Package ‘DeepBlueR’

November 20, 2016

Title DeepBlueR
Type Package
Description Accessing the DeepBlue Epigenetics Data Server through R.
Version 1.0.4
Author Felipe Albrecht, Markus List
Maintainer Felipe Albrecht <felipe.albrecht@mpi-inf.mpg.de>, Markus List <markus.list@mpi-inf.mpg.de>
License GPL (>=2.0)
Imports GenomicRanges, data.table, stringr, dplyr, methods, rjson, utils, R.utils
Depends R (>= 3.3), XML, Rcurl
BuildVignettes TRUE
RoxygenNote 5.0.1
Suggests knitr, rmarkdown, LOLA, Gviz, ggplot2, tidyr
VignetteBuilder knitr
biocViews DataImport, DataRepresentation, ThirdPartyClient, GeneRegulation, GenomeAnnotation, CpGIsland, DNAMethylation, Epigenetics, Annotation, Preprocessing

NeedsCompilation no

R topics documented:

DeepBlueCommand-class ....................................................... 3
deepleague_aggregate .......................................................... 3
deepleague_batch_export_results .......................................... 4
deepleague_cancel_request .................................................. 5
deepleague_chromosomes ...................................................... 6
deepleague_collection_experiments_count ................................ 6
deepleague_commands .......................................................... 7
deepleague_count_regions .................................................... 8
deepleague_coverage ............................................................ 9
deepleague_diff ................................................................. 9
deepleague_download_request_data ....................................... 10
R topics documented:

- deepblue_download_request_data
- DeepBlueCommand
- deepblue_echo
- deepblue_extend
- deepblue_extract_ids
- deepblue_extract_names
- deepblue_faceting_experiments
- deepblue_filter_regions
- deepblue_flank
- deepblue_get_biosource_children
- deepblue_get_biosource_parents
- deepblue_get_biosource_related
- deepblue_get_biosource_synonyms
- deepblue_get_experiments_by_query
- deepblue_get_regions
- deepblue_get_request_data
- deepblue_info
- deepblue_input_regions
- deepblue_intersection
- deepblue_is_biosource
- deepblue_list_annotations
- deepblue_list_biosources
- deepblue_list_column_types
- deepblue_list_epigenetic_marks
- deepblue_list_experiments
- deepblue_list_expressions
- deepblue_list_genes
- deepblue_list_gene_models
- deepblue_list_genomes
- deepblue_list_in_use
- deepblue_list_projects
- deepblue_list_recent_experiments
- deepblue_list_requests
- deepblue_list_samples
- deepblue_list_similar_biosources
- deepblue_list_similar_epigenetic_marks
- deepblue_list_similar_experiments
- deepblue_list_similar_genomes
- deepblue_list_similar_projects
- deepblue_list_similar_techniques
- deepblue_list_techniques
- deepblue_merge_queries
- deepblue_name_to_id
- deepblue_preview_experiment
- deepblue_query_cache
- deepblue_query_experiment_type
- deepblue_score_matrix
- deepblue_search
- deepblue_select_annotations
- deepblue_select_column
- deepblue_select_experiments
- deepblue_select_expressions
- deepblue_select_genes
DeepBlueCommand-class

Description
An S4 class returned when calling a DeepBlue-R function. It holds information about the original call, the query / request status, previous commands, the user_key, and results in GRanges format once a request is downloaded.

Arguments
- call
- language
- status
- character
- query_id
- character
- previous_commands
  - list
- user_key
  - character
- result
  - GRanges

Value
class for managing DeepBlue commands

---

deepblue_aggregate  aggregate

Description
Summarize the data_id content using the regions specified in ranges_id as boundaries. Use the fields @AGG.MIN, @AGG.MAX, @AGG.SUM, @AGG.MEDIAN, @AGG.MEAN, @AGG.VAR, @AGG.SD, @AGG.COUNT in `get_regions` command `format` parameter to retrieve the computed values minimum, maximum, median, mean, variance, standard deviation and number of regions, respectively.

Usage
depthblue_aggregate(data_id = NULL, ranges_id = NULL, column = NULL, user_key = deepblue_USER_KEY)

Arguments
- data_id
  - A string (id of the query with the data)
- ranges_id
  - A string (id of the query with the regions range)
- column
  - A string (name of the column that will be used in the aggregation)
- user_key
  - A string (users token key)
Value
regions - A string (query id of this aggregation operation)

See Also
Other Operating on the data regions: deepblue_count_regions, deepblue_coverage, deepblue_extend, deepblue_filter_regions, deepblue_flank, deepblue_get_experiments_by_query, deepblue_get_regions, deepblue_input_regions, deepblue_intersection, deepblue_merge_queries, deepblue_query_cache, deepblue_query_experiment_type, deepblue_score_matrix, deepblue_select Annotations, deepblue_select_experiments, deepblue_select_regions, deepblue_tiling_regions

Examples
annotation_id = deepblue_select_annotations(
    annotation_name="CpG Islands",
    genome="hg19", chromosome="chr1")
data_id = deepblue_select_experiments(
    experiment_name="E002-H3K9ac.narrowPeak.bed")
deepblue_aggregate(
    data_id = data_id,
    ranges_id=annotation_id,
    column = "SCORE")

depthblue_batch_export_results

Description
Write results from DeepBlue to disk as they become available

Usage
depthblue_batch_export_results(requests, target.directory = NULL,
suffix = "result", prefix = "DeepBlue", sleep.time = 1,
user_key = depthblue_USER_KEY)

Arguments
requests A list of request objects
target.directory Where the results should be saved
suffix File names suffix
prefix File names prefix
sleep.time How long this function will wait after the requests verification
user_key A string used to authenticate the user

Value
A list containing the requests IDs data
Examples

data_id = deepblue_select_experiments(
experiment_name="E002-H3K9ac.narrowPeak.bed", chromosome="chr1")
request_id = deepblue_get_regions(query_id =data_id,
    output_format = "CHROMOSOME,START,END")
request_data = deepblue_batch_export_results(list(request_id))

---

depblue_cancel_request

cancel_request

Description

Stop, cancel, and remove request data. The request processed data is remove if its processing was finished.

Usage

depblue_cancel_request(id = NULL, user_key = deepblue_USER_KEY)

Arguments

id - A string (Request ID to be canceled, stopped or removed.)
user_key - A string (users token key)

Value

id - A string (ID of the canceled request)

See Also

Other Commands for all types of data: deepblue_info, deepblue_is_biosource, deepblue_list_in_use, deepblue_name_to_id, deepblue_search

Examples

depblue_cancel_request(id = "r12345")
deepblue_chromosomes  chromosomes

Description
List the chromosomes of a given Genome.

Usage
deepblue_chromosomes(genome = NULL, user_key = deepblue_USER_KEY)

Arguments
- genome - A string (the target genome)
- user_key - A string (users token key)

Value
- chromosomes - A array (A list containing all chromosomes, with theirs names and sizes)

See Also
Other Inserting and listing genomes: deepblue_list_genomes, deepblue_list_similar_genomes

Examples
deepblue_chromosomes(genome = "g1")

deepblue_collection_experiments_count  collection_experiments_count

Description
Count the number of experiments that match the selection criteria in each term of the selected controlled_vocabulary. The selection can be achieved through specifying a list of BioSources, experimental Techniques, Epigenetic Marks, Samples or Projects.

Usage
deepblue_collection_experiments_count(controlled_vocabulary = NULL, genome = NULL, type = NULL, epigenetic_mark = NULL, biosource = NULL, sample = NULL, technique = NULL, project = NULL, user_key = deepblue_USER_KEY)
Arguments

- controlled_vocabulary: A string (controlled vocabulary name)
- genome: A string or a vector of string (the target genome)
- type: A string or a vector of string (type of the experiment: peaks or signal)
- epigenetic_mark: A string or a vector of string (name(s) of selected epigenetic mark(s))
- biosource: A string or a vector of string (name(s) of selected biosource(s))
- sample: A string or a vector of string (id(s) of selected sample(s))
- technique: A string or a vector of string (name(s) of selected technique(s))
- project: A string or a vector of string (name(s) of selected projects)
- user_key: A string (users token key)

Value

terms: A array (controlled_vocabulary terms with count)

See Also

Other Inserting and listing experiments: deepblue_faceting_experiments, deepblue_list_experiments, deepblue_list_recent_experiments, deepblue_list_similar_experiments, deepblue_preview_experiment

Examples

```
deeble_collection_experiments_count(  
    controlled_vocabulary="epigenetic_marks",  
    genome = "hg19", type = "peaks",  
    biosource = "blood")
```
Examples

```
depthblue_count_regions()
```

---

```
depthblue_count_regions

count_regions
```

### Description

Return the number of genomic regions present in the query.

### Usage

```
depthblue_count_regions(query_id = NULL, user_key = depthblue_USER_KEY)
```

#### Arguments

- **query_id**
  - A string (Query ID)
- **user_key**
  - A string (users token key)

#### Value

```
request_id - A string (Request ID - Use it to retrieve the result with info() and get_request_data())
```

### See Also

Other Operating on the data regions: `depthblue_aggregate`, `depthblue_coverage`, `depthblue_extend`, `depthblue_filter_regions`, `depthblue_flank`, `depthblue_get_experiments_by_query`, `depthblue_get_regions`, `depthblue_input_regions`, `depthblue_intersection`, `depthblue_merge_queries`, `depthblue_query_cache`, `depthblue_query_experiment_type`, `depthblue_score_matrix`, `depthblue_select_annotations`, `depthblue_select_experiments`, `depthblue_select_regions`, `depthblue_tiling_regions`

### Examples

```
data_id = depthblue_select_experiments(
    experiment_name="E002-H3K9ac.narrowPeak.bed")
depthblue_count_regions(query_id = data_id)
```
**deepblue_coverage**

---

**Description**

Send a request to count the number of regions in the result of the given query.

**Usage**

```r
deepblue_coverage(query_id = NULL, genome = NULL,
                   user_key = deepblue_USER_KEY)
```

**Arguments**

- **query_id** - A string (Query ID)
- **genome** - A string (Genome where the coverage will be calculated to)
- **user_key** - A string (users token key)

**Value**

- **request_id** - A string (Request ID - Use it to retrieve the result with info() and get_request_data())

**See Also**

Other Operating on the data regions: `deepblue_aggregate`, `deepblue_count_regions`, `deepblue_extend`, `deepblue_filter_regions`, `deepblue_flank`, `deepblue_get_experiments_by_query`, `deepblue_get_regions`, `deepblue_input_regions`, `deepblue_intersection`, `deepblue_merge_queries`, `deepblue_query_cache`, `deepblue_query_experiment_type`, `deepblue_score_matrix`, `deepblue_select_annotations`, `deepblue_select_experiments`, `deepblue_select_regions`, `deepblue_tiling_regions`

**Examples**

```r
data_id = deepblue_select_experiments(
    experiment_name="E002-H3K9ac.narrowPeak.bed")
deepblue_coverage(query_id = data_id, genome="hg19")
```

---

**deepblue_diff**

---

**Description**

A utility command that creates a diff view of info for two DeepBlue ids

**Usage**

```r
deepblue_diff(id1, id2, user_key = deepblue_USER_KEY)
```
Description

Returns the requested data as the expected type object. Expects two input parameters: Request information and user key. It depends on outputs from several functions, namely: `deepblue_get_request_data`, `convert_to_df`, and `convert_to_grange`.

Usage

```
depthblue_download_request_data(request_id, user_key = depthblue_USER_KEY)
```

Arguments

- `request_id`: Id of the request that will be downloaded
- `user_key`: A string

Value

- `grange_regions`: Final output in GRanges format or as data frame

Examples

```
data_id = depthblue_select_experiments(
    experiment_name="E002-H3K9ac.narrowPeak.bed", chromosome="chr1")
request_id = depthblue_get_regions(query_id = data_id,
    output_format = "CHROMOSOME,START,END")
request_data = depthblue_download_request_data(request_id)
```
### deepblue_download_request_data, DeepBlueCommand-method

**deepblue_download_request_data**

**Description**
Returns the requested data as the expected type object. Expects two input parameters: Request information and user key. It depends on outputs from several functions, namely; deepblue_get_request_data, convert_to_df, and convert_to_grange.

**Usage**

```r
## S4 method for signature 'DeepBlueCommand'
deepblue_download_request_data(request_id)
```

**Arguments**

- `request_id` - DeepBlueCommand object

**Value**

grange_regions - Final output in GRanges format

### deepblue_echo, echo

**Description**

Greet the user with the DeepBlue version.

**Usage**

deepblue_echo(user_key = deepblue_USER_KEY)

**Arguments**

- `user_key` - A string (users token key)

**Value**

message - A string (echo message including version)

**See Also**

Other Checking DeepBlue status: deepblue_commands

**Examples**

deepblue_echo(user_key = "anonymous_key")
**Description**

Extend the genomic regions included in the query. It is possible to extend downstream, upstream or in both directions.

**Usage**

```r
deepblue_extend(query_id = NULL, length = NULL, direction = NULL,
                 use_strand = NULL, user_key = deepblue_USER_KEY)
```

**Arguments**

- `query_id` - A string (id of the query that contains the regions)
- `length` - A int (The new region length)
- `direction` - A string (The direction that the region will be extended: 'BACKWARD', 'FORWARD', 'BOTH'. (Empty value will be used for both direction.)
- `use_strand` - A boolean (Use the region column STRAND to define the region direction)
- `user_key` - A string (users token key)

**Value**

- `id` - A string (id of the new query)

**See Also**

Other Operating on the data regions: `deepblue_aggregate`, `deepblue_count_regions`, `deepblue_coverage`, `deepblue_filter_regions`, `deepblue_flank`, `deepblue_get_experiments_by_query`, `deepblue_get_regions`, `deepblue_input_regions`, `deepblue_intersection`, `deepblue_merge_queries`, `deepblue_query_cache`, `deepblue_query_experiment_type`, `deepblue_score_matrix`, `deepblue_select_annotations`, `deepblue_select_experiments`, `deepblue_select_regions`, `deepblue_tiling_regions`

**Examples**

```r
annotation_id = deepblue_select_annotations(
    annotation_name="CpG Islands",
    genome="hg19", chromosome="chr1")
deepblue_extend(query_id = annotation_id,
                length = 2000, direction = "BOTH",
                use_strand = TRUE)
```
**deepblue_extract_ids**  

**Description**  
A utility command that returns a list of IDs extracted from a data frame of ID and names.

**Usage**  
```r  
deepblue_extract_ids(df = NULL)  
```

**Arguments**
- `df` - A array of IDs and names

**Value**
- `ids` - A vector containing the extracted IDs

**See Also**
Other Utilities for connecting operations: `deepblue_extract_names`

**Examples**
```r  
deepblue_extract_ids(  
  df = data.frame(id = c("a124", "a1235"),  
                    name = c("Annotation 1", "Annotation 2")))  
```

---

**deepblue_extract_names**  

**Description**  
A utility command that returns a list of names extracted from a list of ID and names.

**Usage**  
```r  
deepblue_extract_names(df = NULL)  
```

**Arguments**
- `df` - A array of IDs and Names

**Value**
- `names` - A vector containing the extracted names
See Also

Other Utilities for connecting operations: `deepblue_extract_ids`

Examples

```r
deepblue_extract_ids(
  df = data.frame(id = c("a124", "a1235"),
                   name = c("Annotation 1", "Annotation 2")))
```

```r
deepblue_faceting_experiments
  faceting_experiments
```

Description

Summarize the controlled_vocabulary fields, from experiments that match the selection criteria. It is similar to the `collection_experiments_count` command, but this command return the summarization for all controlled_vocabulary terms.

Usage

```r
deepblue_faceting_experiments(genome = NULL, type = NULL,
                                epigenetic_mark = NULL, biosource = NULL, sample = NULL,
                                technique = NULL, project = NULL, user_key = deepblue_USER_KEY)
```

Arguments

genome  - A string or a vector of string (the target genome)
type    - A string or a vector of string (type of the experiment: peaks or signal)
epigenetic_mark - A string or a vector of string (name(s) of selected epigenetic mark(s))
biosource - A string or a vector of string (name(s) of selected biosource(s))
sample   - A string or a vector of string (id(s) of selected sample(s))
technique - A string or a vector of string (name(s) of selected technique(s))
project  - A string or a vector of string (name(s) of selected projects)
user_key - A string (users token key)

Value

faceting - A struct (Map with the mandatory fields of the experiments metadata, where each contains a list of terms that appears.)

See Also

Other Inserting and listing experiments: `deepblue_collection_experiments_count`, `deepblue_list_experiments`, `deepblue_list_recent_experiments`, `deepblue_list_similar_experiments`, `deepblue_preview_experiment`
**Examples**

```r
depthblue_faceting_experiments(genome = "hg19",
    type = "peaks",
    biosource = "blood")
```

```r
depthblue_filter_regions
```

**Description**

Filter the genomic regions by their content.

**Usage**

```r
depthblue_filter_regions(query_id = NULL, field = NULL, operation = NULL,
    value = NULL, type = NULL, user_key = depthblue_USER_KEY)
```

**Arguments**

- **query_id** - A string (id of the query to be filtered)
- **field** - A string (field that is filtered by)
- **operation** - A string (operation used for filtering. For 'string' must be '== or '!=' and for
  'number' must be one of these: ==,!=,>,>=,<,<=)
- **value** - A string (value the operator is applied to)
- **type** - A string (type of the value: 'number' or 'string' )
- **user_key** - A string (users token key)

**Value**

id - A string (id of filtered query)

**See Also**

Other Operating on the data regions: `depthblue_aggregate, depthblue_count_regions, depthblue_coverage, depthblue_extend, depthblue_flank, depthblue_get_experiments_by_query, depthblue_get_regions, depthblue_input_regions, depthblue_intersection, depthblue_merge_queries, depthblue_query_cache, depthblue_query_experiment_type, depthblue_score_matrix, depthblue_select_annotations, depthblue_select_experiments, depthblue_select_regions, depthblue_tiling_regions`

**Examples**

```r
depthblue_filter_regions(query_id = "q12345",
    field = "VALUE",
    operation = ">",
    value = "100",
    type = "number",
    user_key = "anonymous_key")
```
Description

Create a set of genomic regions that flank the query regions. The original regions are removed from the query. Use the merge command to combine flanking regions with the original query.

Usage

```r
deepblue_flank(query_id = NULL, start = NULL, length = NULL,
               use_strand = NULL, user_key = deepblue_USER_KEY)
```

Arguments

- `query_id` - A string (id of the query that contains the regions)
- `start` - A int (Number of base pairs after the end of the region. Use a negative number to denote the number of base pairs before the start of the region.)
- `length` - A int (The new region length)
- `use_strand` - A boolean (Use the region column STRAND to define the region direction)
- `user_key` - A string (users token key)

Value

- `id` - A string (id of the new query)

See Also

Other Operating on the data regions: `deepblue_aggregate`, `deepblue_count_regions`, `deepblue_coverage`, `deepblue_extend`, `deepblue_filter_regions`, `deepblue_get_experiments_by_query`, `deepblue_get_regions`, `deepblue_input_regions`, `deepblue_intersection`, `deepblue_merge_queries`, `deepblue_query_cache`, `deepblue_query_experiment_type`, `deepblue_score_matrix`, `deepblue_select_annotations`, `deepblue_select_experiments`, `deepblue_select_regions`, `deepblue_tiling_regions`

Examples

```r
annotation_id = deepblue_select_annotations(
    annotation_name="CpG Islands",
    genome="hg19", chromosome="chr1")
deepblue_flank(query_id = annotation_id,
               start = 0, length = 2000,
               use_strand = TRUE)
```
**deepblue_get_biosource_children**

---

**Description**

A BioSource refers to a term describing the origin of a given sample, such as a tissue or cell line. These form a hierarchy in which children of a BioSource term can be fetched with this command. Children terms are more specific terms that are defined in the imported ontologies.

**Usage**

```r
deepblue_get_biosource_children(biosource = NULL,
                                user_key = deepblue_USER_KEY)
```

**Arguments**

- **biosource** - A string (biosource name)
- **user_key** - A string (users token key)

**Value**

- **biosources** - A array (related biosources)

**See Also**

- Other Set the relationship between different biosources: `deepblue_get_biosource_parents`, `deepblue_get_biosource_related`, `deepblue_get_biosource_synonyms`

**Examples**

```r
deepblue_get_biosource_children(biosource = "Blood")
```

---

**deepblue_get_biosource_parents**

---

**Description**

A BioSource refers to a term describing the origin of a given sample, such as a tissue or cell line. These form a hierarchy in which the parent of a BioSource term can be fetched with this command. Parent terms are more generic terms that are defined in the imported ontologies.

**Usage**

```r
deepblue_get_biosource_parents(biosource = NULL,
                                user_key = deepblue_USER_KEY)
```
**Arguments**

- **biosource** - A string (biosource name)
- **user_key** - A string (users token key)

**Value**

- **biosources** - A array (parents biosources)

**See Also**

Other Set the relationship between different biosources: `deepblue_get_biosource_children`, `deepblue_get_biosource_related`, `deepblue_get_biosource_synonyms`

**Examples**

```python
deepblue_get_biosource_related(biosource = "Blood")
```

---

**Description**

A BioSource refers to a term describing the origin of a given sample, such as a tissue or cell line. These form a hierarchy in which the children of a BioSource term and its synonyms can be fetched with this command. Children terms are more specific terms that are defined in the imported ontologies. Synonyms are different aliases for the same biosource.

**Usage**

```python
deepblue_get_biosource_related(biosource = NULL,
   user_key = deepblue_USER_KEY)
```

**Arguments**

- **biosource** - A string (biosource name)
- **user_key** - A string (users token key)

**Value**

- **biosources** - A array (related biosources)

**See Also**

Other Set the relationship between different biosources: `deepblue_get_biosource_children`, `deepblue_get_biosource_related`, `deepblue_get_biosource_synonyms`

**Examples**

```python
deepblue_get_biosource_related(biosource = "Blood")
```
**deepblue_get_biosource_synonyms**

**get_biosource_synonyms**

**Description**

Obtain the synonyms of the specified biosource. Synonyms are different aliases for the same biosource. A BioSource refers to a term describing the origin of a given sample, such as a tissue or cell line.

**Usage**

```r
deepblue_get_biosource_synonyms(biosource = NULL,
user_key = deepblue_USER_KEY)
```

**Arguments**

- `biosource` - A string (name of the biosource)
- `user_key` - A string (users token key)

**Value**

- `synonyms` - A array (synonyms of the biosource)

**See Also**

Other Set the relationship between different biosources: `deepblue_get_biosource_children`, `deepblue_get_biosource_parents`, `deepblue_get_biosource_related`

**Examples**

```r
depthblue_get_biosource_synonyms(biosource = "prostate gland")
```

---

**deepblue_get_experiments_by_query**

**get_experiments_by_query**

**Description**

List the experiments and annotations that have at least one genomic region in the final query result.

**Usage**

```r
deepblue_get_experiments_by_query(query_id = NULL,
user_key = deepblue_USER_KEY)
```

**Arguments**

- `query_id` - A string (id of the query)
- `user_key` - A string (users token key)
Value

experiments - A array (List containing experiments names and ids)

See Also

Other Operating on the data regions: deepblue_aggregate, deepblue_count_regions, deepblue_coverage,
depblue_extend, deepblue_filter_regions, deepblue_flank, deepblue_get_regions, deepblue_input_regions,
depblue_intersection, deepblue_merge_queries, deepblue_query_cache, deepblue_query_experiment_type,
depblue_score_matrix, deepblue_select_annotations, deepblue_select_experiments, deepblue_select_regions,
depblue_tiling_regions

Examples

depblue_get_experiments_by_query(query_id = "q12345")

---

depblue_get_regions  get_regions

Description

Trigger the processing of the query’s genomic regions. The output is a column based format with
columns as defined in the 'output_format' parameter. Use the command 'info' for verifying the
processing status. The 'get_request_data' command is used to download the regions using the
programmatic interface. Alternatively, results can be downloaded using the URL: http://deepblue.mpi-inf.mpg.de/download?r_id=<request_id>&key=<user_key>.

Usage

deepblue_get_regions(query_id = NULL, output_format = NULL,
user_key = deepblue_USER_KEY)

Arguments

query_id - A string (Query ID)
output_format - A string (Output format)
user_key - A string (users token key)

Value

request_id - A string (Request ID - Use it to retrieve the result with info() and get_request_data())

See Also

Other Operating on the data regions: deepblue_aggregate, deepblue_count_regions, deepblue_coverage,
depblue_extend, deepblue_filter_regions, deepblue_flank, deepblue_get_experiments_by_query,
depblue_input_regions, deepblue_intersection, deepblue_merge_queries, deepblue_query_cache, deepblue_query_experiment_type,
depblue_score_matrix, deepblue_select_annotations, deepblue_select_experiments, deepblue_select_regions,
depblue_tiling_regions
**Examples**

```r
data_id = deepblue_select_experiments(
   experiment_name="E002-H3K9ac.narrowPeak.bed"
) deepblue_get_regions(query_id=data_id,
   output_format = "CHROMOSOME,START,END")
```

**Description**

Download the requested data. The output can be (i) a string (get_regions, score_matrix, and count_regions), or (ii) a list of ID and names (get_experiments_by_query), or (iii) a struct (coverage).

**Usage**

```r
deepblue_get_request_data(request_id = NULL, user_key = deepblue_USER_KEY)
```

**Arguments**

- `request_id` - A string (ID of the request)
- `user_key` - A string (users token key)

**Value**

- `data` - A string or a vector of string (the request data)

**See Also**

Other Requests status information and results: `deepblue_list_requests`
**deepblue_info**

**Description**

Information about a DeepBlue data identifier (ID). Any DeepBlue data ID can be queried with this command. For example, it is possible to obtain all available information about an Experiment using its ID, to obtain the actual Request processing status or the information about a Sample. A user can obtain information about him- or herself using the value ‘me’ in the parameter ‘id’. Multiple IDs can be queried in the same operation.

**Usage**

```
depthblue_info(id = NULL, user_key = deepblue_USER_KEY)
```

**Arguments**

- **id** - A string or a vector of string (ID or an array of IDs)
- **user_key** - A string (users token key)

**Value**

information - A array or a vector of array (List of Maps, where each map contains the info of an object.)

**See Also**

Other Commands for all types of data: [deepblue_cancel_request](#), [deepblue_is_biosource](#), [deepblue_list_in_use](#), [deepblue_name_to_id](#), [deepblue_search](#)

**Examples**

```
depthblue_info(id = "e30035")
```

---

**deepblue_input_regions**

**Description**

Upload a set of genomic regions that can be accessed through a query ID. An interesting use case for this command is to upload a set of custom regions for intersecting with genomic regions in DeepBlue to specifically select regions of interest.

**Usage**

```
depthblue_input_regions(genome = NULL, region_set = NULL, user_key = deepblue_USER_KEY)
```

---
**Arguments**

**genome** - A string (the target genome)

**region_set** - A string (Regions in CHROMOSOME START END format)

**user_key** - A string (users token key)

**Value**

**id** - A string (query id)

**See Also**

Other Operating on the data regions: deepblue_aggregate, deepblue_count_regions, deepblue_coverage, deepblue_extend, deepblue_filter_regions, deepblue_flank, deepblue_get_experiments_by_query, deepblue_get_regions, deepblue_intersection, deepblue_merge_queries, deepblue_query_cache, deepblue_query_experiment_type, deepblue_score_matrix, deepblue_select_annotations, deepblue_select_experiments, deepblue_select_regions, deepblue_tiling_regions

**Examples**

```r
regions_set = "chr1 28735 29810
chr1 135124 135563
chr1 327790 328229
chr1 437151 438164
chr1 449273 450544
chr1 533219 534114
chr1 544738 546649
chr1 713984 714547
chr1 762416 763445
chr1 788863 789211"

deepblue_input_regions(genome = "hg19",
region_set = regions_set)
```

**Description**

Select genomic regions that intersect with at least one region of the second query.

**Usage**

```r
deepblue_intersection(query_data_id = NULL, query_filter_id = NULL,
user_key = deepblue_USER_KEY)
```

**Arguments**

**query_data_id** - A string (query data that will be filtered.)

**query_filter_id** - A string (query containing the regions that the regions of the query_data_id must overlap.)

**user_key** - A string (users token key)
Value

id - A string (id of the new query)

See Also

Other Operating on the data regions: deepblue_aggregate, deepblue_count_regions, deepblue_coverage, deepblue_extend, deepblue_filter_regions, deepblue_flank, deepblue_get_experiments_by_query, deepblue_get_regions, deepblue_input_regions, deepblue_merge_queries, deepblue_query_cache, deepblue_query_experiment_type, deepblue_score_matrix, deepblue_select_annotations, deepblue_select_experiments, deepblue_select_regions, deepblue_tiling_regions

Examples

annotation_id = deepblue_select_annotations(
    annotation_name="CpG Islands",
    genome="hg19", chromosome="chr1")
data_id = deepblue_select_experiments(
    experiment_name="E002-H3K9ac.narrowPeak.bed")
deepblue_intersection(query_data_id = annotation_id, query_filter_id = data_id)
**deepblue_list_annotations**

*Description*

List all annotations of genomic regions currently available in DeepBlue.

*Usage*

```
deepblue_list_annotations(genome = NULL, user_key = deepblue_USER_KEY)
```

*Arguments*

- genome - A string or a vector of string (the target genome)
- user_key - A string (users token key)

*Value*

annotations - A array (annotations names and IDs)

*Examples*

```
deepblue_list_annotations(genome = "hg19")
```

---

**deepblue_list_biosources**

*Description*

List BioSources included in DeepBlue. A BioSource refers to a term describing the origin of a given sample, such as a tissue or cell line. It is possible to filter the BioSources by their extra_metadata fields content. These fields vary depending on the original data source.

*Usage*

```
deepblue_list_biosources(extra_metadata = NULL, user_key = deepblue_USER_KEY)
```

*Arguments*

- extra_metadata - A struct (Metadata that must be matched)
- user_key - A string (users token key)

*Value*

biosources - A array (biosources names and IDs)
See Also
Other Inserting and listing biosources: deepblue_list_similar_biosources

Examples

```r
deepblue_list_biosources(extra_metadata = list(ontology_id = "UBERON:0002485"))
```

---

### deepblue_list_column_types

**list_column_types**

**Description**

Lists the ColumnTypes included in DeepBlue.

**Usage**

```r
deepblue_list_column_types(user_key = deepblue_USER_KEY)
```

**Arguments**

- **user_key** - A string (users token key)

**Value**

- **column_types** - A array (column types names and IDS)

**Examples**

```r
deepblue_list_column_types()
```

---

### deepblue_list_epigenetic_marks

**list_epigenetic_marks**

**Description**

List Epigenetic Marks included in DeepBlue. This includes histone marks, DNA methylation, DNA sensitivity, etc. It is possible to filter the Epigenetic Marks by their extra_metadata field content.

**Usage**

```r
deepblue_list_epigenetic_marks(extra_metadata = NULL, user_key = deepblue_USER_KEY)
```

**Arguments**

- **extra_metadata** - A struct (Metadata that must be matched)
- **user_key** - A string (users token key)
Value

epigenetic_marks - A array (epigenetic mark names and IDS)

See Also

Other Inserting and listing epigenetic marks: deepblue_list_similar_epigenetic_marks

Examples

deepblue_list_epigenetic_marks()

Description

List the DeepBlue Experiments that matches the search criteria defined by this command parameters.

Usage

deepblue_list_experiments(genome = NULL, type = NULL, epigenetic_mark = NULL, biosource = NULL, sample = NULL, technique = NULL, project = NULL, user_key = deepblue_USER_KEY)

Arguments

genome - A string or a vector of string (the target genome)
type - A string or a vector of string (type of the experiment: peaks or signal)
epigenetic_mark - A string or a vector of string (name(s) of selected epigenetic mark(s))
biosource - A string or a vector of string (name(s) of selected biosource(s))
sample - A string or a vector of string (id(s) of selected sample(s))
technique - A string or a vector of string (name(s) of selected technique(s))
project - A string or a vector of string (name(s) of selected projects)
user_key - A string (users token key)

Value

experiments - A array (experiment names and IDS)

See Also

Other Inserting and listing experiments: deepblue_collection_experiments_count, deepblue_faceting_experiments, deepblue_list_recent_experiments, deepblue_list_similar_experiments, deepblue_preview_experiment
**Examples**

```r
deepblue_list_experiments(genome = "hg19", type = "peaks",
epigenetic_mark = "H3K27ac", biosource = "blood")
```

---

**Description**

List the Expression currently available in DeepBlue. A expression is a set of data with an identifier and an expression value.

**Usage**

```r
deepblue_list_expressions(expression_type = NULL, sample_id = NULL,
replica = NULL, project = NULL, user_key = deepblue_USER_KEY)
```

**Arguments**

- `expression_type` - A string (expression type (supported: 'gene'))
- `sample_id` - A string or a vector of string (sample ID(s))
- `replica` - A int or a vector of int (replica(s))
- `project` - A string or a vector of string (project(s) name)
- `user_key` - A string (users token key)

**Value**

expressions - A array (expressions names and IDS)

**See Also**

Other Expression data: `deepblue_select_expressions`

**Examples**

```r
deepblue_list_expressions(expression_type='gene')
```
**deepblue_list_genes**  

---

**Description**

List the Genes currently available in DeepBlue.

**Usage**

```r
depthblue_list_genes(genes = NULL, chromosome = NULL, start = NULL,  
end = NULL, gene_models = NULL, user_key = deepblue_USER_KEY)
```

**Arguments**

- **genes** - A string or a vector of string (Name(s) or ENSEMBL ID (ENSGXXXXXXXXXXX.X) of the gene(s)).
- **chromosome** - A string or a vector of string (chromosome name(s))
- **start** - A int (minimum start region)
- **end** - A int (maximum end region)
- **gene_models** - A string or a vector of string (the gene model)
- **user_key** - A string (users token key)

**Value**

- **genes** - A array (genes names and its content)

**See Also**

Other Gene models and genes identifiers: deepblue_list_gene_models, deepblue_select_genes

**Examples**

```r
depthblue_list_genes(  
  chromosome="chr20",  
  start=10000000,  
  end=21696620,  
  gene_models='Genode v22')
```
**deepblue_list_gene_models**

**list_gene_models**

**Description**

List all the Gene Models currently available in DeepBlue. A gene model is a set of genes usually imported from GENCODE. For example Gencode v22.

**Usage**

```python
deepblue_list_gene_models(user_key = deepblue_USER_KEY)
```

**Arguments**

- **user_key**: A string (users token key)

**Value**

- **gene_models**: A array (gene models names and IDs)

**See Also**

Other Gene models and genes identifiers: deepblue_list_genes, deepblue_select_genes

**Examples**

```python
deepblue_list_gene_models()
```

---

**deepblue_list_genomes**

**list_genomes**

**Description**

List Genomes assemblies that are registered in DeepBlue.

**Usage**

```python
deepblue_list_genomes(user_key = deepblue_USER_KEY)
```

**Arguments**

- **user_key**: A string (users token key)

**Value**

- **genomes**: A array (genome names)
**deepblue_list_in_use**

**See Also**

Other Inserting and listing genomes:  
deepblue_chromosomes, deepblue_list_similar_genomes

**Examples**

```
depthblue_list_genomes()
```

---

**Description**

List all terms used by the Experiments mandatory metadata that have at least one Experiment or Annotation using them.

**Usage**

```
depthblue_list_in_use(controlled_vocabulary = NULL,
user_key = depthblue_USER_KEY)
```

**Arguments**

- **controlled_vocabulary**
  - A string (controlled vocabulary name)
- **user_key**
  - A string (users token key)

**Value**

- **terms** - A array (controlled_vocabulary terms with count)

**See Also**

Other Commands for all types of data:  
deepblue_cancel_request, deepblue_info, deepblue_is_biosource,  
depthblue_name_to_id, deepblue_search

**Examples**

```
depthblue_list_in_use(controlled_vocabulary = "biosources")
```
**deepblue_list_projects**

*Description*
List Projects included in DeepBlue.

*Usage*
deepblue_list_projects(user_key = deepblue_USER_KEY)

*Arguments*
- user_key - A string (users token key)

*Value*
- projects - A array (project names)

*See Also*
Other Inserting and listing projects: deepblue_list_similar_projects

*Examples*
deepblue_list_projects()

---

**deepblue_list_recent_experiments**

*Description*
List the latest Experiments included in DeepBlue that match criteria defined in the parameters. The returned experiments are sorted by insertion date.

*Usage*
deepblue_list_recent_experiments(days = NULL, genome = NULL, epigenetic_mark = NULL, sample = NULL, technique = NULL, project = NULL, user_key = deepblue_USER_KEY)
**deepblue_list_requests**

**Arguments**
- **days** - A double (maximum days ago the experiments were added)
- **genome** - A string or a vector of string (the target genome)
- **epigenetic_mark** - A string or a vector of string (name(s) of selected epigenetic mark(s))
- **sample** - A string or a vector of string (id(s) of selected sample(s))
- **technique** - A string or a vector of string (name(s) of selected technique(es))
- **project** - A string or a vector of string (name(s) of selected projects)
- **user_key** - A string (users token key)

**Value**
- **experiments** - A array (names of recent experiments)

**See Also**
- Other Inserting and listing experiments: `deepblue_collection_experiments_count`, `deepblue_faceting_experiments`, `deepblue_list_experiments`, `deepblue_list_similar_experiments`, `deepblue_preview_experiment`

**Examples**
```r
deepblue_list_recent_experiments(days = 2, genome = "hg19")
```

---

**deepblue_list_requests**

```r
list_requests
```

**Description**
List the Requests made by the user. It is possible to obtain only the requests of a given state.

**Usage**
```r
deepblue_list_requests(request_state = NULL, user_key = deepblue_USER_KEY)
```

**Arguments**
- **request_state** - A string (Name of the state to get requests for. The valid states are: new, running, done, and failed.)
- **user_key** - A string (users token key)

**Value**
- **data_state** - A array (Request-IDs and their state)

**See Also**
- Other Requests status information and results: `deepblue_get_request_data`
Examples

```
 deepenblue_list_requests(request_state = 'running')
```

**deepblue_list_samples**  
list_samples

**Description**

List Samples included in DeepBlue. It is possible to filter by the BioSource and by extra_metadata fields content.

**Usage**

```
 deepenblue_list_samples(biosource = NULL, extra_metadata = NULL,
 user_key = deepblue_USER_KEY)
```

**Arguments**

- **biosource** - A string or a vector of string (name(s) of selected biosource(s))
- **extra_metadata** - A struct (Metadata that must be matched)
- **user_key** - A string (users token key)

**Value**

samples - A array (samples id with their content)

**Examples**

```
 deepenblue_list_samples(biosource = "Blood")
```

**deepblue_list_similar_biosources**  
list_similar_biosources

**Description**

List all BioSources that have a similar name compared to the provided name. A BioSource refers to a term describing the origin of a given sample, such as a tissue or cell line. The similarity is calculated using the Levenshtein method.

**Usage**

```
 deepenblue_list_similar_biosources(name = NULL, user_key = deepblue_USER_KEY)
```

**Arguments**

- **name** - A string (biosource name)
- **user_key** - A string (users token key)
**deepblue_list_similar_epigenetic_marks**

**Value**
biosource - A string (biosource name)

**See Also**
Other Inserting and listing biosources: deepblue_list_biosources

**Examples**

```r
depth_list_similar_biosources(name = "blood")
```

---

**Description**
List all Epigenetic Marks that have a similar name compared to the provided name. The similarity is calculated using the Levenshtein method.

**Usage**

```r
depth_list_similar_epigenetic_marks(name = NULL, user_key = depth_USER_KEY)
```

**Arguments**

- **name** - A string (epigenetic mark name)
- **user_key** - A string (users token key)

**Value**
epigenetic_marks - A array (similar epigenetic mark names)

**See Also**
Other Inserting and listing epigenetic marks: deepblue_list_epigenetic_marks

**Examples**

```r
depth_list_similar_epigenetic_marks(name = "H3k27ac")
```
**deepblue_list_similar_experiments**

*list_similar_experiments*

**Description**

List all Experiments that have a similar name compared to the provided name. The similarity is calculated using the Levenshtein method.

**Usage**

```r
deepblue_list_similar_experiments(name = NULL, genome = NULL, user_key = deepblue_USER_KEY)
```

**Arguments**

- `name` - A string (experiment name)
- `genome` - A string or a vector of string (the target genome)
- `user_key` - A string (users token key)

**Value**

- `experiments` - A array (similar experiment names)

**See Also**

Other Inserting and listing experiments: `deepblue_collection_experiments_count`, `deepblue_faceting_experiments`, `deepblue_list_experiments`, `deepblue_list_recent_experiments`, `deepblue_preview_experiment`

**Examples**

```r
deepblue_list_similar_experiments(name = "blood", genome = "hg19")
```

---

**deepblue_list_similar_genomes**

*list_similar_genomes*

**Description**

Lists all Genomes that have a similar name compared to the provided name. The similarity is calculated using the Levenshtein method.

**Usage**

```r
deepblue_list_similar_genomes(name = NULL, user_key = deepblue_USER_KEY)
```

**Examples**

```r
deepblue_list_similar_genomes(name = "blood", genome = "hg19")
```
Arguments

name - A string (genome name)
user_key - A string (users token key)

Value

genomes - A array (similar genome names)

See Also

Other Inserting and listing genomes: deepblue_chromosomes, deepblue_list_genomes

Examples

depblue_list_similar_genomes(name = "grc")

Description

List Projects that have a similar name compared to the provided name. The similarity is calculated using the Levenshtein method.

Usage

depblue_list_similar_projects(name = NULL, user_key = deepblue_USER_KEY)

Arguments

name - A string (project name)
user_key - A string (users token key)

Value

projects - A array (similar project names)

See Also

Other Inserting and listing projects: deepblue_list_projects

Examples

depblue_list_similar_projects(name = "BLUEPRINT")
deepblue_list_similar_techniques

Description
List Techniques that have a similar name compared to the provided name. The similarity is calculated using the Levenshtein method.

Usage
deepblue_list_similar_techniques(name = NULL, user_key = deepblue_USER_KEY)

Arguments
name - A string (technique name)
user_key - A string (users token key)

Value
techniques - A array (similar techniques)

See Also
Other Inserting and listing techniques: deepblue_list_techniques

Examples
deepblue_list_similar_techniques(name = "chip seq")

deepblue_list_techniques

Description
List the Techniques included in DeepBlue.

Usage
deepblue_list_techniques(user_key = deepblue_USER_KEY)

Arguments
user_key - A string (users token key)

Value
techniques - A array (techniques)
**deepblue_merge_queries**

See Also

Other Inserting and listing techniques: deepblue_list_similar_techniques

Examples

deeblue_list_techniques()

---

deepblue_merge_queries

**merge_queries**

Description

Merge regions from two queries in a new query.

Usage

deeblue_merge_queries(query_a_id = NULL, query_b_id = NULL, user_key = deepblue_USER_KEY)

Arguments

query_a_id - A string (id of the first query)
query_b_id - A string (id of the second query)
user_key - A string (users token key)

Value

id - A string (new query id)

See Also

Other Operating on the data regions: deepblue_aggregate, deepblue_count_regions, deepblue_coverage, deepblue_extend, deepblue_filter_regions, deepblue_flank, deepblue_get_experiments_by_query, deepblue_get_regions, deepblue_input_regions, deepblue_intersection, deepblue_query_cache, deepblue_query_experiment_type, deepblue_score_matrix, deepblue_select_annotations, deepblue_select_experiments, deepblue_select_regions, deepblue_tiling_regions

Examples

annotation_id = deepblue_select_annotations(
    annotation_name="CpG Islands",
    genome="hg19", chromosome="chr1")
data_id = deepblue_select_experiments(
    experiment_name="E002-H3K9ac.narrowPeak.bed")
deeblue_merge_queries(
    query_a_id = annotation_id,
    query_b_id = data_id)
**deepblue_name_to_id**  

**name_to_id**

**Description**

Obtain the data ID(s) from the informed data name(s).

**Usage**

```r
deepblue_name_to_id(name = NULL, collection = NULL,  
user_key = deepblue_USER_KEY)
```

**Arguments**

- `name` - A string or a vector of string (ID or an array of IDs)
- `collection` - A string (Collection where the data name is in)
- `user_key` - A string (users token key)

**Value**

- A array or a vector of array (List of IDs.)

**See Also**

Other Commands for all types of data: deepblue_cancel_request, deepblue_info, deepblue_is_biosource, deepblue_list_in_use, deepblue_search

**Examples**

```r
deepblue_name_to_id("E002-H3K9ac.narrowPeak.bed", "experiments")
deepblue_name_to_id("prostate duct", "biosources")
deepblue_name_to_id("DNA Methylation", "Epigenetic_marks")
```

---

**deepblue_preview_experiment**  

**preview_experiment**

**Description**

List the DeepBlue Experiments that matches the search criteria defined by this command parameters.

**Usage**

```r
deepblue_preview_experiment(experiment_name = NULL,  
user_key = deepblue_USER_KEY)
```

**Description**

List the DeepBlue Experiments that matches the search criteria defined by this command parameters.
**deepblue_query_cache**

**Arguments**

- experiment_name
  - A string (name(s) of selected experiment(s))
- user_key
  - A string (users token key)

**Value**

- experiment - A string (experiment’s regions)

**See Also**

Other Inserting and listing experiments: deepblue_collection_experiments_count, deepblue_faceting_experiments, deepblue_list_experiments, deepblue_list_recent_experiments, deepblue_list_similar_experiments

**Examples**

depthblue_preview_experiment(’S00JRH1.ERX683143.H3K4me3.bwa.GRCh38.20150527.bed’)
**Examples**

```r
annotation_id = deepblue_select_annotations(
    annotation_name="CpG Islands",
    genome="hg19", chromosome="chr1")
data_id = deepblue_select_experiments(
    experiment_name="E002-H3K9ac.narrowPeak.bed")
merged_regions = deepblue_merge_queries(
    query_a_id = annotation_id,
    query_b_id = data_id)
deepblue_query_cache(
    query_id = merged_regions, cache = TRUE)
```

---

**deepblue_query_experiment_type**

**query_experiment_type**

---

**Description**

Filter the query ID for regions associated with experiments of a given type. For example, it is possible to select only peaks using this command with the ‘peaks’ parameter.

**Usage**

```r
deepblue_query_experiment_type(query_id = NULL, type = NULL,
    user_key = deepblue_USER_KEY)
```

**Arguments**

- `query_id` - A string (query ID)
- `type` - A string (experiment type (peaks or signal))
- `user_key` - A string (users token key)

**Value**

- `information` - A string (New query ID.)

**See Also**

Other Operating on the data regions: `deepblue_aggregate`, `deepblue_count_regions`, `deepblue_coverage`, `deepblue_extend`, `deepblue_filter_regions`, `deepblue_flank`, `deepblue_get_experiments_by_query`, `deepblue_get_regions`, `deepblue_input_regions`, `deepblue_intersection`, `deepblue_merge_queries`, `deepblue_query_cache`, `deepblue_score_matrix`, `deepblue_select_annotations`, `deepblue_select_experiments`, `deepblue_select_regions`, `deepblue_tiling_regions`
Examples

```r
h3k27ac_regions = deepblue_select_regions(
    genome = 'GRCh38',
    epigenetic_mark = 'H3k27ac',
    project = 'BLUEPRINT Epigenome',
    chromosome = 'chr1')
deeblue_query_experiment_type(
    query_id = h3k27ac_regions,
    type = "peaks")
```

deeblue_score_matrix  score_matrix

Description

Build a matrix containing the aggregation result of the experiments data by the aggregation boundaries.

Usage

```r
deepblue_score_matrix(experiments_columns = NULL,
                        aggregation_function = NULL, aggregation_regions_id = NULL,
                        user_key = deepblue_USER_KEY)
```

Arguments

- **experiments_columns**
  - A struct (map with experiments names and columns to be processed. Example: `wgEncodeBroadHistoneDnd41H3k27acSig.wig': 'VALUE', 'wgEncodeBroadHistoneCd20w001794H3k27acSig.wig': 'VALUE')

- **aggregation_function**
  - A string (aggregation function name: min, max, sum, mean, var, sd, median, count, boolean)

- **aggregation_regions_id**
  - A string (query ID of the regions that will be used as the aggregation boundaries)

- **user_key**
  - A string (users token key)

Value

- **score_matrix** - A string (the score matrix containing the summarized data)

See Also

- Other Operating on the data regions: `deepblue_aggregate`, `deepblue_count_regions`, `deepblue_coverage`, `deepblue_extend`, `deepblue_filter_regions`, `deepblue_flank`, `deepblue_get_experiments_by_query`, `deepblue_get_regions`, `deepblue_input_regions`, `deepblue_intersection`, `deepblue_merge_queries`, `deepblue_query_cache`, `deepblue_query_experiment_type`, `deepblue_select_annotations`, `deepblue_select_experiments`, `deepblue_select_regions`, `deepblue_tiling_regions`
Examples

tiling_regions = deepblue_tiling_regions(
    size=100000, genome="hg19", chromosome="chr1")
deeplue_score_matrix(
    experiments_columns =
        list(ENCFF721EKA="VALUE", ENCFF781VWH="VALUE"),
    aggregation_function = "mean",
    aggregation_regions_id = tiling_regions)

Description

Search all data of all types for the given keyword. A minus (-) character in front of a keyword searches for data without the given keyword. The search can be restricted to the following data types are: Annotations, Biosources, Column_types, Epigenetic_marks, Experiments, Genomes, Gene_models, Gene_expressions, Genes, Projects, Samples, Techniques, Tilings.

Usage

deepblue_search(keyword = NULL, type = NULL, user_key = deepblue_USER_KEY)

Arguments

keyword - A string (keyword to search by)
type - A string or a vector of string (type of data to search for - Annotations, Biosources, Column_types, Epigenetic_marks, Experiments, Genomes, Gene_models, Gene_expressions, Genes, Projects, Samples, Techniques, Tilings)
user_key - A string (users token key)

Value

results - A array (search results as [id, name, type])

See Also

Other Commands for all types of data: deepblue_cancel_request, deepblue_info, deepblue_is_biosource, deepblue_list_in_use, deepblue_name_to_id

Examples

deepblue_search(keyword = "DNA Methylation BLUEPRINT",
    type = "experiments")
**deepblue_select_annotations**

**select_annotations**

---

**Description**

Select regions from the Annotations that match the selection criteria.

**Usage**

```r
deepblue_select_annotations(annotation_name = NULL, genome = NULL,
                             chromosome = NULL, start = NULL, end = NULL,
                             user_key = deepblue_USER_KEY)
```

**Arguments**

- `annotation_name`: A string or a vector of string (name(s) of selected annotation(s))
- `genome`: A string or a vector of string (the target genome)
- `chromosome`: A string or a vector of string (chromosome name(s))
- `start`: A int (minimum start region)
- `end`: A int (maximum end region)
- `user_key`: A string (users token key)

**Value**

- `id`: A string (query id)

**See Also**

Other Operating on the data regions: `deepblue_aggregate`, `deepblue_count_regions`, `deepblue_coverage`, `deepblue_extend`, `deepblue_filter_regions`, `deepblue_flank`, `deepblue_get_experiments_by_query`, `deepblue_get_regions`, `deepblue_input_regions`, `deepblue_merge_queries`, `deepblue_query_cache`, `deepblue_query_experiment_type`, `deepblue_score_matrix`, `deepblue_select_experiments`, `deepblue_select_regions`, `deepblue_tiling_regions`

**Examples**

```r
deepblue_select_annotations(
    annotation_name = "Cpg Islands",
    genome = "hg19",
    chromosome = "chr1",
    start = 0,
    end = 2000000)
```
deepblue_select_column

Description

A utility command that creates a list of experiments in which a specific column is selected. Such a list is needed as input for deepblue_score_matrix.

Usage

deepblue_select_column(experiments, column, user_key = deepblue_USER_KEY)

Arguments

- experiments: A data frame with experiments obtained from deepblue_list_experiments
- column: The name of the column that is extracted from each experiment file
- user_key: A string (user's token key)

Value

A list of experiments with the selected column

See Also

depthblue_score_matrix
depthblue_list_experiments

Other Utilities for information processing: depthblue_diff

Examples

blueprint_DNA_meth <- deepblue_list_experiments(
  genome = "GRCh38",
  epigenetic_mark = "DNA Methylation",
  technique = "Bisulfite-Seq",
  project = "BLUEPRINT EPIGENOME")

blueprint_DNA_meth <- blueprint_DNA_meth[grep("bs_call",
    deepblue_extract_names(blueprint_DNA_meth)),]

exp_columns <- deepblue_select_column(blueprint_DNA_meth, "VALUE")
**Description**

Selects regions from Experiments by the experiments names.

**Usage**

```r
depthblue_select_experiments(experiment_name = NULL, chromosome = NULL, start = NULL, end = NULL, user_key = depthblue_USER_KEY)
```

**Arguments**

- `experiment_name` - A string or a vector of string (name(s) of selected experiment(s))
- `chromosome` - A string or a vector of string (chromosome name(s))
- `start` - A int (minimum start region)
- `end` - A int (maximum end region)
- `user_key` - A string (users token key)

**Value**

- `id` - A string (query id)

**See Also**

Other Operating on the data regions: `depthblue_aggregate`, `depthblue_count_regions`, `depthblue_coverage`, `depthblue_extend`, `depthblue_filter_regions`, `depthblue_flank`, `depthblue_get_experiments_by_query`, `depthblue_get_regions`, `depthblue_input_regions`, `depthblue_intersection`, `depthblue_merge_queries`, `depthblue_query_cache`, `depthblue_query_experiment_type`, `depthblue_score_matrix`, `depthblue_select_annotations`, `depthblue_select_regions`, `depthblue_tiling_regions`

**Examples**

```r
depthblue_select_experiments(
  experiment_name = c("E002-H3K9ac.narrowPeak.bed", "E001-H3K4me3.gappedPeak.bed")
)
```
Description

Select expressions (by their name or ID) as genomic regions from the specified model.

Usage

depthblue_select_expressions(expression_type = NULL, sample_ids = NULL, 
replicas = NULL, identifiers = NULL, projects = NULL, 
gene_model = NULL, user_key = deepblue_USER_KEY)

Arguments

expression_type - A string (expression type (supported: 'gene'))
sample_ids - A string or a vector of string (id(s) of selected sample(s))
replicas - A int or a vector of int (replica(s))
identifiers - A string or a vector of string (identifier(s) (for genes: ensembl ID or ENSB name).)
projects - A string or a vector of string (projects(s))
gene_model - A string (gene model name)
user_key - A string (users token key)

Value

id - A string (query id)

See Also

Other Expression data: deepblue_list_expressions

Examples

genes_names = c("CCR1", "CD164", "CD10", "CD2", "CD34", "CD3G", "CD44")
depthblue_select_expressions(
  expression_type="gene",
  sample_ids="s10205",
  identifiers = genes_names,
  gene_model = "gencode v23")
deepblue_select_genes

Description

Select genes (by their name or ID) as genomic regions from the specified gene model.

Usage

deepblue_select_genes(genes = NULL, gene_model = NULL, chromosome = NULL, start = NULL, end = NULL, user_key = deepblue_USER_KEY)

Arguments

genes - A string or a vector of string (Name(s) or ENSEMBL ID (ENSGXXXXXXXXXXX.X) of the gene(s).)
gene_model - A string (gene model name)
chromosome - A string or a vector of string (chromosome name(s))
start - A int (minimum start region)
end - A int (maximum end region)
user_key - A string (users token key)

Value

id - A string (query id)

See Also

Other Gene models and genes identifiers: deepblue_list_gene_models, deepblue_list_genes

Examples

genes_names =
c('CCR1', 'CD164', 'CD1D', 'CD2', 'CD34', 'CD3G', 'CD44')
deepluee_select_genes(
    genes = genes_names,
    gene_model = "gencode v23")
deepblue_select_regions

**select_regions**

**Description**

Selects Experiment regions that matches the criteria informed by the operation parameters.

**Usage**

```r
deepblue_select_regions(experiment_name = NULL, genome = NULL,
epigenetic_mark = NULL, sample_id = NULL, technique = NULL,
project = NULL, chromosomes = NULL, start = NULL, end = NULL,
user_key = deepblue_USER_KEY)
```

**Arguments**

- `experiment_name` - A string or a vector of string (name(s) of selected experiment(s))
- `genome` - A string or a vector of string (the target genome)
- `epigenetic_mark` - A string or a vector of string (name(s) of selected epigenetic mark(s))
- `sample_id` - A string or a vector of string (id(s) of selected sample(s))
- `technique` - A string or a vector of string (name(s) of selected technique(es))
- `project` - A string or a vector of string (name(s) of selected projects)
- `chromosomes` - A string or a vector of string (chromosome name(s))
- `start` - A int (minimum start region)
- `end` - A int (maximum end region)
- `user_key` - A string (users token key)

**Value**

- `id` - A string (query id)

**See Also**

Other Operating on the data regions: `deepblue_aggregate`, `deepblue_count_regions`, `deepblue_coverage`, `deepblue_extend`, `deepblue_filter_regions`, `deepblue_flank`, `deepblue_get_experiments_by_query`, `deepblue_get_regions`, `deepblue_input_regions`, `deepblue_intersection`, `deepblue_merge_queries`, `deepblue_query_cache`, `deepblue_query_experiment_type`, `deepblue_score_matrix`, `deepblue_select_annotations`, `deepblue_select_experiments`, `deepblue_tiling_regions`

**Examples**

```r
deepblue_select_regions(
  genome="hg19",
  epigenetic_mark = "H3K27ac",
  project = " BLUEPRINT Epigenome")
```
deepblue_tiling_regions

Description

Generate tiling regions across the genome chromosomes. The idea is to "bin" genomic regions systematically in order to obtain discrete regions over which one can aggregate. Using the 'score_matrix' command, these bins (tiles) can be compared directly across experiments.

Usage

depthblue_tiling_regions(size = NULL, genome = NULL, chromosome = NULL, user_key = depthblue_USER_KEY)

Arguments

size - A int (tiling size)
genome - A string (the target genome)
chromosome - A string or a vector of string (chromosome name(s))
user_key - A string (users token key)

Value

id - A string (query id)

See Also

Other Operating on the data regions: deepblue_aggregate, deepblue_count_regions, deepblue_coverage, deepblue_extend, deepblue_filter_regions, deepblue_flank, deepblue_get_experiments_by_query, deepblue_get_regions, deepblue_input_regions, deepblue_intersection, deepblue_merge_queries, deepblue_query_cache, deepblue_query_experiment_type, deepblue_score_matrix, deepblue_select_annotations, deepblue_select_experiments, deepblue_select_regions

Examples

depthblue_tiling_regions(
    size = 10000,
    genome = "hg19",
    chromosome = "chr1")
**deepblue_wait_request**

---

**Description**

Process the user request. Takes in three parameters: requested regions, sleep time, and user key.

**Usage**

```
deepblue_wait_request(request_id, sleep_time = 1,
                      user_key = deepblue_USER_KEY)
```

**Arguments**

- `request_id`  A string with the request_id
- `sleep_time`  An integer with default value 1s
- `user_key`    A string

**Value**

`request_id info`
Index

deepluare_aggregate, 3, 8, 9, 12, 15, 16, 20, 24, 39, 41–43, 45, 47, 50, 51
deepluare_batch_export_results, 4
deepluare_cancel_request, 5, 22, 24, 31, 40, 44
deepluare_chromosomes, 6, 31, 37
deepluare_collection_experiments_count, 6, 14, 27, 33, 36, 41
deepluare_commands, 7, 11
deepluare_count_regions, 4, 8, 9, 12, 15, 16, 20, 23, 24, 39, 41–43, 45, 47, 50, 51
deepluare_coverage, 4, 8, 9, 12, 15, 16, 20, 23, 24, 39, 41–43, 45, 47, 50, 51
deepluare_diff, 9, 46
deepluare_download_request_data, 10
deepluare_download_request_data, DeepluareCommands
11
deepluare_echo, 7, 11
deepluare_extend, 4, 8, 9, 12, 15, 16, 20, 23, 24, 39, 41–43, 45, 47, 50, 51
deepluare_extract_ids, 13, 14
deepluare_extract_names, 13, 13
deepluare_faceting_experiments, 7, 14, 27, 33, 36, 41
deepluare_filter_regions, 4, 8, 9, 12, 15, 16, 20, 23, 24, 39, 41–43, 45, 47, 50, 51
deepluare_flank, 4, 8, 9, 12, 15, 16, 20, 23, 24, 39, 41–43, 45, 47, 50, 51
deepluare_get_biosource_children, 17, 18, 19
deepluare_get_biosource_parents, 17, 17, 18, 19
deepluare_get_biosource_related, 17, 18, 18, 19
deepluare_get_biosource_synonyms, 17, 18, 19
deepluare_get_experiments_by_query, 4, 8, 9, 12, 15, 16, 19, 20, 23, 24, 39, 41–43, 45, 47, 50, 51
deepluare_get_regions, 4, 8, 9, 12, 15, 16, 20, 20, 23, 24, 39, 41–43, 45, 47, 50, 51
deepluare_get_request_data, 21, 33
deepluare_info, 5, 22, 24, 31, 40, 44
deepluare_input_regions, 4, 8, 9, 12, 15, 16, 20, 22, 24, 39, 41–43, 45, 47, 50, 51
deepluare_intersection, 4, 8, 9, 12, 15, 16, 20, 23, 24, 39, 41–43, 45, 47, 50, 51
deepluare_is_biosource, 5, 22, 24, 31, 40, 44
deepluare_list_annotations, 25
deepluare_list_biosources, 25, 35
deepluare_list_column_types, 26
deepluare_list_epigenetic_marks, 26, 35
deepluare_list_experiments, 7, 14, 27, 33, 36, 41, 46
deepluare_list_expressions, 28, 48
deepluare_list_gene_models, 29, 30, 49
deepluare_list_genes, 29, 30, 49
deepluare_list_genomes, 6, 30, 37
deepluare_list_in_use, 5, 22, 24, 31, 40, 44
deepluare_list_projects, 32, 37
deepluare_list_recent_experiments, 7, 14, 27, 32, 36, 41
deepluare_list_requests, 21, 33
deepluare_list_samples, 34
deepluare_list_similar_biosources, 26, 34
deepluare_list_similar_epigenetic_marks, 27, 35
deepluare_list_similar_experiments, 7, 14, 27, 33, 36, 41
deepluare_list_similar_genomes, 6, 31, 36
deepluare_list_similar_projects, 32, 37
deepluare_list_similar_techniques, 38, 39
deepluare_list_techniques, 38, 38
deepluare_merge_queries, 4, 8, 9, 12, 15, 16, 20, 23, 24, 39, 41–43, 45, 47, 50, 51
deepluare_name_to_id, 5, 22, 24, 31, 40, 44
deepluare_preview_experiment, 7, 14, 27, 33, 36, 40
deepluare_query_cache, 4, 8, 9, 12, 15, 16, 20, 23, 24, 39, 41, 42, 43, 45, 47, 50, 51
deepluare_query_experiment_type, 4, 8, 9,
deepblue_score_matrix, 4, 8, 9, 12, 15, 16, 20, 23, 24, 39, 41, 42, 43, 45, 47, 50, 51

deeblue_search, 5, 22, 24, 31, 40, 44

deeblue_select_annotations, 4, 8, 9, 12, 15, 16, 20, 23, 24, 39, 41–43, 45, 47, 50, 51

deeblue_select_column, 10, 46

deeblue_select_experiments, 4, 8, 9, 12, 15, 16, 20, 23, 24, 39, 41–43, 45, 47, 50, 51

deeblue_select_expression, 28, 48

deeblue_select_genes, 29, 30, 49

deeblue_select_regions, 4, 8, 9, 12, 15, 16, 20, 23, 24, 39, 41–43, 45, 47, 50, 51

deeblue_tiling_regions, 4, 8, 9, 12, 15, 16, 20, 23, 24, 39, 41–43, 45, 47, 50, 51

deeblue_wait_request, 52

DeepBlueCommand

(DeepBlueCommand-class), 3

DeepBlueCommand-class, 3