# Package ‘DeepBlueR’

March 28, 2017

**Title**  DeepBlueR  
**Type**  Package  
**Description**  Accessing the DeepBlue Epigenetics Data Server through R.  
**Version**  1.0.11  
**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, diffR, dplyr, methods, rjson, utils, R.utils, foreach, withr, rtracklayer, GenomeInfoDb, settings, filehash  
**Depends**  R (>= 3.3), XML, RCurl  
**BuildVignettes**  TRUE  
**RoxygenNote**  6.0.1  
**Suggests**  knitr, rmarkdown, LOLA, Gviz, gplots, ggplot2, tidyR, RColorBrewer, matrixStats  
**VignetteBuilder**  knitr  
**biocViews**  DataImport, DataRepresentation, ThirdPartyClient, GeneRegulation, GenomeAnnotation, CpGIsland, DNAMethylation, Epigenetics, Annotation, Preprocessing  
**NeedsCompilation**  no  

## R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepBlueCommand-class</td>
<td>3</td>
</tr>
<tr>
<td>deepblue_aggregate</td>
<td>4</td>
</tr>
<tr>
<td>deepblue_batch_export_results</td>
<td>5</td>
</tr>
<tr>
<td>deepblue_binning</td>
<td>5</td>
</tr>
<tr>
<td>deepblue_cache_status</td>
<td>6</td>
</tr>
<tr>
<td>deepblue_calculate_enrichment</td>
<td>7</td>
</tr>
<tr>
<td>deepblue_cancel_request</td>
<td>8</td>
</tr>
<tr>
<td>deepblue_chromosomes</td>
<td>8</td>
</tr>
<tr>
<td>R topic</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>deepblue_clear_cache</td>
<td>9</td>
</tr>
<tr>
<td>deepblue_collection_experiments_count</td>
<td>9</td>
</tr>
<tr>
<td>deepblue_commands</td>
<td>10</td>
</tr>
<tr>
<td>deepblue_count_gene_ontology_terms</td>
<td>11</td>
</tr>
<tr>
<td>deepblue_count_regions</td>
<td>12</td>
</tr>
<tr>
<td>deepblue_coverage</td>
<td>12</td>
</tr>
<tr>
<td>deepblue_delete_request_from_cache</td>
<td>13</td>
</tr>
<tr>
<td>deepblue_diff</td>
<td>14</td>
</tr>
<tr>
<td>deepblue_download_request_data</td>
<td>14</td>
</tr>
<tr>
<td>deepblue_download_request_data,DeepBlueCommand-method</td>
<td>15</td>
</tr>
<tr>
<td>deepblue_echo</td>
<td>16</td>
</tr>
<tr>
<td>deepblue_export_bed</td>
<td>16</td>
</tr>
<tr>
<td>deepblue_export_meta_data</td>
<td>17</td>
</tr>
<tr>
<td>deepblue_export_tab</td>
<td>18</td>
</tr>
<tr>
<td>deepblue_extend</td>
<td>18</td>
</tr>
<tr>
<td>deepblue_extract_ids</td>
<td>19</td>
</tr>
<tr>
<td>deepblue_extract_names</td>
<td>20</td>
</tr>
<tr>
<td>deepblue_faceting_experiments</td>
<td>21</td>
</tr>
<tr>
<td>deepblue_filter_regions</td>
<td>22</td>
</tr>
<tr>
<td>deepblue_flank</td>
<td>23</td>
</tr>
<tr>
<td>deepblue_format_object_size</td>
<td>24</td>
</tr>
<tr>
<td>deepblue_get_biosource_children</td>
<td>24</td>
</tr>
<tr>
<td>deepblue_get_biosource_parents</td>
<td>25</td>
</tr>
<tr>
<td>deepblue_get_biosource_related</td>
<td>26</td>
</tr>
<tr>
<td>deepblue_get_biosource_synonyms</td>
<td>26</td>
</tr>
<tr>
<td>deepblue_get_db</td>
<td>27</td>
</tr>
<tr>
<td>deepblue_get_experiments_by_query</td>
<td>27</td>
</tr>
<tr>
<td>deepblue_get_regions</td>
<td>28</td>
</tr>
<tr>
<td>deepblue_get_request_data</td>
<td>29</td>
</tr>
<tr>
<td>deepblue_info</td>
<td>30</td>
</tr>
<tr>
<td>deepblue_input_regions</td>
<td>30</td>
</tr>
<tr>
<td>deepblue_intersection</td>
<td>31</td>
</tr>
<tr>
<td>deepblue_is_biosource</td>
<td>32</td>
</tr>
<tr>
<td>deepblue_liftover</td>
<td>33</td>
</tr>
<tr>
<td>deepblue_list_annotations</td>
<td>34</td>
</tr>
<tr>
<td>deepblue_list_biosources</td>
<td>34</td>
</tr>
<tr>
<td>deepblue_list_cached_requests</td>
<td>35</td>
</tr>
<tr>
<td>deepblue_list_column_types</td>
<td>35</td>
</tr>
<tr>
<td>deepblue_list_epigenetic_marks</td>
<td>36</td>
</tr>
<tr>
<td>deepblue_list_experiments</td>
<td>36</td>
</tr>
<tr>
<td>deepblue_list_expressions</td>
<td>37</td>
</tr>
<tr>
<td>deepblue_list_genes</td>
<td>38</td>
</tr>
<tr>
<td>deepblue_list_gene_models</td>
<td>39</td>
</tr>
<tr>
<td>deepblue_list_genomes</td>
<td>39</td>
</tr>
<tr>
<td>deepblue_list_in_use</td>
<td>40</td>
</tr>
<tr>
<td>deepblue_list_projects</td>
<td>41</td>
</tr>
<tr>
<td>deepblue_list_recent_experiments</td>
<td>41</td>
</tr>
<tr>
<td>deepblue_list_requests</td>
<td>42</td>
</tr>
<tr>
<td>deepblue_list_samples</td>
<td>43</td>
</tr>
<tr>
<td>deepblue_list_similar_biosources</td>
<td>43</td>
</tr>
<tr>
<td>deepblue_list_similar_epigenetic_marks</td>
<td>44</td>
</tr>
<tr>
<td>deepblue_list_similar_experiments</td>
<td>45</td>
</tr>
</tbody>
</table>
**DeepBlueCommand-class**

Index

DeepBlueCommand-class  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

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

deepblue_aggregate(data_id = NULL, ranges_id = NULL, column = NULL, user_key = deepblue_options("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_binning, deepblue_calculate_enrichment, 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_overlap, 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")
deepblue_batch_export_results

**Description**
Write results from DeepBlue to disk as they become available

**Usage**
```r
deepblue_batch_export_results(requests, target.directory = NULL, suffix = "result", prefix = "DeepBlue", sleep.time = 1, bed.format = TRUE, user_key = deepblue_options("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
- **bed.format** whether to store the results as BED files or tab delimited.
- **user_key** A string used to authenticate the user

**Value**
A list containing the requests IDs data

**Examples**
```r
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))
```

deeblue_binning

**Description**
Bin results according to counts.

**Usage**
```r
deepblue_binning(query_data_id = NULL, column = NULL, bins = NULL, user_key = deepblue_options("user_key"))
```
deepblue_cache_status

Description
Report on the cache size and status

Usage
deepblue_cache_status()

Value
cache size in byte

Examples

depthblue_cache_status()
deepblue_calculate_enrichment

calculate_enrichment

Description

Enrich the regions based on Gene Ontology terms.

Usage

depthblue_calculate_enrichment(query_id = NULL, gene_model = NULL, user_key = depthblue_options("user_key"))

Arguments

query_id - A string (Query ID)
gene_model - A string (the gene model)
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_binning, deepblue_count_regions, deepblue_coverage, deepblue_extend, deepblue_filter_regions, deepblue_flank, deepblue_get_experiments, deepblue_get_regions, deepblue_input_regions, deepblue_intersection, deepblue_merge_queries, deepblue_overlap, 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")

filtered_id = deepblue_filter_regions(query_id = data_id, 
    field = "VALUE", 
    operation = ">",
    value = "100", 
    type = "number", 
    user_key = "anonymous_key")

depthblue_calculate_enrichment(query_id = filtered_id, 
    gene_model = "gencode v23")
```
### deepblue_cancel_request

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

**Usage**
```r
deepblue_cancel_request(id = NULL, user_key = deepblue_options("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**
```r
deepblue_cancel_request(id = "r12345")
```

---

### deepblue_chromosomes

**Description**
List the chromosomes of a given Genome.

**Usage**
```r
deepblue_chromosomes(genome = NULL, user_key = deepblue_options("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)
**deepblue_clear_cache**

**See Also**

Other Inserting and listing genomes: `deepblue_list_genomes, deepblue_list_similar_genomes`

**Examples**

```r
deepblue_chromosomes(genome = "g1")
```

**deepblue_clear_cache  Clear cache**

**Description**

Clear cache

**Usage**

```r
deepblue_clear_cache()
```

**Value**

TRUE if successful

**Examples**

```r
deepblue_clear_cache()
```

**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**

```r
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_options("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

deepblue_collection_experiments_count(
  controlled_vocabulary="epigenetic_marks",
  genome = "hg19", type = "peaks",
  biosource = "blood")

---

depthCommands commands

depthCommands

Description

List all available DeepBlue commands.

Usage

depthCommands()

Value

commands - A struct (command descriptions)

See Also

Other Checking DeepBlue status: depthEcho
Examples

deepblue_commands()

depthblue_count_gene_ontology_terms
count_gene_ontology_terms

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

depthblue_count_gene_ontology_terms(genes = NULL, go_terms = NULL,
chromosome = NULL, start = NULL, end = NULL, gene_model = NULL,
user_key = depthblue_options("user_key"))

Arguments

genes - A string or a vector of string (Name(s) or ENSEMBL ID (ENSGXXXXXXXXXXX.X ) of the gene(s).)
go_terms - A string or a vector of string (gene ontology terms - ID or label)
chromosome - A string or a vector of string (chromosome name(s))
start - A int (minimum start region)
end - A int (maximum end region)
gene_model - A string (the gene model)
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 Gene models and genes identifiers: depthblue_list_gene_models, depthblue_list_genes, depthblue_select_genes

Examples

gene_names = c("CCR1", "CD164", "CD1D", "CD2", "CD34", "CD3G", "CD44")
depthblue_count_gene_ontology_terms (genes = gene_names, gene_model = "gencode v23")
deepblue_count_regions

count_regions

Description
Return the number of genomic regions present in the query.

Usage
deepblue_count_regions(query_id = NULL, 
user_key = deepblue_options("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: deepblue_aggregate, deepblue_binning, deepblue_calculate_enrichment, 
deepblue_coverage, deepblue_extend, deepblue_filter_regions, deepblue_flank, deepblue_get_experiment, 
deepblue_get_regions, deepblue_input_regions, deepblue_intersection, deepblue_merge_queries, 
deepblue_overlap, deepblue_query_cache, deepblue_query_experiment_type, deepblue_score_matrix, 
deepblue_select_annotations, deepblue_select_experiments, deepblue_select_regions, 
deepblue_tiling_regions

Examples
data_id = deepblue_select_experiments(
    experiment_name="E002-H3K9ac.narrowPeak.bed")
deepblue_count_regions(query_id = data_id)

depblue_coverage

coverage

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

Usage
deepblue_coverage(query_id = NULL, genome = NULL, 
user_key = deepblue_options("user_key"))
deepblue_delete_request_from_cache

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_binning, deepblue_calculate_enrichment, deepblue_count_regions, deepblue_extend, deepblue_filter_regions, deepblue_flank, deepblue_get_experiments, deepblue_get_regions, deepblue_input_regions, deepblue_intersection, deepblue_merge_queries, deepblue_overlap, deepblue_query_cache, deepblue_query_experiment_type, deepblue_score_matrix, deepblue_select_annotations, deepblue_select_experiments, deepblue_select_regions, deepblue_tiling_regions

Examples

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

deepblue_delete_request_from_cache

Delete a specific request from the cache

Description

Delete a specific request from the cache

Usage

deepblue_delete_request_from_cache(request_id)

Arguments

request_id - the request to delete from the cache

Value

TRUE if the request was successfully deleted, FALSE otherwise

Examples

deepblue_delete_request_from_cache("non-existing-request-id")
# returns FALSE
deepblue_diff  

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

**Usage**  
```
deepblue_diff(id1, id2, user_key = deepblue_options("user_key"))
```

**Arguments**  
- **id1** - A DeepBlue id  
- **id2** - Another DeepBlue id  
- **user_key** - A string (users token key)

**Value**  
None

**See Also**  
Other Utilities for information processing: deepblue_select_column

**Examples**  
```
deepblue_diff(
    id1 = "e16918",
    id2 = "e16919")
```

---

deppeblue_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**  
```
deppeblue_download_request_data(request_id,  
    user_key = deepblue_options("user_key"),  
    force_download = deepblue_options("force_download"),  
    do_not_cache = deepblue_options("do_not_cache"))
```
**Arguments**

- `request_id` - Id of the request that will be downloaded
- `user_key` - A string
- `force_download` - forces DeepBlueR to download the request overwriting any results that might already be in the cache
- `do_not_cache` - whether to use local caching of requests

**Value**

grange_regions Final output in GRanges format or as data frame

**Examples**

```r
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_download_request_data(request_id)
```

---

**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**

```r
depthblue_echo(user_key = depthblue_options("user_key"))
```

**Arguments**

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

**Value**

- `message` - A string (echo message including version)

**See Also**

Other Checking DeepBlue status: `depthblue_commands`

**Examples**

```r
depthblue_echo(user_key = "anonymous_key")
```

---

**deepblue_export_bed**  
*Export GenomicRanges result as BED file*

**Description**

Export GenomicRanges result as BED file

**Usage**

```r
depthblue_export_bed(result, target.directory = "./", file.name, score.field = NULL)
```

**Arguments**

- `result` - A result from a DeepBlue request such as a set of genomic regions.
- `target.directory` - The directory to save the file to
- `file.name` - The name of the file without suffix
- `score.field` - Which column of the results should be used to populate the score column of the BED file (optional)
deepblue_export_meta_data

Value
return value of write.table

Examples
query_id = deepblue_select_experiments (experiment=c("GC_T14_10.CPG_methylation_calls.bs_call.GRCh38.20160531.wig"), chromosome="chr1", start=0, end=50000000)
cpg_islands = deepblue_select_annotations(abbreviation="CpG Islands", genome="GRCh38", chromosome="chr1", start=0, end=50000000)
overlapped = deepblue_aggregate (data_id=query_id, ranges_id=cpg_islands, column="VALUE")
request_id = deepblue_get_request (query_id=overlapped,
output_format= "CHROMOSOME,START,END,@AGG_MIN,@AGG_MAX,@AGG_MEAN,@AGG_VAR")
regions = deepblue_download_request_data(request_id=request_id)
temp_dir = tempdir()
depthblue_export_bed(regions, target.directory = temp_dir,
file.name = "GC_T14_10.CpG_islands")

---

depthblue_export_meta_data

Export meta data as tab delimited file

Description
Export meta data as tab delimited file

Usage
depthblue_export_meta_data(ids, target.directory = "./", file.name,
user_key = depthblue_options("user_key"))

Arguments
ids an id or a list of DeepBlue ids
target.directory where the meta data should be stored
file.name name of the file
user_key DeepBlue user key

Value
return value of write.table

Examples
depthblue_export_meta_data(list("e30035", "e30036"),
file.name = "test_export",
target.directory = tempdir())
deepblue_export_tab  Export a DeepBlue result as ordinary tab delimited file

Description
Export a DeepBlue result as ordinary tab delimited file

Usage
deepblue_export_tab(result, target.directory = "./", file.name)

Arguments
result  A result from a DeepBlue request such as a set of genomic regions.
target.directory  The directory to save the file to
file.name  The name of the file without suffix

Value
return value of write.table

Examples
query_id = deepblue_select_experiments(
  experiment=c("GC_T14_10.CPG_methylation_calls.bscall.GRCH38.20160531.wig"),
  chromosome="chr1", start=0L, end=50000000L)
cpg_islands = deepblue_select_annotations(annotation_name="CpG Islands",
genome="GRCh38", chromosome="chr1", start=0L, end=50000000L)
overlapped = deepblue_aggregate(data_id=query_id, ranges_id=cpg_islands,
column="VALUE")
request_id = deepblue_get_regions(query_id=overlapped,
  output_format="CHROMOSOME,START,END,@AGG.MIN,@AGG.MAX,@AGG.MEAN,@AGG.VAR")
regions = deepblue_download_request_data(request_id=request_id)
temp_dir = tempdir()
deepblue_export_tab(regions, target.directory = temp_dir,
  file.name = "GC_T14_10.CpG_islands")

depthblue_extend  extend

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

Usage
deepblue_extend(query_id = NULL, length = NULL, direction = NULL,
  use_strand = NULL, user_key = deepblue_options("user_key"))
**deepblue_extract_ids**

**Arguments**

- **query_id** - A string (Query ID)
- **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_binning, deepblue_calculate_enrichment, deepblue_count_regions, deepblue_coverage, deepblue_filter_regions, deepblue_flank, deepblue_get_experiments_by_query, deepblue_get_regions, deepblue_input_region, deepblue_intersect, deepblue_merge_queries, deepblue_overlap, 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")
depth Extend(query_id = annotation_id,
  length = 2000, direction = "BOTH",
  use_strand = TRUE)
```

---

**deepblue_extract_ids**

**extract_ids**

**Description**

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

**Usage**

```r
depth Extract_ids(df = NULL)
```

**Arguments**

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

**Value**

- **ids** - A vector containing the extracted IDs)
deepblue_extract_names

See Also

Other Utilities for connecting operations: deepblue_extract_names

Examples

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

depthblue_extract_names

```r
extract_names
```

Description

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

Usage

```r
depthblue_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: depthblue_extract_ids

Examples

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

deeperblue_faceting_experiments(genome = NULL, type = NULL, epigenetic_mark = NULL, biosource = NULL, sample = NULL, technique = NULL, project = NULL, user_key = depeblue_options("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: depeblue_collection_experiments_count, depeblue_list_experiments, depeblue_list_recent_experiments, depeblue_list_similar_experiments, depeblue_preview_experiment

Examples

deeperblue_faceting_experiments(genome = "hg19", type = "peaks", biosource = "blood")
deepblue_filter_regions

Description

Filter the genomic regions by their content.

Usage

```
depthblue_filter_regions(query_id = NULL, field = NULL, operation = NULL, value = NULL, type = NULL, user_key = deepblue_options("user_key"))
```

Arguments

- **query_id**: A string (Query ID)
- **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: deepblue_aggregate, deepblue_binning, deepblue_calculate_enrichment, deepblue_count_regions, deepblue_coverage, deepblue_extend, deepblue_flank, deepblue_get_experiments, deepblue_get_regions, deepblue_input_regions, deepblue_intersection, deepblue_merge_queries, deepblue_overlap, deepblue_query_cache, deepblue_query_experiment_type, deepblue_score_matrix, deepblue_select_annotations, deepblue_select_experiments, deepblue_select_regions, deepblue_tiling_regions

Examples

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

---

**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_options("user_key"))
```

**Arguments**

- `query_id` - A string (Query ID)
- `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_binning`, `deepblue_calculate_enrichment`, `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_overlap`, `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")
deeble_flank(query_id = annotation_id,
              start = 0, length = 2000,
              use_strand = TRUE)
```
deepblue_format_object_size

*Format byte size as human readable units*

**Description**

Format byte size as human readable units

**Usage**

```r
deepblue_format_object_size(x, units = "b")
```

**Arguments**

- `x` size in bytes
- `units` target unit or 'auto'

**Value**

formatted size

**Source**

`utils:::format.object_size`

---

deepblue_get_biosource_children

*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_options("user_key"))
```

**Arguments**

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

**Value**

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

**See Also**

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

**Examples**

```r
depthblue_get_biosource_parents(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 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
depthblue_get_biosource_parents(biosource = NULL,
       user_key = depthblue_options("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**

```r
depthblue_get_biosource_parents(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

```r
deepblue_get_biosource_related(biosource = NULL,
user_key = deepblue_options("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_parents`, `deepblue_get_biosource_synonyms`

## Examples

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

---

## 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_options("user_key"))
```
**deepblue_get_db**

**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_db**

Sets up the DeepBlueR cache and returns a filehash db object.

**Description**

Sets up the DeepBlueR cache and returns a filehash db object.

**Usage**

```r
depthblue_get_db()
```

**Value**

A filehash package database.

---

**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
depthblue_get_experiments_by_query(query_id = NULL, user_key = depthblue_options("user_key"))
```

**Arguments**

- query_id: A string (Query ID)
- 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_binning, deepblue_calculate_enrichment, deepblue_count_regions, deepblue_coverage, deepblue_extend, deepblue_filter_regions, deepblue_flank, deepblue_get_regions, deepblue_input_regions, deepblue_intersection, deepblue_merge_queries, deepblue_overlap, deepblue_query_cache, deepblue_query_experiment_type, deepblue_score_matrix, deepblue_select_annotations, deepblue_select_experiments, deepblue_select_regions, deepblue_tiling_regions

Examples

depthblue_get_experiments_by_query(query_id = "q12345")

depthblue_get_regions

depthblue_get_regions  get_regions

depthblue_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 download using the URL: http://deepblue.mpi-inf.mpg.de/download?r_id=<request_id>&key=<user_key>.

Usage

depthblue_get_regions(query_id = NULL, output_format = NULL, user_key = depthblue_options("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_binning, deepblue_calculate_enrichment, deepblue_count_regions, deepblue_coverage, deepblue_extend, deepblue_filter_regions, deepblue_flank, deepblue_get_experiments_by_query, deepblue_input_regions, deepblue_intersection, deepblue_merge_queries, deepblue_overlap, deepblue_query_cache, deepblue_query_experiment_type, deepblue_score_matrix, deepblue_select_annotations, deepblue_select_experiments, deepblue_select_regions, deepblue_tiling_regions
**deepblue_get_request_data**

**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")
```

**deepblue_get_request_data**

get_request_data

**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_options("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`

**Examples**

```r
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")
deepblue_get_request_data(request_id = request_id)
```
**deepblue_info**  
*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_options("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**  
*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_options("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_binning`, `deepblue_calculate_enrichment`, `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_overlap`, `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. This command is a simplified version of the ‘overlap’ command.

**Usage**

```r
deepblue_intersection(query_data_id = NULL, query_filter_id = NULL,
user_key = deepblue_options("user_key"))
```
**deepblue_is_biosource**

**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_binning, deepblue_calculate_enrichment, 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_overlap, 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")
data_id = deepblue_select_experiments(
    experiment_name="E002-H3K9ac.narrowPeak.bed")
deeblue_intersection(query_data_id = annotation_id, query_filter_id = data_id)
```

**deepblue_is_biosource is_biosource**

**Description**

Verify if the name is an existing and valid DeepBlue BioSource name. A BioSource refers to a term describing the origin of a given sample, such as a tissue or cell line.

**Usage**

```r
deeblue_is_biosource(biosource = NULL, user_key = deepblue_options("user_key"))
```

**Arguments**

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

**Value**

- **information** - A string or a vector of string (A string containing the biosource name)
deepblue_liftover

See Also

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

Examples

deeplblue_is_biosource(biosource = "blood")

deepblue_liftover(regions, source = "hg19", target = "hg38", collapse = TRUE)

Description

This is a wrapper function for the liftOver function found in the rtracklayer package. For common genome assemblies available in DeepBlue, this function automatically downloads the necessary chain file and calls liftOver.

Usage

deeplblue_liftover(regions, source = "hg19", target = "hg38", collapse = TRUE)

Arguments

regions The GRanges object to lift over to another assembly
source The source assembly version, e.g. hg38. If NULL, we try to read the genome version from the GRanges object.
target The target assembly version, e.g. hg19. Required.
collapse Whether to return a single GRanges object or a list of GRanges (one per region in the input). The latter is the default behavior of liftOver since multiple assignments are possible.

Value

A GRanges object using the target chromosome positions

Examples

data_id = deeplblue_select_experiments(
  experiment_name="E002-H3K9ac.narrowPeak.bed", chromosome="chr1")
request_id = deeplblue_get Regions(query_id =data_id,
  output_format = "CHROMOSOME,START,END")
request_data = deeplblue_download_request_data(request_id)
deepblue_liftover(request_data, source = "hg38", target = "hg19")
**deepblue_list_annotations**

`list_annotations`

**Description**

List all annotations of genomic regions currently available in DeepBlue.

**Usage**

```r
depthblue_list_annotations(genome = NULL, 
user_key = depthblue_options("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**

```r
depthblue_list_annotations(genome = "hg19")
```

---

**deepblue_list_biosources**

`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**

```r
depthblue_list_biosources(extra_metadata = NULL, 
user_key = depthblue_options("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)
deepblue_list_cached_requests

See Also
Other Inserting and listing biosources: deepblue_list_similar_biosources

Examples

depthue_list_biosources(extra_metadata = list(ontology_id = "UBERON:0002485"))

depthue_list_cached_requests

List cached requests

Description
List cached requests

Usage

depthue_list_cached_requests()

Value
list of request ids that are cached

Examples

depthue_list_cached_requests()

depthue_list_column_types

list_column_types

Description
Lists the ColumnTypes included in DeepBlue.

Usage

depthue_list_column_types(user_key = depthue_options("user_key"))

Arguments
user_key - A string (users token key)

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

Examples

depthue_list_column_types()
deepblue_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
deepblue_list_epigenetic_marks(extra_metadata = NULL, 
user_key = deepblue_options("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()

depthblue_list_experiments

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

Usage
depthblue_list_experiments(genome = NULL, type = NULL, 
epigenetic_mark = NULL, biosource = NULL, sample = NULL, 
technique = NULL, project = NULL, 
user_key = deepblue_options("user_key"))
**deepblue_list_expressions**

### 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")
```

---

**deepblue_list_expressions**

**list_expressions**

### 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_options("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**

deeperblue_list_expressions(expression_type='gene')

deeperblue_list_genes(genes = NULL, go_terms = NULL, chromosome = NULL, start = NULL, end = NULL, gene_model = NULL, user_key = deeperblue_options("user_key"))

**Description**

List the Genes currently available in DeepBlue.

**Usage**

deeperblue_list_genes(genes = NULL, go_terms = NULL, chromosome = NULL, start = NULL, end = NULL, gene_model = NULL, user_key = deeperblue_options("user_key"))

**Arguments**

genes - A string or a vector of string (Name(s) or ENSEMBL ID (ENSGXXXXXXXXXXX.X ) of the gene(s).)
go_terms - A string or a vector of string (gene ontology terms - ID or label)
chromosome - A string or a vector of string (chromosome name(s))
start - A int (minimum start region)
end - A int (maximum end region)
gene_model - A 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_count_gene_ontology_terms, deepblue_list_gene_models, deepblue_select_genes`

**Examples**

deeperblue_list_genes(
    chromosome="chr20",
    start=10000000,
    end=21696620,
gene_model='Gencode v22')
**deepblue_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*

```r
deepblue_list_gene_models(user_key = deepblue_options("user_key"))
```

*Arguments*

- `user_key` - A string (user's token key)

*Value*

- `gene_models` - A array (gene models names and IDs)

*See Also*

Other Gene models and genes identifiers: deepblue_count_gene_ontology_terms, deepblue_list_genes, deepblue_select_genes

*Examples*

```r
deepblue_list_gene_models()
```

---

**deepblue_list_genomes**

*Description*

List Genomes assemblies that are registered in DeepBlue.

*Usage*

```r
deepblue_list_genomes(user_key = deepblue_options("user_key"))
```

*Arguments*

- `user_key` - A string (user's token key)

*Value*

- `genomes` - A array (genome names)
See Also

Other Inserting and listing genomes: deepblue_chromosomes, deepblue_list_similar_genomes

Examples

depthblue_list_genomes()
**deepblue_list_projects**

**list_projects**

---

**Description**

List Projects included in DeepBlue.

**Usage**

```r
deepblue_list_projects(user_key = deepblue_options("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**

```r
deepblue_list_projects()
```

---

**deepblue_list_recent_experiments**

**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**

```r
deepblue_list_recent_experiments(days = NULL, genome = NULL, epigenetic_mark = NULL, sample = NULL, technique = NULL, project = NULL, user_key = deepblue_options("user_key"))
```
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(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 (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

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

depthblue_list_requests(request_state = 'running')

---

depthblue_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**

```r
depthblue_list_samples(biosource = NULL, extra_metadata = NULL,
user_key = depthblue_options("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**

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

---

depthblue_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**

```r
depthblue_list_similar_biosources(name = NULL,
user_key = depthblue_options("user_key"))
```
Arguments

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

Value

- biosource: A string (biosource name)

See Also

Other Inserting and listing biosources: deepblue_list_biosources

Examples

deeperblue_list_similar_biosources(name = "blood")

深蓝色列表相似的表观遗传标记

Description

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

Usage

deeperblue_list_similar_epigenetic_marks(name = NULL,
user_key = deeperblue_options("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: deeperblue_list_epigenetic_marks

Examples

deeperblue_list_similar_epigenetic_marks(name = "H3k27ac")
deepblue_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
deepblue_list_similar_experiments(name = NULL, genome = NULL, user_key = deepblue_options("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
deepblue_list_similar_experiments(name = "blood", genome = "hg19")

depthblue_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
deepblue_list_similar_genomes(name = NULL, user_key = deepblue_options("user_key"))
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

deepblue_list_similar_genomes(name = "grc")

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

default

deepblue_list_similar_projects

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

Usage

deepblue_list_similar_projects(name = NULL, 
user_key = deepblue_options("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

deepblue_list_similar_projects(name = "BLUEPRINT")
**deepblue_list_similar_techniques**

*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**
```
depth_list_similar_techniques(name = NULL, user_key = depth_options("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**
```
depth_list_similar_techniques(name = "chip seq")
```

---

**deepblue_list_techniques**

*list_techniques*

**Description**
List the Techniques included in DeepBlue.

**Usage**
```
depth_list_techniques(user_key = depth_options("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**

deepblue_list_techniques()

```r

---

depthblue_merge_queries

merge_queries

---

**Description**

Merge regions from two queries in a new query.

**Usage**

```r
depthblue_merge_queries(query_a_id = NULL, query_b_id = NULL, 
user_key = depthblue_options("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: depthblue_aggregate, depthblue_binning, depthblue_calculate_enrichment, 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_overlap, depthblue_query_cache, depthblue_query_experiment_type, depthblue_score_matrix, depthblue_select_annotations, depthblue_select_experiments, depthblue_select_regions, depthblue_tiling_regions

**Examples**

```r

annotation_id = depthblue_select_annotations( 
    annotation_name="CpG Islands", 
    genome="hg19", chromosome="chr1")
data_id = depthblue_select_experiments( 
    experiment_name="E002-H3K9ac.narrowPeak.bed")
depthblue_merge_queries( 
    query_a_id = annotation_id, 
    query_b_id = data_id)
```
**deepblue_meta_data_to_table**

*Convert XML structured meta data to table format*

**Description**

Convert XML structured meta data to table format

**Usage**

```
deepblue_meta_data_to_table(ids, user_key = deepblue_options("user_key"))
```

**Arguments**

- **ids**
  - an id or a list of ids

- **user_key**
  - a DeepBlue user key (optional for public data)

**Value**

- a data frame with meta data

**Examples**

# works for sample ids
deepblue_meta_data_to_table(list("s2694", "s2695"))

# or experiment ids
deepblue_meta_data_to_table(list("e30035", "e30036"))

**name_to_id**

**Description**

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

**Usage**

```
deepblue_name_to_id(name = NULL, collection = NULL,
                    user_key = deepblue_options("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**

- information - 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")
```

deprecated

deepblue_overlap

Description

Select genomic regions that overlap or not overlap with with the specified number of regions of the second query. Important: This command is still experimental and changes may occur.

Usage

```r
deepblue_overlap(query_data_id = NULL, query_filter_id = NULL,
  overlap = NULL, amount = NULL, amount_type = NULL,
  user_key = deepblue_options("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.)
- **overlap**: A boolean (True if must overlap, or false if must not overlap.)
- **amount**: A int (Amount of regions that must overlap. Use the parameter ‘amount_type’ (‘bp’ or ‘%’) to specify the unit. For example, use the value ‘10’ with the amount_type ‘%’ to specify that 10% of the bases in both regions must overlap, or use ‘10’ with the amount_type ‘bp’ to specify that at least 10 bases must or must not overlap.)
- **amount_type**: A string (Type of the amount: ‘bp’ for base pairs and ‘%’ for percentage.)
- **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_binning, deepblue_calculate_enrichment, 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

```python
annotation_id = deepblue_select_annotations(
    annotation_name="CpG Islands",
    genome="hg19", chromosome="chr1")

experiment_id = deepblue_select_experiments(
    experiment_name="S00XDKH1.ERX712765.H3K27ac.bwa.GRCh38.20150527.bed")

deepblue_overlap(query_data_id = experiment_id, query_filter_id = annotation_id,
    overlap = TRUE, amount=10, amount_type="%")
```

Description

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

**Usage**
```
depthblue_preview_experiment(experiment_name = NULL, 
    user_key = depthblue_options("user_key"))
```

**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("/quotesingle.VarS00JJRH1.ERX683143.H3K4me3.bwa.GRCh38.20150527.bed")
```

**deepblue_query_cache**  
**query_cache**

**Description**
Cache a query result in DeepBlue memory. This command is useful when the same query ID is used multiple times in different requests. The command is an advice for DeepBlue to cache the query result and there is no guarantee that this query data access will be faster.

**Usage**
```
depthblue_query_cache(query_id = NULL, cache = NULL, 
    user_key = depthblue_options("user_key"))
```

**Arguments**
- **query_id**
  - A string (Query ID)
- **cache**
  - A boolean (set or unset this query caching)
- **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_binning, deepblue_calculate_enrichment, 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_overlap, 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")
    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

    deepblue_query_experiment_type(query_id = NULL, type = NULL,
        user_key = deepblue_options("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.)
deepblue_reset_options

Reset DeepBlueR options

Description
Reset DeepBlueR options

Usage
deepblue_reset_options(new_options = NULL)

Arguments
new_options: list of new options that should be used. default options if NULL

Value
new (default) options

Examples
deepblue_reset_options()
**deepblue_score_matrix**  

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

**Usage**  

```r  
depdeepblue_score_matrix(experiments_columns = NULL,  
aggregation_function = NULL, aggregation_regions_id = NULL,  
user_key = deepblue_options("user_key"))  
```

**Arguments**  
- **experiments_columns**  
  - A struct (map with experiments names and columns to be processed. Example: `'wgEncodeBroadHistoneDnd41H3k27acSig.wig':'VALUE', 'wgEncodeBroadHistoneCd20ro01794H3k27acSig.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_binning, deepblue_calculate_enrichment, 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_overlap, deepblue_query_cache, deepblue_query_experiment_type, deepblue_select_annotations, deepblue_select_experiments, deepblue_select_regions, deepblue_tiling_regions

**Examples**  

```r  
tiling_regions = deepblue_tiling_regions(  
  size=100000, genome="hg19", chromosome="chr1")  
depdeepblue_score_matrix(  
  experiments_columns =  
    list(ENCFF721EKA="VALUE", ENCFF781VVH="VALUE"),  
  aggregation_function = "mean",  
  aggregation_regions_id = tiling_regions)  
```
deepblue_search

search

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, Gene_ontology, Projects, Samples, Techniques, Tilings.

Usage

depthblue_search(keyword = NULL, type = NULL,
user_key = deepblue_options("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, Gene_ontology, 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

depthblue_search(keyword = "DNA Methylation BLUEPRINT",
type = "experiments")

deepblue_select_annotations

select_annotations

Description

Select regions from the Annotations that match the selection criteria.

Usage

depthblue_select_annotations(annotation_name = NULL, genome = NULL,
chromosome = NULL, start = NULL, end = NULL,
user_key = deepblue_options("user_key"))
**deepblue_select_column**

**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_binning, deepblue_calculate_enrichment, 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_overlap, 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)
```

---

**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**

```r
deepblue_select_column(experiments, column,
    user_key = deepblue_options("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 (users token key)
**Value**

A list of experiments with the selected column

**See Also**

- `deepblue_score_matrix`
- `deepblue_list_experiments`

Other Utilities for information processing: `deepblue_diff`

**Examples**

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

blueprint_DNA_meth <- blueprint_DNA_meth[grepl("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
deepblue_select_experiments(experiment_name = NULL, chromosome = NULL,
  start = NULL, end = NULL, user_key = deepblue_options("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: `deepblue_aggregate`, `deepblue_binning`, `deepblue_calculate_enrichment`, `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_overlap`, `deepblue_query_cache`, `deepblue_query_experiment_type`, `deepblue_score_matrix`, `deepblue_select_annotations`, `deepblue_select_regions`, `deepblue_tiling_regions`

Examples

```r
deepblue_select_experiments(  
extperiment_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

```r
deepblue_select_expressions(expression_type = NULL, sample_ids = NULL,  
replicas = NULL, identifiers = NULL, projects = NULL,  
gene_model = NULL, user_key = deepblue_options("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(c("CCR1", "CD164", "CD1D", "CD2", "CD34", "CD3G", "CD44")
deepblue_select_expressions(
    expression_type="gene",
    sample_ids="s10205",
    identifiers = genes_names,
    gene_model = "gencode v23")

deepblue_select_genes  select_genes

Description

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

Usage

deepblue_select_genes(genes = NULL, go_terms = NULL, gene_model = NULL,
    chromosome = NULL, start = NULL, end = NULL,
    user_key = deepblue_options("user_key"))

Arguments

genes - A string or a vector of string (Name(s) or ENSEMBL ID (ENSGXXXXXXXXXXXX.X
     ) of the gene(s).)
go_terms - A string or a vector of string (gene ontology terms - ID or label)
gene_model - A string (the gene model)
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_count_gene_ontology_terms, deepblue_list_gene_models, deepblue_list_genes

Examples

genes_names =
c(c("CCR1", "CD164", "CD1D", "CD2", "CD34", "CD3G", "CD44")
deepblue_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

depthblue_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_options("user_key"))

Arguments

eperiment_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_binning, deepblue_calculate_enrichment, 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_overlap, deepblue_query_cache, deepblue_query_experiment_type, deepblue_score_matrix, deepblue_select_annotations, deepblue_select_experiments, deepblue_tiling_regions

Examples

depthblue_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

```r
deepblue_tiling_regions(size = NULL, genome = NULL, chromosome = NULL, user_key = deepblue_options("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_binning`, `deepblue_calculate_enrichment`, `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_overlap`, `deepblue_query_cache`, `deepblue_query_experiment_type`, `deepblue_score_matrix`, `deepblue_select_annotations`, `deepblue_select_experiments`, `deepblue_select_regions`

### Examples

```r
deepblue_tiling_regions(
  size = 10000,
  genome = "hg19",
  chromosome = "chr1")
```
**Description**

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

**Usage**

```python
deepblue_wait_request(request_id, sleep_time = 1,
                      user_key = deepblue_options("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

deeplearning_aggregate, 4, 6, 7, 12, 13, 19, 22, 23, 28, 31, 32, 48, 51, 53–55, 57, 59, 61, 62
deeplearning_batch_export_results, 5
deeplearning_binning, 4, 5, 7, 12, 13, 19, 22, 23, 28, 31, 32, 48, 51, 53–55, 57, 59, 61, 62
deeplearning_cache_status, 6
deeplearning_calculate_enrichment, 4, 6, 7, 12, 13, 19, 22, 23, 28, 31, 32, 48, 51, 53–55, 57, 59, 61, 62
deeplearning_cancel_request, 8, 30, 33, 40, 50, 56
deeplearning_chromosomes, 8, 40, 46
deeplearning_clear_cache, 9
deeplearning_collection_experiments_count, 9, 21, 37, 42, 45, 52
deeplearning_commands, 10, 16
deeplearning_count_gene_ontology_terms, 11, 38, 39, 60
deeplearning_count_regions, 4, 6, 7, 12, 13, 19, 22, 23, 28, 31, 32, 48, 51, 53–55, 57, 59, 61, 62
deeplearning_coverage, 4, 6, 7, 12, 13, 19, 22, 23, 28, 31, 32, 48, 51, 53–55, 57, 59, 61, 62
deeplearning_delete_request_from_cache, 13
deeplearning_diff, 14, 58
deeplearning_download_request_data, 14
deeplearning_download_request_data, DeepBlueCommand-method, 15
deeplearning_echo, 10, 16
deeplearning_export_bed, 16
deeplearning_export_meta_data, 17
deeplearning_export_tab, 18
deeplearning_extend, 4, 6, 7, 12, 13, 18, 22, 23, 28, 31, 32, 48, 51, 53–55, 57, 59, 61, 62
deeplearning_extract_ids, 19, 20
deeplearning_extract_names, 20, 20
deeplearning_faceting_experiments, 10, 21, 37, 42, 45, 52
deeplearning_filter_regions, 4, 6, 7, 12, 13, 19, 22, 23, 28, 31, 32, 48, 51, 53–55, 57, 59, 61, 62
deeplearning_flank, 4, 6, 7, 12, 13, 19, 22, 23, 28, 31, 32, 48, 51, 53–55, 57, 59, 61, 62
deeplearning_format_object_size, 24
deeplearning_get_biosource_children, 24, 25–27
deeplearning_get_biosource_parents, 25, 25, 26, 27
deeplearning_get_biosource_related, 25, 26, 27
deeplearning_get_biosource_synonyms, 25, 26, 26
deeplearning_get_db, 27
deeplearning_get_experiments_by_query, 4, 6, 7, 12, 13, 19, 22, 23, 27, 28, 31, 32, 48, 51, 53–55, 57, 59, 61, 62
deeplearning_get_regions, 4, 6, 7, 12, 13, 19, 22, 23, 28, 31, 32, 48, 51, 53–55, 57, 59, 61, 62
deeplearning_get_request_data, 29, 42
deeplearning_info, 8, 30, 33, 40, 50, 56
deeplearning_input_regions, 4, 6, 7, 12, 13, 19, 22, 23, 28, 30, 32, 48, 51, 53–55, 57, 59, 61, 62
deeplearning_intersection, 4, 6, 7, 12, 13, 19, 22, 23, 28, 31, 32, 48, 51, 53–55, 57, 59, 61, 62
deeplearning_is_biosource, 8, 30, 32, 40, 50, 56
deeplearning_liftover, 33
deeplearning_list_annotations, 34
deeplearning_list_biosources, 34, 44
deeplearning_list_cached_requests, 35
deeplearning_list_column_types, 35
deeplearning_list_epigenetic_marks, 36, 44
deeplearning_list_experiments, 10, 21, 36, 42, 45, 52, 58
deeplearning_list_expressions, 37, 59
deeplearning_list_gene_models, 11, 38, 39, 60
deeplearning_list_genes, 11, 38, 39, 60
deeplearning_list_genomes, 9, 39, 46
deeplearning_list_in_use, 8, 30, 33, 40, 50, 56
INDEX

deepblue_list_projects, 41, 46
deepblue_list_recent_experiments, 10, 21, 37, 41, 45, 52
deeplblue_list_requests, 29, 42
deeplblue_list_samples, 43
deeplblue_list_similar_biosources, 35, 43
deeplblue_list_similar_epigenetic_marks, 36, 44
deeplblue_list_similar_experiments, 10, 21, 37, 42, 45, 52
deeplblue_list_similar_genomes, 9, 40, 45
deeplblue_list_similar_projects, 41, 46
deeplblue_list_similar_techniques, 47, 48
deeplblue_list_techniques, 47, 47
deeplblue_merge_queries, 4, 6, 7, 12, 13, 19, 22, 23, 28, 31, 32, 48, 51, 53–55, 57, 59, 61, 62
deeplblue_meta_data_to_table, 49
deeplblue_name_to_id, 8, 30, 33, 40, 49, 56
deeplblue_options, 50
deeplblue_overlap, 4, 6, 7, 12, 13, 19, 22, 23, 28, 31, 32, 48, 50, 53–55, 57, 59, 61, 62
deeplblue_preview_experiment, 10, 21, 37, 42, 45, 51
deeplblue_query_cache, 4, 6, 7, 12, 13, 19, 22, 23, 28, 31, 32, 48, 51, 52, 54, 55, 57, 59, 61, 62
deeplblue_query_experiment_type, 4, 6, 7, 12, 13, 19, 22, 23, 28, 31, 32, 48, 51, 53, 53, 55, 57, 59, 61, 62
deeplblue_reset_options, 54
deeplblue_score_matrix, 4, 6, 7, 12, 13, 19, 22, 23, 28, 31, 32, 48, 51, 53, 54, 55, 57–59, 61, 62
deeplblue_search, 8, 30, 33, 40, 50, 56
deeplblue_select_annotations, 4, 6, 7, 12, 13, 19, 22, 23, 28, 31, 32, 48, 51, 53–55, 56, 59, 61, 62
deeplblue_select_column, 14, 57
deeplblue_select_experiments, 4, 6, 7, 12, 13, 19, 22, 23, 28, 31, 32, 48, 51, 53–55, 57, 58, 61, 62
deeplblue_select_expressions, 38, 59
deeplblue_select_genes, 11, 38, 39, 60
deeplblue_select_regions, 4, 6, 7, 12, 13, 19, 22, 23, 28, 31, 32, 48, 51, 53–55, 57, 59, 61, 62
deeplblue_tiling_regions, 4, 6, 7, 12, 13, 19, 22, 23, 28, 31, 32, 48, 51, 53–55,

DeepBlueCommand
DeepBlueCommand-class, 3
DeepBlueCommand-class, 3