

# Package ‘DeepBlueR’

October 17, 2017

**Title** DeepBlueR

**Type** Package

**Description** Accessing the DeepBlue Epigenetics Data Server through R.

**Version** 1.2.10

**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

**Collate** options.R class\_DeepBlueCommand.R deepblue.R internalFunctions.R export.R liftover.R helperFunctions.R methods\_DeepBlueCommand.R zzz.R caching.R

**BuildVignettes** TRUE

**RoxygenNote** 6.0.1

**Suggests** knitr, rmarkdown, LOLA, Gviz, gplots, ggplot2, tidyr, RColorBrewer, matrixStats

**VignetteBuilder** knitr

**biocViews** DataImport, DataRepresentation, ThirdPartyClient, GeneRegulation, GenomeAnnotation, CpGIIsland, DNAMethylation, Epigenetics, Annotation, Preprocessing

**NeedsCompilation** no

## R topics documented:

DeepBlueCommand-class . . . . .	3
deepblue_aggregate . . . . .	4
deepblue_batch_export_results . . . . .	5
deepblue_binning . . . . .	5
deepblue_cache_status . . . . .	6
deepblue_cancel_request . . . . .	7
deepblue_chromosomes . . . . .	7
deepblue_clear_cache . . . . .	8

deepblue_collection_experiments_count . . . . .	8
deepblue_commands . . . . .	9
deepblue_count_gene_ontology_terms . . . . .	10
deepblue_count_regions . . . . .	11
deepblue_coverage . . . . .	11
deepblue_delete_request_from_cache . . . . .	12
deepblue_diff . . . . .	13
deepblue_distinct_column_values . . . . .	13
deepblue_download_request_data . . . . .	14
deepblue_download_request_data,DeepBlueCommand-method . . . . .	15
deepblue_echo . . . . .	15
deepblue_enrich_regions_go_terms . . . . .	16
deepblue_enrich_region_overlap . . . . .	17
deepblue_export_bed . . . . .	18
deepblue_export_meta_data . . . . .	19
deepblue_export_tab . . . . .	19
deepblue_extend . . . . .	20
deepblue_extract_ids . . . . .	21
deepblue_extract_names . . . . .	22
deepblue_faceting_experiments . . . . .	22
deepblue_filter_regions . . . . .	23
deepblue_find_motif . . . . .	24
deepblue_flank . . . . .	25
deepblue_format_object_size . . . . .	26
deepblue_get_biosource_children . . . . .	26
deepblue_get_biosource_parents . . . . .	27
deepblue_get_biosource_related . . . . .	28
deepblue_get_biosource_synonyms . . . . .	28
deepblue_get_db . . . . .	29
deepblue_get_experiments_by_query . . . . .	29
deepblue_get_regions . . . . .	30
deepblue_get_request_data . . . . .	31
deepblue_info . . . . .	32
deepblue_input_regions . . . . .	32
deepblue_intersection . . . . .	33
deepblue_is_biosource . . . . .	34
deepblue_liftover . . . . .	35
deepblue_list_annotations . . . . .	36
deepblue_list_biosources . . . . .	36
deepblue_list_cached_requests . . . . .	37
deepblue_list_column_types . . . . .	37
deepblue_list_epigenetic_marks . . . . .	38
deepblue_list_experiments . . . . .	38
deepblue_list_expressions . . . . .	39
deepblue_list_genes . . . . .	40
deepblue_list_gene_models . . . . .	41
deepblue_list_genomes . . . . .	41
deepblue_list_in_use . . . . .	42
deepblue_list_projects . . . . .	43
deepblue_list_recent_experiments . . . . .	43
deepblue_list_requests . . . . .	44
deepblue_list_samples . . . . .	45

deepblue_list_similar_biosources . . . . .	45
deepblue_list_similar_epigenetic_marks . . . . .	46
deepblue_list_similar_experiments . . . . .	47
deepblue_list_similar_genomes . . . . .	47
deepblue_list_similar_projects . . . . .	48
deepblue_list_similar_techniques . . . . .	49
deepblue_list_techniques . . . . .	49
deepblue_merge_queries . . . . .	50
deepblue_meta_data_to_table . . . . .	51
deepblue_name_to_id . . . . .	51
deepblue_options . . . . .	52
deepblue_overlap . . . . .	52
deepblue_preview_experiment . . . . .	53
deepblue_query_cache . . . . .	54
deepblue_query_experiment_type . . . . .	55
deepblue_reset_options . . . . .	56
deepblue_score_matrix . . . . .	57
deepblue_search . . . . .	58
deepblue_select_annotations . . . . .	58
deepblue_select_column . . . . .	59
deepblue_select_experiments . . . . .	60
deepblue_select_expressions . . . . .	61
deepblue_select_genes . . . . .	62
deepblue_select_regions . . . . .	63
deepblue_tiling_regions . . . . .	64
deepblue_wait_request . . . . .	65

**Index****66**


---

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 managin 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

```
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\\_count\\_regions](#), [deepblue\\_coverage](#), [deepblue\\_distinct\\_column\\_values](#), [deepblue\\_extend](#), [deepblue\\_filter\\_regions](#), [deepblue\\_flank](#), [deepblue\\_get\\_experiments\\_by\\_query](#), [deepblue\\_get\\_regions](#), [deepblue\\_input\\_regions](#), [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\\_re](#), [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  
*batch\_export\_results*

---

### Description

Write results from DeepBlue to disk as they become available

### Usage

```
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

```
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))
```

---

deepblue\_binning      *binning*

---

### Description

Bin results according to counts.

### Usage

```
deepblue_binning(query_data_id = NULL, column = NULL, bins = NULL,
  user_key = deepblue_options("user_key"))
```

**Arguments**

query\_data\_id - A string (query data that will made by the binning.)  
 column - A string (name of the column that will be used in the aggregation)  
 bins - A int (number of of bins)  
 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](#), [deepblue\\_distinct\\_column\\_values](#), [deepblue\\_extend](#), [deepblue\\_filter\\_regions](#), [deepblue\\_flank](#), [deepblue\\_get\\_experiments\\_by\\_query](#), [deepblue\\_get\\_regions](#), [deepblue\\_input\\_regions](#), [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\\_re](#), [deepblue\\_tiling\\_regions](#)

**Examples**

```

experiment_id = deepblue_select_experiments(
    experiment_name="S00XDKH1.ERX712765.H3K27ac.bwa.GRCh38.20150527.bed")
deepblue_binning (query_data_id=experiment_id,
    column="SIGNAL_VALUE",
    bins=40)
  
```

---

deepblue\_cache\_status *Report on the cache size and status*

---

**Description**

Report on the cache size and status

**Usage**

```
deepblue_cache_status()
```

**Value**

cache size in byte

**Examples**

```
deepblue_cache_status()
```

---

deepblue\_cancel\_request     *cancel\_request*

---

**Description**

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

**Usage**

```
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**

```
deepblue_cancel_request(id = "r12345")
```

---

deepblue\_chromosomes     *chromosomes*

---

**Description**

List the chromosomes of a given Genome.

**Usage**

```
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)

**See Also**

Other Inserting and listing genomes: [deepblue\\_list\\_genomes](#), [deepblue\\_list\\_similar\\_genomes](#)

**Examples**

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

---

```
deepblue_clear_cache Clear cache
```

---

**Description**

Clear cache

**Usage**

```
deepblue_clear_cache()
```

**Value**

TRUE if successful

**Examples**

```
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**

```
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")
```

---

deepblue\_commands      *commands*

---

**Description**

List all available DeepBlue commands.

**Usage**

```
deepblue_commands()
```

**Value**

commands - A struct (command descriptions)

**See Also**

Other Checking DeepBlue status: [deepblue\\_echo](#)

**Examples**

```
deepblue_commands()
```

---

```
deepblue_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**

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

**Arguments**

genes	- A string or a vector of string (Name(s) or ENSEMBL ID (ENSGXXXXXXXXXXXXX.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: [deepblue\\_list\\_gene\\_models](#), [deepblue\\_list\\_genes](#), [deepblue\\_select\\_genes](#)

**Examples**

```
gene_names = c('CCR1', 'CD164', 'CD1D', 'CD2', 'CD34', 'CD3G', 'CD44')
deepblue_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\\_coverage](#), [deepblue\\_distinct\\_column\\_values](#), [deepblue\\_extend](#), [deepblue\\_filter\\_regions](#), [deepblue\\_flank](#), [deepblue\\_get\\_experiments\\_by\\_query](#), [deepblue\\_get\\_regions](#), [deepblue\\_input\\_regions](#), [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\\_re](#), [deepblue\\_tiling\\_regions](#)

**Examples**

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

---

```
deepblue_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"))
```

**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\\_count\\_regions](#), [deepblue\\_distinct\\_column\\_values](#), [deepblue\\_extend](#), [deepblue\\_filter\\_regions](#), [deepblue\\_flank](#), [deepblue\\_get\\_experiments\\_by\\_query](#), [deepblue\\_get\\_regions](#), [deepblue\\_input\\_regions](#), [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\\_re](#), [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	<i>diff</i>
---------------	-------------

---

**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")
```

---

deepblue_distinct_column_values	<i>distinct_column_values</i>
---------------------------------	-------------------------------

---

**Description**

Obtain the distinct values of the field.

**Usage**

```
deepblue_distinct_column_values(query_id = NULL, field = NULL,
  user_key = deepblue_options("user_key"))
```

**Arguments**

query_id	- A string (Query ID)
field	- A string (field that is filtered by)
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\\_count\\_regions](#), [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**

```
css_experiment <- deepblue_select_experiments ( "wgEncodeBroadHmmK562HMM")
distinct_names_request <- deepblue_distinct_column_values (css_experiment, "NAME")
```

---

```
deepblue_download_request_data
      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**

```
deepblue_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

```
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)
```

---

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

```
## 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_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: [deepblue\\_commands](#)

**Examples**

```
deepblue_echo(user_key = "anonymous_key")
```

---

```
deepblue_enrich_regions_go_terms  
    enrich_regions_go_terms
```

---

**Description**

Enrich the regions based on Gene Ontology terms.

**Usage**

```
deepblue_enrich_regions_go_terms(query_id = NULL, gene_model = NULL,  
    user_key = deepblue_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()`.  
The result is a list containing the GO terms that overlap with the query\_id regions.)

**See Also**

Other Enrich the genome regions: [deepblue\\_enrich\\_region\\_overlap](#)

**Examples**

```
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")  
  
deepblue_enrich_regions_go_terms(query_id = filtered_id,  
    gene_model = "gencode v23")
```



---

```
deepblue_enrich_region_overlap
      enrich_region_overlap
```

---

## Description

Enrich the regions based on regions overlap analysis.

## Usage

```
deepblue_enrich_region_overlap(query_id = NULL, background_query_id = NULL,
  datasets = NULL, genome = NULL, user_key = deepblue_options("user_key"))
```

## Arguments

query\_id - A string (Query ID)  
background\_query\_id - A string (query\_id containing the regions that will be used as the background data.)  
datasets - A struct (a map where each key is an identifier and the value is a list containing experiment names or query\_ids (you can use both together).)  
genome - A string (the target genome)  
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()).  
The result is a list containing the datasets that overlap with the query\_id regions.)

## See Also

Other Enrich the genome regions: [deepblue\\_enrich\\_regions\\_go\\_terms](#)

## Examples

```
query_id = deepblue_select_experiments(
  experiment_name="S00VEQA1.hypo_meth.bs_call.GRCh38.20150707.bed")

filtered_query_id = deepblue_filter_regions(
  query_id = query_id,
  field = "AVG_METHYL_LEVEL",
  operation = "<",
  value = "0.0025",
  type="number")

rg_10kb_tiling = deepblue_tiling_regions(
  size = 1000,
  genome = "hg19")

# We could have included more Epigenetic Marks here
epigenetic_marks <- c("h3k27ac", "H3K27me3", "H3K4me3")
```

```

histones_datasets = c()
for (i in 1:length(epigenetic_marks)) {
  experiments_list <- deepblue_list_experiments(
    epigenetic_mark=epigenetic_marks[[i]],
    type="peaks",
    genome="grch38",
    project="BLUEPRINT Epigenome");

  experiment_names = deepblue_extract_names(experiments_list)
  histones_datasets[[epigenetic_marks[[i]]]] = experiment_names
}

deepblue_enrich_region_overlap(
  query_id=filtered_query_id,
  background_query=rg_10kb_tilling,
  datasets=histones_datasets,
  genome="grch38")

```

---

deepblue\_export\_bed     *Export GenomicRanges result as BED file*

---

## Description

Export GenomicRanges result as BED file

## Usage

```

deepblue_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)

## 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(annotation_name="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_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_bed(regions, target.directory = temp_dir,
                   file.name = "GC_T14_10.CpG_islands")

```

---

deepblue\_export\_meta\_data

*Export meta data as tab delimited file*

---

### Description

Export meta data as tab delimited file

### Usage

```

deepblue_export_meta_data(ids, target.directory = "./", file.name,
                          user_key = deepblue_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

```

deepblue_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.bs_call.GRCh38.20160531.wig"),
  chromosome="chr1", start=0, end=50000000)
cpg_islands = deepblue_select_annotations(annotation_name="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_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")
  
```

---

deepblue_extend	<i>extend</i>
-----------------	---------------

---

**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"))
  
```

**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\\_count\\_regions](#), [deepblue\\_coverage](#), [deepblue\\_distinct\\_column\\_values](#), [deepblue\\_filter\\_regions](#), [deepblue\\_flank](#), [deepblue\\_get\\_experiments\\_by\\_query](#), [deepblue\\_get\\_regions](#), [deepblue\\_input\\_regions](#), [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\\_re](#), [deepblue\\_tiling\\_regions](#)

**Examples**

```
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    *extract\_ids*

---

**Description**

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

**Usage**

```
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**

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

deepblue\_extract\_names

*extract\_names*

---

### Description

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

### Usage

```
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

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

---

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

```
deepblue_faceting_experiments(genome = NULL, type = NULL,  
  epigenetic_mark = NULL, biosource = NULL, sample = NULL,  
  technique = NULL, project = NULL,  
  user_key = deepblue_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: [deepblue\\_collection\\_experiments\\_count](#), [deepblue\\_list\\_experiments](#), [deepblue\\_list\\_recent\\_experiments](#), [deepblue\\_list\\_similar\\_experiments](#), [deepblue\\_preview\\_experiment](#)

**Examples**

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

---

```
deepblue_filter_regions
  filter_regions
```

---

**Description**

Filter the genomic regions by their content.

**Usage**

```
deepblue_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\\_count\\_regions](#), [deepblue\\_coverage](#), [deepblue\\_distinct\\_column\\_values](#), [deepblue\\_extend](#), [deepblue\\_flank](#), [deepblue\\_get\\_experiments\\_by\\_query](#), [deepblue\\_get\\_regions](#), [deepblue\\_input\\_regions](#), [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\\_re](#), [deepblue\\_tiling\\_regions](#)

**Examples**

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

---

deepblue\_find\_motif    *find\_motif*

---

**Description**

Find genomic regions based on a given motif that appears in the genomic sequence.

**Usage**

```
deepblue_find_motif(motif = NULL, genome = NULL, chromosomes = NULL,
  start = NULL, end = NULL, overlap = NULL,
  user_key = deepblue_options("user_key"))
```

**Arguments**

motif	- A string (motif (PERL regular expression))
genome	- A string (the target genome)
chromosomes	- A string or a vector of string (chromosome name(s))
start	- A int (minimum start region)
end	- A int (maximum end region)
overlap	- A boolean (if the matching should do overlap search)
user_key	- A string (users token key)

**Value**

id - A string (id of the annotation that contains the positions of the given motif)



**See Also**

Other Inserting and listing annotations: [deepblue\\_list\\_annotations](#)

**Examples**

```
deepblue_find_motif(motif = "C[GT]+C", chromosomes=c("chr11", "chr12"),
  genome = "hg19", overlap = FALSE)
```

---

deepblue_flank	<i>flank</i>
----------------	--------------

---

**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**

```
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\\_count\\_regions](#), [deepblue\\_coverage](#), [deepblue\\_distinct\\_column\\_values](#), [deepblue\\_extend](#), [deepblue\\_filter\\_regions](#), [deepblue\\_get\\_experiments\\_by\\_query](#), [deepblue\\_get\\_regions](#), [deepblue\\_input\\_regions](#), [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\\_re](#), [deepblue\\_tiling\\_regions](#)

**Examples**

```
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_format_object_size`*Format byte size as human readable units*

---

**Description**

Format byte size as human readable units

**Usage**

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

**Arguments**

<code>x</code>	size in bytes
<code>units</code>	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**

```
deepblue_get_biosource_children(biosource = NULL,  
  user_key = deepblue_options("user_key"))
```

**Arguments**

<code>biosource</code>	- A string (biosource name)
<code>user_key</code>	- 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\\_synonyms](#)

**Examples**

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

---

```
deepblue_get_biosource_parents  
  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**

```
deepblue_get_biosource_parents(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 (parents biosources)

**See Also**

Other Set the relationship between different biosources: [deepblue\\_get\\_biosource\\_children](#), [deepblue\\_get\\_biosource\\_related](#), [deepblue\\_get\\_biosource\\_synonyms](#)

**Examples**

```
deepblue_get_biosource_parents(biosource = "Blood")
```

deepblue\_get\_biosource\_related  
*get\_biosource\_related*

---

### 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

```
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

```
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

```
deepblue_get_biosource_synonyms(biosource = NULL,  
    user_key = deepblue_options("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**

```
deepblue_get_biosource_synonyms(biosource = "prostate gland")
```

---

deepblue_get_db	<i>Sets up the DeepBlueR cache and returns a filehash db object</i>
-----------------	---

---

**Description**

Sets up the DeepBlueR cache and returns a filehash db object

**Usage**

```
deepblue_get_db()
```

**Value**

A filehash package database

---

deepblue_get_experiments_by_query	<i>get_experiments_by_query</i>
-----------------------------------	---------------------------------

---

**Description**

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

**Usage**

```
deepblue_get_experiments_by_query(query_id = NULL,  
  user_key = deepblue_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\\_count\\_regions](#), [deepblue\\_coverage](#), [deepblue\\_distinct\\_column\\_values](#), [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\\_re](#), [deepblue\\_tiling\\_regions](#)

**Examples**

```
deepblue_get_experiments_by_query(query_id = "q12345")
```

---

deepblue\_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 download using the URL: [http://deepblue.mpi-inf.mpg.de/download?r\\_id=<request\\_id>&key=<user\\_key>](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_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\\_count\\_regions](#), [deepblue\\_coverage](#), [deepblue\\_distinct\\_column\\_values](#), [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\\_re](#), [deepblue\\_tiling\\_regions](#)

## Examples

```
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

```
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

```
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

```
deepblue_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

```
deepblue_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

```
deepblue_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\\_count\\_regions](#), [deepblue\\_coverage](#), [deepblue\\_distinct\\_column\\_values](#), [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\\_re](#), [deepblue\\_tiling\\_regions](#)

**Examples**

```
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)
```

---

deepblue\_intersection *intersection*

---

**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**

```
deepblue_intersection(query_data_id = NULL, query_filter_id = 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.)  
 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\\_count\\_regions](#), [deepblue\\_coverage](#), [deepblue\\_distinct\\_column\\_values](#), [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\\_re](#), [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\_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**

```
deepblue_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)

**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**

```
deepblue_is_biosource(biosource = "blood")
```

---

deepblue_liftover	<i>Lift over region results between Genome Assemblies used in DeepBlue</i>
-------------------	--

---

**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**

```
deepblue_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 = 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)
deepblue_liftover(request_data, source = "hg38", target = "hg19")
```

```
deepblue_list_annotatons  
    list_annotatons
```

---

**Description**

List all annotations of genomic regions currently available in DeepBlue.

**Usage**

```
deepblue_list_annotatons(genome = NULL,  
    user_key = deepblue_options("user_key"))
```

**Arguments**

genome - A string or a vector of string (the target genome)  
user\_key - A string (users token key)

**Value**

annotatons - A array (annotatons names and IDs)

**See Also**

Other Inserting and listing annotations: [deepblue\\_find\\_motif](#)

**Examples**

```
deepblue_list_annotatons(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**

```
deepblue_list_biosources(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**

biosources - A array (biosources names and IDS)

**See Also**

Other Inserting and listing biosources: [deepblue\\_list\\_similar\\_biosources](#)

**Examples**

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

---

deepblue\_list\_cached\_requests  
*List cached requests*

---

**Description**

List cached requests

**Usage**

```
deepblue_list_cached_requests()
```

**Value**

list of request ids that are cached

**Examples**

```
deepblue_list_cached_requests()
```

---

deepblue\_list\_column\_types  
*list\_column\_types*

---

**Description**

Lists the ColumnTypes included in DeepBlue.

**Usage**

```
deepblue_list_column_types(user_key = deepblue_options("user_key"))
```

**Arguments**

user\_key - A string (users token key)

**Value**

column\_types - A array (column types names and IDS)

**Examples**

```
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**

```
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()
```

---

```
deepblue_list_experiments  
    list_experiments
```

---

**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_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**

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**

```
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**

```
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**

```
deepblue_list_expressions(expression_type='gene')
```

---

```
deepblue_list_genes    list_genes
```

---

**Description**

List the Genes currently available in DeepBlue.

**Usage**

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

**Arguments**

genes	- A string or a vector of string (Name(s) or ENSEMBL ID (ENSGXXXXXXXXXXXXX.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**

```
deepblue_list_genes(
  chromosome="chr20",
  start=10000000,
  end=21696620,
  gene_model='Gencode 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**

```
deepblue_list_gene_models(user_key = deepblue_options("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\\_count\\_gene\\_ontology\\_terms](#), [deepblue\\_list\\_genes](#), [deepblue\\_select\\_genes](#)

**Examples**

```
deepblue_list_gene_models()
```

---

deepblue\_list\_genomes *list\_genomes*

---

**Description**

List Genomes assemblies that are registered in DeepBlue.

**Usage**

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

**Arguments**

user\_key - A string (users token key)

**Value**

genomes - A array (genome names)

**See Also**

Other Inserting and listing genomes: [deepblue\\_chromosomes](#), [deepblue\\_list\\_similar\\_genomes](#)

**Examples**

```
deepblue_list_genomes()
```

---

deepblue\_list\_in\_use    *list\_in\_use*

---

**Description**

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

**Usage**

```
deepblue_list_in_use(controlled_vocabulary = NULL,  
                    user_key = deepblue_options("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](#), [deepblue\\_name\\_to\\_id](#), [deepblue\\_search](#)

**Examples**

```
deepblue_list_in_use(controlled_vocabulary = "biosources")
```

---

deepblue\_list\_projects  
*list\_projects*

---

**Description**

List Projects included in DeepBlue.

**Usage**

```
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**

```
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**

```
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(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**

```
deepblue_list_recent_experiments(days = 2, genome = "hg19")
```

---

```
deepblue_list_requests
      list_requests
```

---

**Description**

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

**Usage**

```
deepblue_list_requests(request_state = NULL,
  user_key = deepblue_options("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**

```
deepblue_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**

```
deepblue_list_samples(biosource = NULL, extra_metadata = NULL,  
  user_key = deepblue_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**

```
deepblue_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**

```
deepblue_list_similar_biosources(name = NULL,  
  user_key = deepblue_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**

```
deepblue_list_similar_biosources(name = "blood")
```

---

```
deepblue_list_similar_epigenetic_marks  
    list_similar_epigenetic_marks
```

---

**Description**

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

**Usage**

```
deepblue_list_similar_epigenetic_marks(name = NULL,  
    user_key = deepblue_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: [deepblue\\_list\\_epigenetic\\_marks](#)

**Examples**

```
deepblue_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

```
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")
```

---

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

```
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")
```

---

deepblue\_list\_similar\_projects  
*list\_similar\_projects*

---

**Description**

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**

```
deepblue_list_similar_techniques(name = NULL,  
    user_key = deepblue_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**

```
deepblue_list_similar_techniques(name = "chip seq")
```

---

deepblue\_list\_techniques  
*list\_techniques*

---

**Description**

List the Techniques included in DeepBlue.

**Usage**

```
deepblue_list_techniques(user_key = deepblue_options("user_key"))
```

**Arguments**

user\_key - A string (users token key)

**Value**

techniques - A array (techniques)

**See Also**

Other Inserting and listing techniques: [deepblue\\_list\\_similar\\_techniques](#)

**Examples**

```
deepblue_list_techniques()
```

---

```
deepblue_merge_queries  
    merge_queries
```

---

**Description**

Merge regions from two queries in a new query.

**Usage**

```
deepblue_merge_queries(query_a_id = NULL, query_b_id = NULL,  
    user_key = deepblue_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: [deepblue\\_aggregate](#), [deepblue\\_binning](#), [deepblue\\_count\\_regions](#), [deepblue\\_coverage](#), [deepblue\\_distinct\\_column\\_values](#), [deepblue\\_extend](#), [deepblue\\_filter\\_regions](#), [deepblue\\_flank](#), [deepblue\\_get\\_experiments\\_by\\_query](#), [deepblue\\_get\\_regions](#), [deepblue\\_input\\_regions](#), [deepblue\\_intersection](#), [deepblue\\_overlap](#), [deepblue\\_query\\_cache](#), [deepblue\\_query\\_experiment\\_type](#), [deepblue\\_score\\_matrix](#), [deepblue\\_select\\_annotations](#), [deepblue\\_select\\_experiments](#), [deepblue\\_select\\_re](#), [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_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"))
```

---

 deepblue\_name\_to\_id    *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**

```
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_options	<i>options</i>
------------------	----------------

---

**Description**

options manager from the settings package

**Usage**

```
deepblue_options(..., __defaults = FALSE, __reset = FALSE)
```

**Arguments**

...	list of new options
__defaults	disallowed option
__reset	disallowed option

**Value**

default options

---

deepblue_overlap	<i>overlap</i>
------------------	----------------

---

**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**

```
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\\_count\\_regions](#), [deepblue\\_coverage](#), [deepblue\\_distinct\\_column\\_values](#), [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\\_re](#), [deepblue\\_tiling\\_regions](#)

**Examples**

```
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="%")
```

---

```
deepblue_preview_experiment
    preview_experiment
```

---

**Description**

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

**Usage**

```
deepblue_preview_experiment(experiment_name = NULL,
  user_key = deepblue_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**

```
deepblue_preview_experiment('S00JJRH1.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**

```
deepblue_query_cache(query_id = NULL, cache = NULL,
  user_key = deepblue_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\\_count\\_regions](#), [deepblue\\_coverage](#), [deepblue\\_distinct\\_column\\_values](#), [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\\_re](#), [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.)

**See Also**

Other Operating on the data regions: [deepblue\\_aggregate](#), [deepblue\\_binning](#), [deepblue\\_count\\_regions](#), [deepblue\\_coverage](#), [deepblue\\_distinct\\_column\\_values](#), [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\\_score\\_matrix](#), [deepblue\\_select\\_annotations](#), [deepblue\\_select\\_experiments](#), [deepblue\\_select\\_re](#), [deepblue\\_tiling\\_regions](#)

**Examples**

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

---

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 *score\_matrix*

---

## Description

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

## Usage

```
deepblue_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\\_count\\_regions](#), [deepblue\\_coverage](#), [deepblue\\_distinct\\_column\\_values](#), [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

```
tiling_regions = deepblue_tiling_regions(  
  size=100000, genome="mm10", chromosome="chr1")  
deepblue_score_matrix(  
  experiments_columns =  
    list(ENCF721EKA="VALUE", ENCF781VH="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

```
deepblue_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

```
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

```
deepblue_select_annotations(annotation_name = NULL, genome = NULL,
  chromosome = NULL, start = NULL, end = NULL,
  user_key = deepblue_options("user_key"))
```

**Arguments**

annotation_name	- A string or a vector of string (name(s) of selected annotation(s))
genome	- A 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\\_count\\_regions](#), [deepblue\\_coverage](#), [deepblue\\_distinct\\_column\\_values](#), [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**

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

---

deepblue\_select\_column

*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_options("user_key"))
```

**Arguments**

experiments	- A data frame with experiments obtained from <code>deepblue_list_experiments</code>
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**

```
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")
```

---

```
deepblue_select_experiments
      select_experiments
```

---

**Description**

Selects regions from Experiments by the experiments names.

**Usage**

```
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\\_count\\_regions](#), [deepblue\\_coverage](#), [deepblue\\_distinct\\_column\\_values](#), [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**

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

---

```
deepblue_select_expressions
      select_expressions
```

---

**Description**

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

**Usage**

```
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**

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

```
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_options("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\\_binning](#), [deepblue\\_count\\_regions](#), [deepblue\\_coverage](#), [deepblue\\_distinct\\_column\\_values](#), [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

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

---

```
deepblue_tiling_regions
    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

```
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\\_count\\_regions](#), [deepblue\\_coverage](#), [deepblue\\_distinct\\_column\\_values](#), [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

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



---

deepblue\_wait\_request *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_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

- deepblue\_aggregate, [4](#), [6](#), [11](#), [12](#), [14](#), [21](#), [24](#), [25](#), [30](#), [33](#), [34](#), [50](#), [53](#), [55–57](#), [59](#), [61](#), [63](#), [64](#)
- deepblue\_batch\_export\_results, [5](#)
- deepblue\_binning, [4](#), [5](#), [11](#), [12](#), [14](#), [21](#), [24](#), [25](#), [30](#), [33](#), [34](#), [50](#), [53](#), [55–57](#), [59](#), [61](#), [63](#), [64](#)
- deepblue\_cache\_status, [6](#)
- deepblue\_cancel\_request, [7](#), [32](#), [35](#), [42](#), [52](#), [58](#)
- deepblue\_chromosomes, [7](#), [42](#), [48](#)
- deepblue\_clear\_cache, [8](#)
- deepblue\_collection\_experiments\_count, [8](#), [23](#), [39](#), [44](#), [47](#), [54](#)
- deepblue\_commands, [9](#), [16](#)
- deepblue\_count\_gene\_ontology\_terms, [10](#), [40](#), [41](#), [62](#)
- deepblue\_count\_regions, [4](#), [6](#), [11](#), [12](#), [14](#), [21](#), [24](#), [25](#), [30](#), [33](#), [34](#), [50](#), [53](#), [55–57](#), [59](#), [61](#), [63](#), [64](#)
- deepblue\_coverage, [4](#), [6](#), [11](#), [11](#), [14](#), [21](#), [24](#), [25](#), [30](#), [33](#), [34](#), [50](#), [53](#), [55–57](#), [59](#), [61](#), [63](#), [64](#)
- deepblue\_delete\_request\_from\_cache, [12](#)
- deepblue\_diff, [13](#), [60](#)
- deepblue\_distinct\_column\_values, [4](#), [6](#), [11](#), [12](#), [13](#), [21](#), [24](#), [25](#), [30](#), [33](#), [34](#), [50](#), [53](#), [55–57](#), [59](#), [61](#), [63](#), [64](#)
- deepblue\_download\_request\_data, [14](#)
- deepblue\_download\_request\_data, DeepBlueCommand-method, [15](#)
- deepblue\_echo, [9](#), [15](#)
- deepblue\_enrich\_region\_overlap, [16](#), [17](#)
- deepblue\_enrich\_regions\_go\_terms, [16](#), [17](#)
- deepblue\_export\_bed, [18](#)
- deepblue\_export\_meta\_data, [19](#)
- deepblue\_export\_tab, [19](#)
- deepblue\_extend, [4](#), [6](#), [11](#), [12](#), [14](#), [20](#), [24](#), [25](#), [30](#), [33](#), [34](#), [50](#), [53](#), [55–57](#), [59](#), [61](#), [63](#), [64](#)
- deepblue\_extract\_ids, [21](#), [22](#)
- deepblue\_extract\_names, [21](#), [22](#)
- deepblue\_faceting\_experiments, [9](#), [22](#), [39](#), [44](#), [47](#), [54](#)
- deepblue\_filter\_regions, [4](#), [6](#), [11](#), [12](#), [14](#), [21](#), [23](#), [25](#), [30](#), [33](#), [34](#), [50](#), [53](#), [55–57](#), [59](#), [61](#), [63](#), [64](#)
- deepblue\_find\_motif, [24](#), [36](#)
- deepblue\_flank, [4](#), [6](#), [11](#), [12](#), [14](#), [21](#), [24](#), [25](#), [30](#), [33](#), [34](#), [50](#), [53](#), [55–57](#), [59](#), [61](#), [63](#), [64](#)
- deepblue\_format\_object\_size, [26](#)
- deepblue\_get\_biosource\_children, [26](#), [27–29](#)
- deepblue\_get\_biosource\_parents, [27](#), [27](#), [28](#), [29](#)
- deepblue\_get\_biosource\_related, [27](#), [28](#), [29](#)
- deepblue\_get\_biosource\_synonyms, [27](#), [28](#), [28](#)
- deepblue\_get\_db, [29](#)
- deepblue\_get\_experiments\_by\_query, [4](#), [6](#), [11](#), [12](#), [14](#), [21](#), [24](#), [25](#), [29](#), [30](#), [33](#), [34](#), [50](#), [53](#), [55–57](#), [59](#), [61](#), [63](#), [64](#)
- deepblue\_get\_regions, [4](#), [6](#), [11](#), [12](#), [14](#), [21](#), [24](#), [25](#), [30](#), [30](#), [33](#), [34](#), [50](#), [53](#), [55–57](#), [59](#), [61](#), [63](#), [64](#)
- deepblue\_get\_request\_data, [31](#), [44](#)
- deepblue\_info, [7](#), [32](#), [35](#), [42](#), [52](#), [58](#)
- deepblue\_input\_regions, [4](#), [6](#), [11](#), [12](#), [14](#), [21](#), [24](#), [25](#), [30](#), [32](#), [34](#), [50](#), [53](#), [55–57](#), [59](#), [61](#), [63](#), [64](#)
- deepblue\_intersection, [4](#), [6](#), [11](#), [12](#), [14](#), [21](#), [24](#), [25](#), [30](#), [33](#), [33](#), [50](#), [53](#), [55–57](#), [59](#), [61](#), [63](#), [64](#)
- deepblue\_is\_biosource, [7](#), [32](#), [34](#), [42](#), [52](#), [58](#)
- deepblue\_liftover, [35](#)
- deepblue\_list\_annotations, [25](#), [36](#)
- deepblue\_list\_biosources, [36](#), [46](#)
- deepblue\_list\_cached\_requests, [37](#)
- deepblue\_list\_column\_types, [37](#)
- deepblue\_list\_epigenetic\_marks, [38](#), [46](#)
- deepblue\_list\_experiments, [9](#), [23](#), [38](#), [44](#), [47](#), [54](#), [60](#)
- deepblue\_list\_expressions, [39](#), [61](#)

- deepblue\_list\_gene\_models, [10](#), [40](#), [41](#), [62](#)
- deepblue\_list\_genes, [10](#), [40](#), [41](#), [62](#)
- deepblue\_list\_genomes, [8](#), [41](#), [48](#)
- deepblue\_list\_in\_use, [7](#), [32](#), [35](#), [42](#), [52](#), [58](#)
- deepblue\_list\_projects, [43](#), [48](#)
- deepblue\_list\_recent\_experiments, [9](#), [23](#), [39](#), [43](#), [47](#), [54](#)
- deepblue\_list\_requests, [31](#), [44](#)
- deepblue\_list\_samples, [45](#)
- deepblue\_list\_similar\_biosources, [37](#), [45](#)
- deepblue\_list\_similar\_epigenetic\_marks, [38](#), [46](#)
- deepblue\_list\_similar\_experiments, [9](#), [23](#), [39](#), [44](#), [47](#), [54](#)
- deepblue\_list\_similar\_genomes, [8](#), [42](#), [47](#)
- deepblue\_list\_similar\_projects, [43](#), [48](#)
- deepblue\_list\_similar\_techniques, [49](#), [50](#)
- deepblue\_list\_techniques, [49](#), [49](#)
- deepblue\_merge\_queries, [4](#), [6](#), [11](#), [12](#), [14](#), [21](#), [24](#), [25](#), [30](#), [33](#), [34](#), [50](#), [53](#), [55–57](#), [59](#), [61](#), [63](#), [64](#)
- deepblue\_meta\_data\_to\_table, [51](#)
- deepblue\_name\_to\_id, [7](#), [32](#), [35](#), [42](#), [51](#), [58](#)
- deepblue\_options, [52](#)
- deepblue\_overlap, [4](#), [6](#), [11](#), [12](#), [14](#), [21](#), [24](#), [25](#), [30](#), [33](#), [34](#), [50](#), [52](#), [55–57](#), [59](#), [61](#), [63](#), [64](#)
- deepblue\_preview\_experiment, [9](#), [23](#), [39](#), [44](#), [47](#), [53](#)
- deepblue\_query\_cache, [4](#), [6](#), [11](#), [12](#), [14](#), [21](#), [24](#), [25](#), [30](#), [33](#), [34](#), [50](#), [53](#), [54](#), [56](#), [57](#), [59](#), [61](#), [63](#), [64](#)
- deepblue\_query\_experiment\_type, [4](#), [6](#), [11](#), [12](#), [14](#), [21](#), [24](#), [25](#), [30](#), [33](#), [34](#), [50](#), [53](#), [55](#), [55](#), [57](#), [59](#), [61](#), [63](#), [64](#)
- deepblue\_reset\_options, [56](#)
- deepblue\_score\_matrix, [4](#), [6](#), [11](#), [12](#), [14](#), [21](#), [24](#), [25](#), [30](#), [33](#), [34](#), [50](#), [53](#), [55](#), [56](#), [57](#), [59–61](#), [63](#), [64](#)
- deepblue\_search, [7](#), [32](#), [35](#), [42](#), [52](#), [58](#)
- deepblue\_select\_annotations, [4](#), [6](#), [11](#), [12](#), [14](#), [21](#), [24](#), [25](#), [30](#), [33](#), [34](#), [50](#), [53](#), [55–57](#), [58](#), [61](#), [63](#), [64](#)
- deepblue\_select\_column, [13](#), [59](#)
- deepblue\_select\_experiments, [4](#), [6](#), [11](#), [12](#), [14](#), [21](#), [24](#), [25](#), [30](#), [33](#), [34](#), [50](#), [53](#), [55–57](#), [59](#), [60](#), [63](#), [64](#)
- deepblue\_select\_expressions, [40](#), [61](#)
- deepblue\_select\_genes, [10](#), [40](#), [41](#), [62](#)
- deepblue\_select\_regions, [4](#), [6](#), [11](#), [12](#), [14](#), [21](#), [24](#), [25](#), [30](#), [33](#), [34](#), [50](#), [53](#), [55–57](#), [59](#), [61](#), [63](#), [64](#)
- deepblue\_tiling\_regions, [4](#), [6](#), [11](#), [12](#), [14](#), [21](#), [24](#), [25](#), [30](#), [33](#), [34](#), [50](#), [53](#), [55–57](#), [59](#), [61](#), [63](#), [64](#)
- deepblue\_wait\_request, [65](#)
- DeepBlueCommand
  - (DeepBlueCommand-class), [3](#)
- DeepBlueCommand-class, [3](#)