file.name="geno_test.tfam")

# Shares the file named "DNA.fasta" with the iPlant user "dude"
## Not run: ShareFile(file.name="DNA.fasta", shared.username="dude")

# Checks permissions on file "DNA.fasta"
## Not run: PermissionsFile(file.name="DNA.fasta")

# Rename a file
## Not run: RenameFile(file.name="DNA.fasta", new.file.name="lp.fasta")

# Copy a file from the subdirectory 'rPlant' to the home directory
## Not run: CopyFile(file.name="lp.fasta", file.path="", end.path="data")

# Move a file from the subdirectory 'rPlant' to the home directory
## Not run: MoveFile(file.name="lp.fasta", file.path="", end.path="data")
```

```
# Delete a file in the home directory
## Not run: DeleteFile(file.name="lp.fasta", DE.file.path="data")

# Lists file types that are supported
## Not run: SupportFile()
```

`Validate` *Authentication functions*

Description

This basic function authenticates users.

Usage

```
Validate(user, pwd, api="agave", print.curl=FALSE)
```

Arguments

`user`	iPlant Discovery Environment username
`pwd`	iPlant Discovery Environment password
`api`	The API to be interfaced with. The only supported API is "agave"
`print.curl`	Prints the curl statement that can be used in the terminal, if curl is installed on your computer

Details

All that is necessary for the `Validate` function is for the user to provide the username and password combination; then it will verify that they are a valid pair for iPlant. To get a valid pair of credentials go to `http://user.iplantcollaborative.org/` and register.

Sometimes the API will have trouble validating. If this is the case, you will receive the following message from 'rPlant': "API Error, please retry". This only seems to occur in a new R session. If this occurs, please try the validation a second time. If it does not work the second time, then there may be a problem with your account.

Value

The `Validate` function will create a new R environment that stores variables that communicate automatically with the API. These include, an access token (foundation) or an API consumer key and secret (agave), the expiration, the user name and password, and a few other variables. To see the entire list of environmental variables, see `ls(rplant.env)`.

The function will print the error "Authentication failed" if if fails. Sometimes the API will require a second validation, in which case 'rPlant' will report an error asking you to retry. If it is successful, nothing will print to screen, however you can check that validation occurred by checking rplant.env$consumer_key and rplant.env$consumer_secret.

Validation will have to occur at every new R session. If a session times out and expires, you do not have to renew validation. If the rplant.env is an object found within the workspace, your token will be auto-renewed by using one of 'rPlant's' functions (ie. `SubmitJob`, `CheckJobStatus`, etc.).

Examples

```
## Not run: Validate("username", "password")
```

Index