Package 'BgeeCall'

October 16, 2020

Type Package

Title Automatic RNA-Seq present/absent gene expression calls generation

Version 1.4.1

Date 2019-03

Description BgeeCall allows to generate present/absent gene expression calls without using an arbitrary cutoff like TPM<1.
Calls are generated based on reference intergenic sequences. These sequences are generated based on expression of all RNA-Seq
libraries of each species integrated in Bgee (https://bgee.org).

Depends R (>= 3.6)

Imports GenomicFeatures, rhdf5, tximport, Biostrings, rtracklayer, biomaRt, jsonlite, methods, grDevices, graphics, stats, utils

License GPL-3

URL https://github.com/BgeeDB/BgeeCall

BugReports https://github.com/BgeeDB/BgeeCall/issues

VignetteBuilder knitr

biocViews Software, GeneExpression, RNASeq

Suggests knitr, testthat, rmarkdown, AnnotationHub, httr

RoxygenNote 7.0.2

Encoding UTF-8

NeedsCompilation no

SystemRequirements kallisto

git_url https://git.bioconductor.org/packages/BgeeCall

git_branch RELEASE_3_11

git_last_commit 0dbbf56

git_last_commit_date 2020-05-12

Date/Publication 2020-10-16

Author Julien Wollbrett [aut, cre],
Julien Roux [aut],
Sara Fonseca Costa [ctb],
Marc Robinson Rechavi [ctb],
Frederic Bastian [aut]

Maintainer Julien Wollbrett <julien.wollbrett@unil.ch>

R topics documented:

`AbundanceMetadata-class`

AbundanceMetadata s4 class

Description

An S4 class that is the parent class of all abundance tool Classes. It contains information needed to all abundance tools. This class can be seen as an abstract class, you should never instanciate it.

Slots

`txOut` Similar to tximport txOut parameter. Allows to keep abundance at transcript level if TRUE (default = FALSE)

`ignoreTxVersion` logical used to remove transcript version in transcript ID if TRUE (default = FALSE)

`cutoff` numeric corresponding to the proportion of intergenic present divided by proportion of protein coding present (default = 0.05). In the Bgee pipeline this cutoff is fixed and its value is 0.05. Be careful when changing this parameter as it could have a huge impact on your present/absent calls.

`full_transcriptome_file` Name of the fasta file containing both transcriptomic and intergenic regions. This file is created by the pipeline. You should edit this slot only if you already have such a file with a different name.

`tx2gene_file` Name of the file containing the mapping between transcript IDs and gene IDs (See the tximport package vignette for more details). This file is created by the pipeline. You should edit this slot only if you already have such a file with a different name. This file must be store at get_species_path()

`tx2gene_file_without_version` Name of the file containing the mapping between transcript IDs and gene IDs if ignoreTxVersion == TRUE (See the tximport package vignette for more details). This file is created by the pipeline. You should edit this slot only if you already have such a file with a different name. This file must be store at get_species_path()

`gene2biotype_file` Name of the file containing the mapping between gene IDs and biotypes. This file is created by the pipeline. You should edit this slot only if you already have such a file with a different name.

`tool_name` Name of the tool that will be use to generate transcript abundance estimation. All descendant of this class have to define a value for this slot (in the prototype section)

`abundance_file` Name of the transcript-level abundance file. All descendant of this class have to define a value for this slot (in the prototype section)

`read_size_kmer_threshold` read size of the library below which transcript index is created using a smaller kmer size

`transcript_id_header` Name of the header of the column that contains transcript ID

`count_header` Name of the header of the column that contains count

`abundance_header` Name of the header of the column that contains abundance

`eff_length_header` Name of the header of the column that contains effective length

`transcript_calls_file_name` default name of file containing all transcript ids and calls (if calls created at transcript level)

`gene_calls_file_name` default name of file containing all gene ids and calls (if calls created at gene level)

`transcript_cutoff_file_name` default name of file containing summary of cutoff used to generate transcript expression calls (if calls created at transcript level)

`gene_cutoff_file_name` default name of file containing summary of cutoff used to generate gene expression calls (if calls created at gene level)

`transcript_distribution_file_name` default name of density plot file containing TPM distribution of all transcripts (if calls created at transcript level)

`gene_distribution_file_name` default name of density plot file containing TPM distribution of all genes (if calls created at gene level)

BgeeMetadata-class *BgeeMetadata S4 class*

Description

An S4 class that contains all information to retrieve intergenic regions generated by Bgee.

Slots

`intergenic_release` Bgee intergenic release that will be used

`all_releases` list of all reference intergenic releases that can be used to generate your present/absent expression calls.

`intergenic_prefix` String used to generate an intergenic release specific output directory

`custom_intergenic_path` path to a local version of reference intergenic fasta file. If NULL (by default) the reference intergenic fasta file will be downloaded. If not NULL BgeeCall will merge this local reference intergenic file with the transcriptome. Except if you generated your own intergenic regions always keep it NULL.

create_kallisto_index *Create kallisto indexes.*

Description

This function creates kallisto indexes. Two indexes can be created depending on the reads size (see 'AbundanceMetadata@read_size_kmer_threshold' and 'UserMetadata@reads_size' for more information). One with default kmer value (31 nt) and one with kmer size of 15 nt. In order to generate.

Usage

```
create_kallisto_index(
  myKallistoMetadata,
  myBgeeMetadata,
  myUserMetadata,
  transcriptome_path = ""
)
```

Arguments

`myKallistoMetadata` A Reference Class KallistoMetadata object.

`myBgeeMetadata` A Reference Class BgeeMetadata object.

`myUserMetadata` A Reference Class UserMetadata object.

`transcriptome_path` path to the transcriptome fasta file. If no path is provided the default path created using BgeeCall will be used. IMPORTANT : in BgeeCall the transcriptome used to generate present/absent calls contains both intergenic sequences downloaded from Bgee and the reference transcriptome. If this function is run to generate present/absent then 'transcriptome_path' has to be empty

Value

create kallisto index and save it on the hard drive

Author(s)

Julien Wollbrett.

Examples

```
## Not run:
# first a transcriptome is needed. Here it is downloaded from AnnotationHub
library(AnnotationHub)
ah <- AnnotationHub()
ah_resources <- query(ah, c('Ensembl', 'Caenorhabditis elegans', '84'))

# kallisto can not deal with S4 objects. A Path to a transcriptome file is
# required
transcriptome_object <- rtracklayer::import.2bit(ah_resources[['AH50453']])
transcriptome_path <- file.path(getwd(),'transcriptome.fa')
Biostrings::writeXStringSet(transcriptome_object, transcriptome_path)



# initialize objects needed to create destination folder
bgee <- new('BgeeMetadata')
user <- new('UserMetadata', species_id = '6239')
kallisto <- new('KallistoMetadata')

# generate transcriptome index
create_kallisto_index(kallisto, bgee, user, transcriptome_path)

## End(Not run)
```

---

download_fasta_intergenic

Download fasta intergenic

Description

Check if reference intergenic fasta file has already been downloaded. If not the file is downloaded from Bgee FTP or from the community repository depending on myBgeeMetadata@intergenic_release. if myBgeeMetadata@intergenic_release == "community" then reference intergenic wil be downloaded from the Zenodo community repository. Otherwise Reference intergenic sequences will be downloaded from the official Bgee FTP. Be careful when using reference intergenic sequences generated by the community as the Bgee team do not deeply review them.

Usage

```
download_fasta_intergenic(
  myBgeeMetadata = new("BgeeMetadata"),
  myUserMetadata,
  intergenic_file
)
```

Arguments

myBgeeMetadata	A Reference Class BgeeMetadata object (optional)
myUserMetadata	A Reference Class UserMetadata object.
intergenic_file	path where intergenic file will be saved

Value

download fasta intergenic from Bgee FTP or from the Zenodo community and save it locally

Examples

```
{
bgee_intergenic_file <- file.path(getwd(), 'intergenic.fasta')
userMetadata <- new('UserMetadata', species_id = '7227')
}
```

---

download_kallisto *Download binary version of kallisto.*

Description

Check your OS and download correct binary version of kallisto.

Usage

```
download_kallisto(myKallistoMetadata, myUserMetadata)
```

Arguments

myKallistoMetadata	A Reference Class KallistoMetadata object.
myUserMetadata	A Reference Class UserMetadata object.

Value

save uncompressed executable of kallisto on the hard drive

Author(s)

Julien Wollbrett.

Examples

```
{
  kallisto <- new('KallistoMetadata')
  user <- new('UserMetadata')
  download_kallisto(kallisto, user)
}
```

generate_calls_workflow *generate present/absent calls*

Description

Main function running the workflow allowing to generate present/absent calls from a file, a data.frame, or objects of the classe UserMetadata (please choose only 1 out of the 3) This workflow is highly tunable by editing default values of the slots of S4 objects. For more information on how to tune the workflow please have a look at the vignette and the documentation of the classes KallistoMetadata, AbundanceMetadata, UserMetadata, and BgeeMetadata

Usage

```
generate_calls_workflow(
  abundanceMetadata = new("KallistoMetadata"),
  bgeeMetadata = new("BgeeMetadata"),
  userMetadata = NULL,
  userDataFrame = NULL,
  userFile = NULL
)
```

Arguments

abundanceMetadata	A Reference Class BgeeMetadata object (optional) allowing to tune your gene quantification abundance analyze
bgeeMetadata	A Reference Class BgeeMetadata object (optional) allowing to choose the version of reference intergenic sequences
userMetadata	A Reference Class UserMetadata object (optional). generate present/allows calls using objects of the UserMetadata class. Can be one object or a list of objects.
userDataFrame	a data.frame comtaining all information to generate present/absent calls. Each line of this data.frame will generate calls for one RNA-Seq library. This data.frame must contains 7 columns : - species_id : The ensembl species ID - run_ids : (optional) allows to generate calls for a subpart of all runs of the library. must be a character or a list of characters. - reads_size (optional) the size of the reads of the library (Default = 50) if the reads size is lower than 51 abundance quantification wil be run with a smaller kmer size - rnaseq_lib_path : path to RNA-Seq library directory - transcriptome_path : path to transcriptome file - annotation_path : path to annotation file - working_path : root of the directory where results will be written
userFile	path to a tsv file containing 7 columns. these columns are the same than for userDataFrame (see above). a template of this file is available at the root of the package and accessible with the command system.file('userMetadataTemplate.tsv', package = 'BgeeCall')

Value

paths to the 4 results files (see vignette for more details)

Author(s)

Julien Wollbrett

See Also

AbundanceMetadata, KallistoMetadata, BgeeMetadata, UserMetadata

Examples

```
## Not run:
# import gene annotation and transcriptome from AnnotationHub
library(AnnotationHub)
ah <- AnnotationHub()
ah_resources <- query(ah, c('Ensembl', 'Caenorhabditis elegans', '84'))
annotation_object <- ah_resources[['AH50789']]
transcriptome_object <- rtracklayer::import.2bit(ah_resources[['AH50453']])

# instanciate BgeeCall object
# add annotation and transcriptome in the user_BgeeCall object
# it is possible to import them using an S4 object (GRanges, DNAStringSet)
# or a file (gtf, fasta) with methods setAnnotationFromFile() and
# setTranscriptomeFromFile()
user_BgeeCall <- setAnnotationFromObject(user_BgeeCall,
                                         annotation_object,
                                         'WBcel235_84')
user_BgeeCall <- setTranscriptomeFromObject(user_BgeeCall,
                                          transcriptome_object,
                                          'WBcel235')
# provide path to the directory of your RNA-Seq library
user_BgeeCall <- setRNASeqLibPath(user_BgeeCall,
                 system.file('extdata', 'SRX099901_subset',
                 package = 'BgeeCall'))

# run the full BgeeCall workflow
calls_output <- generate_calls_workflow(
             userMetadata = user_BgeeCall)

## End(Not run)
```

---

generate_presence_absence

Generate presence absence

Description

Generate presence absence calls. It correponds to the last part of the generation of the expression calls workflow. It runs the last part of the workflow generating presetn/absent expression calls. This function should only be used by advanced user who already manually run all previous parts of the pipeline. If you are not an advanced user it is safer to run the function “generate_calls_workflow“ that run all steps of the worklow

Usage

```
generate_presence_absence(
  myAbundanceMetadata = new("KallistoMetadata"),
  myBgeeMetadata = new("BgeeMetadata"),
  myUserMetadata
)
```

Arguments

myAbundanceMetadata	A descendant object of the Class myAbundanceMetadata (optional).
myBgeeMetadata	A Class BgeeMetadata object (optional).
myUserMetadata	A Class UserMetadata object.

Value

path to the 4 output files

Author(s)

Julien Wollbrett

Julien Roux

See Also

generate_calls_workflow

Examples

```
{
# this example reuse data present in the directory 'extdata' of the package.
user <- new('UserMetadata', working_path = system.file('extdata',
package = 'BgeeCall'), species_id = '6239', rnaseq_lib_path = system.file(
'extdata', 'SRX099901_subset', package = 'BgeeCall'),
annotation_name = 'WBcel235_84', simple_arborescence = TRUE)
calls_output <- generate_presence_absence(myUserMetadata = user)

#
}
```

---

getIntergenicPrefix *'intergenic_prefix' Getter*

Description

Get value of the 'intergenic_prefix' slot

Usage

```
getIntergenicPrefix(bgeeObject)

## S4 method for signature 'BgeeMetadata'
getIntergenicPrefix(bgeeObject)
```

Arguments

bgeeObject	The BgeeMetadata object

Value

the value of the ‘intergenic_prefix‘ slot of the object

Examples

```
{
bgee <- new("BgeeMetadata")
intergenic_prefix <- getIntergenicPrefix(bgee)
}
```

---

getIntergenicRelease *‘intergenic_release‘ Getter*

Description

Get value of the ‘intergenic_release‘ slot

Usage

```
getIntergenicRelease(bgeeObject)

## S4 method for signature 'BgeeMetadata'
getIntergenicRelease(bgeeObject)
```

Arguments

bgeeObject	The BgeeMetadata object

Value

the value of the ‘intergenic_release‘ slot of the object

Examples

```
{
bgee <- new("BgeeMetadata")
intergenic_release <- getIntergenicRelease(bgee)
}
```

getRunIds *'run_ids' Getter*

Description

Get value of the 'run_ids' slot

Usage

```
getRunIds(userObject)

## S4 method for signature 'UserMetadata'
getRunIds(userObject)
```

Arguments

userObject	The UserMetadata object

Value

the value of the 'run_ids' slot of the object

Examples

```
{
user <- new("UserMetadata")
run_ids <- getRunIds(user)
}
```

getSimpleArborescence *'simple_arborescence' Getter*

Description

Get value of the 'simple_arborescence' slot

Usage

```
getSimpleArborescence(userObject)

## S4 method for signature 'UserMetadata'
getSimpleArborescence(userObject)
```

Arguments

userObject	The UserMetadata object

Value

the value of the 'simple_arborescence' slot of the object

Examples

```
{
user <- new("UserMetadata")
simple_arborescence <- getSimpleArborescence(user)
}
```

`getWorkingPath` *'working_path' Getter*

Description

Get value of the 'working_path' slot

Usage

```
getWorkingPath(userObject)

## S4 method for signature 'UserMetadata'
getWorkingPath(userObject)
```

Arguments

`userObject`	The UserMetadata object

Value

the value of the 'working_path' slot of the object

Examples

```
{
user <- new("UserMetadata")
working_path <- getWorkingPath(user)
}
```

`KallistoMetadata-class` *KallistoMetadata S4 class*

Description

An S4 class that is the desendant of the AbundanceMetadata class. It contains all metadata needed to run kallisto analysis. All slots of this class have a default value. You do not need to edit them to run the package

Slots

`download_kallisto` A logical allowing to use an already installed version of kallisto or to download a version that will be used only by this package

`kallisto_windows_url` URL to the binary of kallisto for windows

`kallisto_linux_url` URL to the binary of kallisto for linux

`kallisto_osx_url` URL to the binary of kallisto for MacOS

`kallisto_windows_dir` Name of the directory where kallisto will be installed on windows

`kallisto_linux_dir` Name of the directory where kallisto will be installed on linux

`kallisto_osx_dir` Name of the directory where kallisto will be installed on Mac

`unix_kallisto_name` Name of the kallisto executable in linux and macOS

`windows_kallisto_name` Name of the kallisto executable in windows

`index_file` Name of index file generated by kallisto with default kmer size. It will be generated using the fasta file that contains both transcriptomic and intergenic regions. Do not use an index you generated outside of this package. This file is created by the pipeline. You should edit this slot only if you already have such a file with a different name. This file must be store at get_tool_path()

`k15_index_file` same as index_file. This index is generated with smallest kmers and will be used only for libraries containing reads smallest than 50nt.

`single_end_parameters` kallisto parameters used to run a single end mapping

`pair_end_parameters` kallisto parameters used to run a pair end mapping

`overwrite_index` logical allowing to overwrite already existing index. FALSE by default. Then by default already existing index files will not be generated again.

`overwrite_quant` logical allowing to overwrite already existing abundance.txt files. FALSE by default. Then by default already existing quantitfdication files will not be generated again.

`overwrite_calls` logical allowing to overwrite already existing present/absent calls. FALSE by default. Then by default already generated calls will not be generated again.

`list_bgee_ref_intergenic_species` *List species having Bgee reference intergenic sequences*

Description

Return information related to species having Bgee reference intergenic sequences available for the selected Bgee intergenic release: - speciesId : the NCBI species ID of the species - specieName : scientific species name - numberOfLibraries : number of libraries used to generate these reference intergenic sequences If a BgeeMetadata object is provided this function retrieve the list of species using BgeeMetadata@intergenic_release. If only a 'release' is provided it will use it to retrieve the list of species. If none of them are provided the default Bgee reference intergenic release will be used.

Usage

```
list_bgee_ref_intergenic_species(myBgeeMetadata = NULL, release = NULL)
```

Arguments

myBgeeMetadata	A Reference Class BgeeMetadata object
release	A Bgee reference intergenic release name

Value

list all species having reference intergenic sequences available in the selected release

Author(s)

Julien Wollbrett

Examples

```
{
bgee <- new("BgeeMetadata")
list_bgee_ref_intergenic_species(myBgeeMetadata = bgee)
list_bgee_ref_intergenic_species(release = '0.2')
}
```

---

list_community_ref_intergenic_species

List species having reference intergenic sequences created by the BgeeCall community

---

Description

Return information related to species having reference intergenic sequences created by the BgeeCall community - speciesId : the NCBI species ID of the species - url : url to the reference intergenic fasta file - numberOfLibraries : number of libraries used to generate these reference intergenic sequences

Usage

```
list_community_ref_intergenic_species()
```

Value

list all species having reference intergenic sequences created by the community

Author(s)

Julien Wollbrett

Examples

```
{
list_community_ref_intergenic_species()
}
```

list_intergenic_release

List reference intergenic releases usable with the BgeeCall package

Description

Returns information on available Bgee intergenic releases, the access URL for FTP, and the date of release

Usage

```
list_intergenic_release(release = NULL)
```

Arguments

release	A character specifying a targeted release number (e.g., '0.1'). If not specified, all available releases are shown.

Value

A data frame with information on Bgee intergenic releases available to use with the BgeeCall package.

Author(s)

Julien Wollbrett

Examples

```
{
 list_intergenic_release()
}
```

merge_transcriptome_and_intergenic

Merge transcriptome file provided by the user with the Bgee intergenic fasta file.

Description

This function will create a file corresponding to the concatenation of the transcriptome fasta file provided by the user and the corresponding intergenic fasta file created by Bgee.

Usage

```
merge_transcriptome_and_intergenic(
  myKallistoMetadata,
  myBgeeMetadata,
  myUserMetadata
)
```

Arguments

myKallistoMetadata	A Reference Class KallistoMetadata object.
myBgeeMetadata	A Reference Class BgeeMetadata object.
myUserMetadata	A Reference Class UserMetadata object.

Value

save merged file on the hard drive

Author(s)

Julien Wollbrett.

Examples

```
{
bgee <- new('BgeeMetadata', intergenic_release = '0.1')
user <- new ('UserMetadata', species_id = '6239')
kallisto <- new('KallistoMetadata')
user <- setTranscriptomeFromFile(user, system.file("extdata",
"transcriptome.fa", package = "BgeeCall"), 'WBcel235')
merge_transcriptome_and_intergenic(kallisto, bgee, user)
}
```

---

run_kallisto *Run one kallisto abundance analyse*

Description

Run kallisto and all preliminary steps if needed like : - creation of transcriptome with intergenic (if needed) - installation of kallisto (if needed) - index creation (if needed) - run kallisto quantification

Usage

```
run_kallisto(
  myKallistoMetadata,
  myBgeeMetadata,
  myUserMetadata,
  transcriptome_path = ""
)
```

Arguments

myKallistoMetadata	A Reference Class KallistoMetadata object.
myBgeeMetadata	A Reference Class BgeeMetadata object.
myUserMetadata	A Reference Class UserMetadata object. This object has to be edited before running kallisto @seealso UserMetadata.R

transcriptome_path
: path to the transcriptome fasta file. If no path is provided the default path created using BgeeCall will be used. IMPORTANT : in BgeeCall the transcriptome used to generate present/absent calls contains both intergenic sequences downloaded from Bgee and the reference transcriptome.

Value

create kallisto output files and save them on the hard drive

Author(s)

Julien Wollbrett.

Examples

```
## Not run:
# first a transcriptome is needed. Here it is downloaded from AnnotationHub
library(AnnotationHub)
ah <- AnnotationHub()
ah_resources <- query(ah, c('Ensembl', 'Caenorhabditis elegans', '84'))

# kallisto can not deal with S4 objects. Path to transcriptome file is
# required
transcriptome_object <- rtracklayer::import.2bit(ah_resources[['AH50453']])
transcriptome_path <- file.path(getwd(),'transcriptome.fa')
Biostrings::writeXStringSet(transcriptome_object, transcriptome_path)

# initialize objects needed to create destination folder
bgee <- new('BgeeMetadata')
user <- new('UserMetadata', species_id = '6239')
user <- setRNASeqLibPath(user, system.file(
                    'extdata', 'SRX099901_subset',
                    package = 'BgeeCall'))
kallisto <- new('KallistoMetadata')

# generate transcriptome index
run_kallisto(kallisto, bgee, user, transcriptome_path)

## End(Not run)
```

---

run_tximport *Run tximport*

Description

Run tximport. Will summarize abundance estimation from transcript level to gene level if 'myAbundanceMetadata@txout == FALSE'. Otherwise keep abundance estimation at transcript level.

Usage

```
run_tximport(
  myAbundanceMetadata = new("KallistoMetadata"),
  myBgeeMetadata = new("BgeeMetadata"),
  myUserMetadata,
  abundanceFile = ""
)
```

Arguments

myAbundanceMetadata	A descendant object of the Class myAbundanceMetadata.
myBgeeMetadata	A Reference Class BgeeMetadata object.
myUserMetadata	A Reference Class UserMetadata object.
abundanceFile	(Optional) Path to the abundance file. NULL by default. If not NULL, the file located at 'abundanceFile' will be used to run tximport. Otherwise (Default) the path to the abundance file is deduced fom attributes of classes 'BgeeMetadata', 'UserMetadata' and 'AbundanceMetadata'

Value

a tximport object

Author(s)

Julien Wollbrett

Examples

```
{
user <- new("UserMetadata", working_path = system.file("extdata",
    package = "BgeeCall"), species_id = "6239",
  rnaseq_lib_path = system.file("extdata",
    "SRX099901_subset", package = "BgeeCall"),
  annotation_name = "WBcel235_84", simple_arborescence = TRUE)
abundance_file <- system.file('extdata', 'abundance.tsv', package = 'BgeeCall')
tx_import <- run_tximport(myUserMetadata = user,
abundanceFile = abundance_file)
}
```

---

setAnnotationFromFile *Set annotation_object of one UserMetadata object*

Description

Method of the class UserMetadata. Set annotation_object of one UserMetadata object by providing the path to a fasta transcriptome file.

Usage

```
setAnnotationFromFile(userObject, annotationPath, annotationName)

## S4 method for signature 'UserMetadata,character,missing'
setAnnotationFromFile(userObject, annotationPath, annotationName)

## S4 method for signature 'UserMetadata,character,character'
setAnnotationFromFile(userObject, annotationPath, annotationName)
```

Arguments

userObject	The UserMetadata object
annotationPath	Absolute path to the annotation file
annotationName	(optional) Name of the annotation. Will be used to create folders.

Details

If no annotationName is provided the name of the annotation file will be used to create folders.

Value

An object of the class UserMetadata

Examples

```
{
# path to gtf annotation file
annotation_file <- system.file("extdata", "annotation.gtf", package = "BgeeCall")
user <- new("UserMetadata")
user <- setAnnotationFromFile(user, annotation_file,
                              "annotation_name")
}
```

---

setAnnotationFromObject

Set annotation_object of one UserMetadata object

Description

Method of the class UserMetadata. Set annotation_object of one UserMetadata object by using one GRanges object as input.

Usage

```
setAnnotationFromObject(userObject, annotationObject, annotationName)

## S4 method for signature 'UserMetadata,GRanges,character'
setAnnotationFromObject(userObject, annotationObject, annotationName = "")
```

Arguments

userObject	The UserMetadata object
annotationObject	object of thr GRanges S4 class
annotationName	(optional) Name of the annotation. Will be used to create folders.

Details

If no annotationName is provided the name of the file is used to create folders.

Value

An object of the class UserMetadata

Examples

```
{
user <- new("UserMetadata")
annotation_object <- rtracklayer::import(system.file("extdata",
"annotation.gtf", package = "BgeeCall"))
user <- setAnnotationFromObject(user, annotation_object,
                                "annotation_name")
}
```

---

setIntergenicRelease *'intergenic_release' Setter*

Description

Set value of the 'intergenic_release' slot

Usage

```
setIntergenicRelease(bgeeObject, intergenicRelease)

## S4 method for signature 'BgeeMetadata,character'
setIntergenicRelease(bgeeObject, intergenicRelease)
```

Arguments

bgeeObject	The BgeeMetadata object
intergenicRelease	character corresponding to the 'intergenic_release'

Value

An object of the class BgeeMetadata with new 'intergenic_release' value

Examples

```
{
bgee <- new("BgeeMetadata")
bgee <- setIntergenicRelease(bgee, "0.1")
}
```

`setOutputDir` *'output_dir' Setter*

Description

Set value of the 'output_dir' slot

Usage

```
setOutputDir(userObject, outputDir)

## S4 method for signature 'UserMetadata,character'
setOutputDir(userObject, outputDir)
```

Arguments

`userObject`	The UserMetadata object
`outputDir`	path to the directory wanted as 'output_dir'

Value

An object of the class UserMetadata with new 'output_dir' value

Examples

```
{
user <- new("UserMetadata")
user <- setOutputDir(user, getwd())
}
```

`setRNASeqLibPath` *'rnaseq_lib_path' Setter*

Description

Set value of the 'rnaseq_lib_path' slot

Usage

```
setRNASeqLibPath(userObject, rnaSeqLibPath)

## S4 method for signature 'UserMetadata,character'
setRNASeqLibPath(userObject, rnaSeqLibPath)
```

Arguments

userObject	The UserMetadata object
rnaSeqLibPath	path to the directory wanted as ‘rnaseq_lib_path‘

Value

An object of the class UserMetadata with new ‘rnaseq_lib_path‘ value

Examples

```
{
user <- new("UserMetadata")
user <- setRNASeqLibPath(user, getwd())
}
```

---

setRunIds *‘run_ids‘ Setter*

Description

Method of the class UserMetadata. Set run_ids of one UserMetadata object by providing the id of all wanted runs

Usage

```
setRunIds(userObject, runIds)

## S4 method for signature 'UserMetadata,character'
setRunIds(userObject, runIds)
```

Arguments

userObject	The UserMetadata object
runIds	id of all wanted runs

Value

An object of the class UserMetadata

Examples

```
{
user <- new("UserMetadata")
user <- setRunIds(user, c("RUN_1", "RUN_2"))
}
```

setSimpleArborescence *'simple_arborescence' Setter*

Description

Set value of the 'simple_arborescence' slot

Usage

```
setSimpleArborescence(userObject, simpleArborescence)

## S4 method for signature 'UserMetadata,logical'
setSimpleArborescence(userObject, simpleArborescence)
```

Arguments

userObject	The UserMetadata object
simpleArborescence	boolean defining if output files will be created a simple arborescence (TRUE) or not (FALSE)

Value

An object of the class UserMetadata with new 'simple_arborescence' value

Examples

```
{
user <- new("UserMetadata")
user <- setSimpleArborescence(user, FALSE)
}
```

setTranscriptomeFromFile *Set transcriptome_object of one UserMetadata object*

Description

Method of the class UserMetadata. Set transcriptome_object of one UserMetadata object by providing the path to a fasta transcriptome file.

Usage

```
setTranscriptomeFromFile(userObject, transcriptomePath, transcriptomeName)

## S4 method for signature 'UserMetadata,character,missing'
setTranscriptomeFromFile(userObject, transcriptomePath, transcriptomeName)

## S4 method for signature 'UserMetadata,character,character'
setTranscriptomeFromFile(userObject, transcriptomePath, transcriptomeName)
```

Arguments

userObject	The UserMetadata object
transcriptomePath	Absolute path to the transcriptome file
transcriptomeName	(optional) Name of the trancriptome. Will be used to create folders.

Details

If no transcriptomeName is provided the name of the transcriptome file will be used to create folders.

Value

An object of the class UserMetadata

Examples

```
{
transcriptome_path <- system.file("extdata", "transcriptome.fa", package = "BgeeCall")
user <- new("UserMetadata")
user <- setTranscriptomeFromFile(user, transcriptome_path,
                                 "transcriptome_name")
}
```

---

setTranscriptomeFromObject

Set transcriptome_object of one UserMetadata object

Description

Method of the class UserMetadata. Set transcriptome_object of one UserMetadata object by using one DNAStringSet object as input.

Usage

```
setTranscriptomeFromObject(userObject, transcriptomeObject, transcriptomeName)

## S4 method for signature 'UserMetadata,DNAStringSet,character'
setTranscriptomeFromObject(userObject, transcriptomeObject, transcriptomeName)
```

Arguments

userObject	UserMetadata object
transcriptomeObject	Object of the DNAStringSet S4 class
transcriptomeName	Name of the transcriptome. Will be used to create transcriptome folders.

Details

Please use a DNAStringSet object as input. This class is defined in the Biostrings package

Value

an object of UserMetadata

Examples

```
{
user <- new("UserMetadata")
transcriptome_object <- Biostrings::readDNAStringSet(
    system.file("extdata", "transcriptome.fa", package = "BgeeCall"))
user <- setTranscriptomeFromObject(user,
                transcriptome_object,
                "transcriptome_name")
}
```

`setWorkingPath` *'working_path' Setter*

Description

Set value of the 'working_path' slot

Usage

```
setWorkingPath(userObject, workingPath)

## S4 method for signature 'UserMetadata,character'
setWorkingPath(userObject, workingPath)
```

Arguments

`userObject`	The UserMetadata object
`workingPath`	path to the directory wanted as 'working_path'

Value

An object of the class UserMetadata with new 'working_path' value

Examples

```
{
user <- new("UserMetadata")
user <- setWorkingPath(user, getwd())
}
```

`UserMetadata-class` *UserMetadata S4 class*

Description

An S4 class containing all metadata that have to be provided by the user It is mandatory to edit 'species_id', 'rnaseq_lib_path', 'transcriptome_path', 'annotation_name', 'annotation_object' and potentialy 'run_ids' before using the package.

Slots

`species_id` The NCBI Taxon Id of the species

`run_ids` A vector of charater. Has to be provided only if a subset of runs present in UserMetadata@rnaseq_lib_path has to be run. If empty, all fastq files present in the rnaseq_lib_path will be considered as technical replicates and merged to run one transcript expression estimation analyse.

`reads_size` The size of the reads. If smaller than 'KallistoMetadata@read_size_kmer_threshold', an index with a kmer size of 15 bp will be used.

`rnaseq_lib_path` Path to the directory of the RNA-Seq library that contains fastq files. The extension of the fastq files name must be .fq, .fastq, .fq.gz, or .fastq.gz

`transcriptome_name` Name of the transcriptome used to generate arborescence of output repositories.

`transcriptome_object` Object containing transcriptome

`annotation_name` Name of the annotation used to generate arborescence of output repositories.

`annotation_object` Object containing annotations from GTF or GFF file

`working_path` Working directory. By default the working directory is defined with the 'getwd()' function.

`simple_arborescence` logical allowing to create a simple arborescence of directory. If 'TRUE' (default), all results will be on the same directory (working_path/intergenic_release/all_results/libraryId). Use 'FALSE' if you plan to generate expression calls for the same library using different transcriptomes or gene annotations, otherwise you will overwrite previous results. When 'FALSE' the path to result folder looks like : working_path/intergenic_release/speciesId/kallisto/transcriptome_name/annotat

`output_dir` (optional) Allows to manually define your output directory. By default the path to output directory is created automatically from the working_path (working_path/intergenic_release/all_results/libraryId

`verbose` logical allowing to use the verbose mode. TRUE by default.

Index