

# Package ‘BgeeCall’

November 5, 2024

**Type** Package

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

**Version** 1.23.0

**Date** 2021-05

**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 ( $\geq 3.6$ )

**Imports** GenomicFeatures, tximport, Biostrings, rtracklayer, biomaRt, jsonlite, methods, dplyr, data.table, sjmisc, grDevices, graphics, stats, utils, rslurm, rhdf5, txdbmaker

**License** GPL-3 + file LICENSE

**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** 7.1.1

**Encoding** UTF-8

**NeedsCompilation** no

**SystemRequirements** kallisto

**git\_url** <https://git.bioconductor.org/packages/BgeeCall>

**git\_branch** devel

**git\_last\_commit** 6e9ba9e

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.21

**Date/Publication** 2024-11-05

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

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

## Contents

AbundanceMetadata-class	3
BgeeCall	4
BgeeMetadata-class	5
create_kallisto_index	5
download_fasta_intergenic	7
download_kallisto	8
generate_calls_workflow	8
generate_presence_absence	10
generate_slurm_calls	12
generate_slurm_indexes	13
getIntergenicPrefix	15
getIntergenicRelease	16
getRunIds	16
getSimpleArborescence	17
getWorkingPath	18
get_summary_stats	18
KallistoMetadata-class	19
list_bgee_ref_intergenic_species	20
list_community_ref_intergenic_species	21
list_intergenic_release	21
merge_transcriptome_and_intergenic	22
merging_libraries	23
run_kallisto	24
run_tximport	25
setAnnotationFromFile	26
setAnnotationFromObject	27
setIntergenicRelease	28
setOutputDir	29
setRNASeqLibPath	30
setRunIds	30
setSimpleArborescence	31
setTranscriptomeFromFile	32
setTranscriptomeFromObject	33
setWorkingPath	34
UserMetadata-class	34

**Index**

**36**

---

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 instantiate 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_type` Defines the approach used to generate present/absent calls. default value is 'pValue', allowing calls to be generated using a pValue. Other possible values are 'intergenic' allowing to use a ratio of intergenic sequences considered as present as a threshold, or use qValue allowing calls to be generated from a qValue.

`cutoff` numeric value of the cutoff used to generate the present/absent calls. If value of the slot `cutoff_type` is 'pValue' this cutoff will correspond to the highest pValue allowing to define a gene as present. If value of the slot `cutoff_type` is 'intergenic' this cutoff will correspond to the proportion of intergenic present divided by proportion of protein coding present. If value of the slot `cutoff_type` is 'qValue' this cutoff will correspond to the highest qValue allowing to define a gene as present. The qValue is calculated based on the proportion of intergenic/(intergenic + genic) at each unique abundance value (TPM). The default value is 0.05. Be careful when modifying this value as it could have a huge impact on 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)

---

BgeeCall

*generate gene expression calls with BgeeCall*

---

## Description

BgeeCall allows to generate present/absent gene expression calls without using an arbitrary cut-off 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>).

## Details

These most important functions are :

- `generate_calls_workflow` : generate present/absent calls on a computer
- `generate_slurm_indexes` : generate kallisto indexes for a list of libraries on a cluster with slurm queuing system.
- `generate_slurm_calls` : generate present/absent calls for a list of libraries on a cluster with slurm queuing system. Indexes have to be generated first with the function ‘`generate_slurm_indexes`’
- `merging_libraries` : merge calls from different libraries corresponding to the same condition. Extremely useful if different libraries correspond to same condition (e.g. same anatomical entity from same species)

For more details please have a look at the vignette with the command `vignette("BgeeCall")`

**Author(s)**

Julien Wollbrett

**See Also**<https://github.com/BgeeDB/BgeeCall>

---

BgeeMetadata-class	<i>BgeeMetadata S4 class</i>
--------------------	------------------------------

---

**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

---

<code>create_kallisto_index</code>	<i>Create kallisto indexes.</i>
------------------------------------	---------------------------------

---

**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 will 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 that generates 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,
  checkTxVersion = FALSE
)

```

**Arguments**

abundanceMetadata	A Class AbundanceMetadata object (optional) allowing to tune your gene quantification abundance analyze
bgeeMetadata	A Class BgeeMetadata object (optional) allowing to choose the version of reference intergenic sequences
userMetadata	A Class UserMetadata object (optional). generate present/absent calls using slots of the UserMetadata class.
userDataFrame	a data.frame containing 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 between 4 and 8 columns : <ul style="list-style-type: none"> <li>• species_id : The ensembl species ID</li> <li>• 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</li> <li>• reads_size (optional) the size of the reads of the library (Default = 51) if the reads size is lower than 51 abundance quantification will be run from an index generated with a smaller kmer size</li> <li>• rnaseq_lib_path : path to RNA-Seq library directory</li> <li>• transcriptome_path : path to transcriptome file</li> <li>• annotation_path : path to annotation file</li> <li>• output_dir : (optional)root of the directory where results will be written</li> <li>• custom_intergenic_path : (optional) To use if the "custom" reference intergenic release has been selected. Provide the path to the reference intergenic file</li> </ul>
userFile	path to a tsv file containing between 4 and 8 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 <code>system.file('userMetadataTemplate.tsv', package = 'BgeeCall')</code>
checkTxVersion	boolean used to define if BgeeCall check rather transcript version should be removed. Default value is FALSE

**Value**

paths to the 5 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']])

# instantiate 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 corresponds to the last part of the generation of the expression calls workflow. It runs the last part of the workflow generating present/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 workflow

**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  
Sara Fonseca Costa

**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)  
  
#  
}
```

---

generate\_slurm\_calls *Generate present/absent calls on slurm queuing system*

---

### Description

This function is meant to be used with a cluster where the Slurm queuing system is installed. It processes all steps to generate present/absent calls at RNA-Seq library level. This function does not generate the kallisto indexes. If they are not already generated please run function “generate\_slurm\_indexes” first. Steps of present/absent gene expression calls generation are :

- Quantifying abundances of transcripts from RNA-Seq libraries
- Summarizing abundance at gene level
- generate present/absent expression calls

### Usage

```
generate_slurm_calls(
  kallistoMetadata = new("KallistoMetadata"),
  bgeeMetadata = new("BgeeMetadata"),
  userMetadata = new("UserMetadata"),
  userFile,
  submit_sh_template = NULL,
  slurm_options = NULL,
  rscript_path = NULL,
  modules = NULL,
  submit = TRUE,
  nodes = 10,
  checkTxVersion = FALSE
)
```

### Arguments

kallistoMetadata	A Reference Class KallistoMetadata object (optional) allowing to tune your gene quantification abundance analyze. If no object is provided a new one will be created with default values.
bgeeMetadata	A Reference Class BgeeMetadata object (optional) allowing to choose the version of reference intergenic sequences. If no object is provided a new one will be created with default values.
userMetadata	A Class UserMetadata object (optional). If no object is provided a new one will be created with default values.
userFile	Path to the file where each line corresponds to one abundance quantification to be run. The structure of the file is the same than the ‘userFile‘ used as input of the ‘generate_calls_workflow‘ function. A template of this file can be loaded with the command : “inputFile <- read.table(system.file("userMetadataTemplate.tsv", package = "BgeeCall"), header = TRUE)” It is important to keep the same column names.

submit_sh_template	A template of the bash script used to submit the jobs. By default the submission script provided by rslurm is used. Modify only if module dependancies have to be added (like kallisto or R)
slurm_options	A named list of options recognized by sbatch. More details in the documentation of the rslurm::slurm_apply function
rscript_path	The location of the Rscript command. If not specified, defaults to the location of Rscript within the R installation being run.
modules	A list of modules you want to load in the invironment. Should stay empty except if you need to load R and/or kallisto (e.g module add R)
submit	Whether or not to submit the job to the cluster with sbatch. Default value is TRUE
nodes	The (maximum) number of cluster nodes to spread the calculation over. slurm_apply automatically divides params in chunks of approximately equal size to send to each node. Less nodes are allocated if the parameter set is too small to use all CPUs on the requested nodes. By default this number is 10.
checkTxVersion	boolean used to define if BgeeCall check rather transcript version should be removed. Default value is FALSE

**Value**

generate calls

**Examples**

```
## Not run:
# use function with all default values
userFile <- "/path/to/userList.tsv"
sjobs <- generate_slurm_calls(userFile = userFile)

## End(Not run)
```

---

generate\_slurm\_indexes

*Generate all indexes for the abundance quantification step*

---

**Description**

Check all unique lines of the input file to check which indexes have to be generated before running all abundance quantification. This function is meant to be used with a cluster where the Slurm queuing system is installed. This step has to be run before the quantification otherwise indexes will be created for each abundance quantification. This will slow down the abundance quantification and can generate errors when writing the same file at the same time from different nodes. This function also generate tx2gene and gene2biotype mapping files.

**Usage**

```

generate_slurm_indexes(
  kallistoMetadata = new("KallistoMetadata"),
  bgeeMetadata = new("BgeeMetadata"),
  userMetadata = new("UserMetadata"),
  userFile,
  submit_sh_template = NULL,
  slurm_options = NULL,
  rscript_path = NULL,
  modules = NULL,
  submit = TRUE,
  nodes = 10
)

```

**Arguments**

kallistoMetadata	A Reference Class KallistoMetadata object (optional) allowing to tune your gene quantification abundance analyze. If no object is provided a new one will be created with default values.
bgeeMetadata	A Reference Class BgeeMetadata object (optional) allowing to choose the version of reference intergenic sequences. If no object is provided a new one will be created with default values.
userMetadata	A Class UserMetadata object (optional). If no object is provided a new one will be created with default values.
userFile	Path to the file where each line corresponds to one abundance quantification to be run. The structure of the file is the same than the 'userFile' used as input of the 'generate_calls_workflow' function. A template of this file can be loaded with the command : <code>“inputFile &lt;- read.table(system.file("userMetadataTemplate.tsv", package = "BgeeCall"), header = TRUE)“</code> It is important to keep the same column names.
submit_sh_template	A template of the bash script used to submit the jobs. By default the submission script provided by rslurm is used. Modify only if module dependencies have to be added (like kallisto or R)
slurm_options	A named list of options recognized by sbatch. More details in the documentation of the rslurm::slurm_apply function
rscript_path	The location of the Rscript command. If not specified, defaults to the location of Rscript within the R installation being run.
modules	A list of modules you want to load in the environment. Should stay empty except if you need to load R and/or kallisto (e.g module add R)
submit	Whether or not to submit the job to the cluster with sbatch. Default value is TRUE
nodes	The (maximum) number of cluster nodes to spread the calculation over. slurm_apply automatically divides params in chunks of approximately equal size to send to each node. Less nodes are allocated if the parameter set is too small to use all CPUs on the requested nodes. By default this number is 10.

**Value**

generate index files

**Examples**

```
## Not run:  
# use function with all default values  
userFile <- "/path/to/userList.tsv"  
sjobs <- generate_slurm_indexes(userFile = userFile)  
  
## End(Not run)
```

---

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)
}
```

---

get\_summary\_stats            *Gather statistical information*

---

**Description**

Collect the statistics provided by the gene\_cutoff\_info\_file from each individual library, in order to generate a global summary file.

**Usage**

```
get_summary_stats(userFile, outDir)
```

**Arguments**

userFile            A data frame containing all information of each library  
outDir            Output directory where the generated file should be saved

**Value**

A tsv file

**Author(s)**

Sara Fonseca Costa

---

KallistoMetadata-class

*KallistoMetadata S4 class*

---

**Description**

An S4 class that is the descendant 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 stored 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 quantification 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
- `genomeVersion` version of the genome used to generate the 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)
}
```

---

merging\_libraries

*Calls of expression in combined libraries*


---

**Description**

Merging/combine libraries based in a condition specified by the user. The merging can be done using the p-values of the libraries, by applying the BH method, or using the q-values of the libraries using the `fdr_inverse` method.

**Usage**

```
merging_libraries(
  userFile = NULL,
  approach = "BH",
  condition = "species_id",
  cutoff = 0.05,
  outDir = NULL
)
```

**Arguments**

<code>userFile</code>	A file provided by the user with correspondent conditions
<code>approach</code>	Approach used to do the merging of libraries
<code>condition</code>	Condition/s where the merging should be done
<code>cutoff</code>	Cutoff that should be applied to call Present/Absent genes
<code>outDir</code>	Directory where the output files should be saved

**Value**

A dataframe containing the minimum quantitative value (p-value or q-value) and the calls to each gene id for the referent condition.

**Author(s)**

Sara Fonseca Costa

**Examples**

```
## Not run:
callsMerging_species <- merging_libraries(userFile = 'PATH_USER_FILE', approach = 'BH',
condition = 'species_id', cutoff = 0.05, outDir = 'PATH_OUTPUT')
callsMerging_species_sex <- merging_libraries(userFile = 'PATH_USER_FILE', approach = 'fdr_inverse',
condition = c(species_id, sex), cutoff = 0.01, outDir = 'PATH_OUTPUT')
callsMerging_all <- merging_libraries(userFile = 'PATH_USER_FILE', approach = 'fdr_inverse',
condition = c(species_id, anatEntity, devStage, sex, strain), cutoff = 0.05, outDir = 'PATH_OUTPUT')

## End(Not run)
```

---

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 from 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	<i>'output_dir' Setter</i>
--------------	----------------------------

---

**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)  
}
```





---

`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 potentially 'run\_ids' before using the package.

**Slots**

- `species_id` The NCBI Taxon Id of the species
- `run_ids` A vector of character. 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 analysis.
- `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 file 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.
- `gtf_source` The source name from where the gtf file comes from. By default is `ensembl`.
- `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/annotation_name`
- `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.
- `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`.
- `encrypted_pattern` Allows to manage encrypted libraries. If a fastq file with the suffix `.enc` is found for a run, this slot will allow to use a string pattern to decrypt it. . This `encrypted_pattern` needs to contain the string `FASTQ_PATH` that will be transformed to the actual path to the fastq file.

# Index

- \* **RNA-Seq**,
  - BgeeCall, 4
- \* **abundance**
  - BgeeCall, 4
- \* **calls**
  - BgeeCall, 4
- \* **present/absent**
  - BgeeCall, 4
- \* **quantification**,
  - BgeeCall, 4
  
- AbundanceMetadata
  - (AbundanceMetadata-class), 3
- AbundanceMetadata-class, 3
  
- BgeeCall, 4
- BgeeMetadata (BgeeMetadata-class), 5
- BgeeMetadata-class, 5
  
- create\_kallisto\_index, 5
  
- download\_fasta\_intergenic, 7
- download\_kallisto, 8
  
- generate\_calls\_workflow, 8
- generate\_presence\_absence, 10
- generate\_slurm\_calls, 12
- generate\_slurm\_indexes, 13
- get\_summary\_stats, 18
- getIntergenicPrefix, 15
- getIntergenicPrefix, bgeeMetadata
  - (getIntergenicPrefix), 15
- getIntergenicPrefix, BgeeMetadata-method
  - (getIntergenicPrefix), 15
- getIntergenicRelease, 16
- getIntergenicRelease, bgeeMetadata
  - (getIntergenicRelease), 16
- getIntergenicRelease, BgeeMetadata-method
  - (getIntergenicRelease), 16
- getRunIds, 16
- getRunIds, userMetadata (getRunIds), 16
  
- getRunIds, UserMetadata-method
  - (getRunIds), 16
- getSimpleArborescence, 17
- getSimpleArborescence, userMetadata
  - (getSimpleArborescence), 17
- getSimpleArborescence, UserMetadata-method
  - (getSimpleArborescence), 17
- getWorkingPath, 18
- getWorkingPath, userMetadata
  - (getWorkingPath), 18
- getWorkingPath, UserMetadata-method
  - (getWorkingPath), 18
  
- KallistoMetadata
  - (KallistoMetadata-class), 19
- KallistoMetadata-class, 19
  
- list\_bgee\_ref\_intergenic\_species, 20
- list\_community\_ref\_intergenic\_species,
  - 21
- list\_intergenic\_release, 21
  
- merge\_transcriptome\_and\_intergenic, 22
- merging\_libraries, 23
  
- run\_kallisto, 24
- run\_tximport, 25
  
- setAnnotationFromFile, 26
- setAnnotationFromFile, userMetadata, character, character
  - (setAnnotationFromFile), 26
- setAnnotationFromFile, UserMetadata, character, character-method
  - (setAnnotationFromFile), 26
- setAnnotationFromFile, UserMetadata, character, missing-method
  - (setAnnotationFromFile), 26
- setAnnotationFromObject, 27
- setAnnotationFromObject, userMetadata, GRanges, character
  - (setAnnotationFromObject), 27
- setAnnotationFromObject, UserMetadata, GRanges, character-method
  - (setAnnotationFromObject), 27
- setIntergenicRelease, 28

setIntergenicRelease,bgeeMetadata,character  
(setIntergenicRelease), 28

setIntergenicRelease,BgeeMetadata,character-method  
(setIntergenicRelease), 28

setOutputDir, 29

setOutputDir,userMetadata,character  
(setOutputDir), 29

setOutputDir,UserMetadata,character-method  
(setOutputDir), 29

setRNASeqLibPath, 30

setRNASeqLibPath,userMetadata,character  
(setRNASeqLibPath), 30

setRNASeqLibPath,UserMetadata,character-method  
(setRNASeqLibPath), 30

setRunIds, 30

setRunIds,userMetadata,character  
(setRunIds), 30

setRunIds,UserMetadata,character-method  
(setRunIds), 30

setSimpleArborescence, 31

setSimpleArborescence,userMetadata,logical  
(setSimpleArborescence), 31

setSimpleArborescence,UserMetadata,logical-method  
(setSimpleArborescence), 31

setTranscriptomeFromFile, 32

setTranscriptomeFromFile,userMetadata,character,character  
(setTranscriptomeFromFile), 32

setTranscriptomeFromFile,UserMetadata,character,character-method  
(setTranscriptomeFromFile), 32

setTranscriptomeFromFile,userMetadata,character,missing  
(setTranscriptomeFromFile), 32

setTranscriptomeFromFile,UserMetadata,character,missing-method  
(setTranscriptomeFromFile), 32

setTranscriptomeFromObject, 33

setTranscriptomeFromObject,userMetadata,DNAStringSet,character  
(setTranscriptomeFromObject),  
33

setTranscriptomeFromObject,UserMetadata,DNAStringSet,character-method  
(setTranscriptomeFromObject),  
33

setWorkingPath, 34

setWorkingPath,userMetadata,character  
(setWorkingPath), 34

setWorkingPath,UserMetadata,character-method  
(setWorkingPath), 34

UserMetadata (UserMetadata-class), 34

UserMetadata-class, 34