D-Genies Documentation

Release 1.1

Floréal Cabanettes

Mar 30, 2022

CONTENTS

Dot plots are widely used to quickly compare sequence sets. They provide a synthetic similarity overview, highlighting repetitions, breaks and inversions. Different tools have been developed to easily generated genomic alignment dot plots, but they are often limited in the input sequence size. D-GENIES is a standalone and web application performing large genome alignments using minimap2 software package and generating interactive dot plots. It enables users to sort query sequences along the reference, zoom in the plot and download several image, alignment or sequence files. D-GENIES is an easy-to-install, open-source software package (GPL) developed in Python and JavaScript. The source code is available at https://github.com/genotoul-bioinfo/dgenies and it can be tested at http://dgenies.toulouse.inra.fr/.

CHAPTER

ONE

HOW TO CITE?

Cabanettes F, Klopp C. (2018) D-GENIES: dot plot large genomes in an interactive, efficient and simple way. PeerJ 6:e4958 https://doi.org/10.7717/peerj.4958

CHAPTER

TWO

HOW TO INSTALL?

See the doc here

CHAPTER

THREE

HOW TO USE?

CHAPTER

FOUR

CODE API

4.1 Index

- genindex

4.2 Python server part

4.2.1 Python packages & modules

Subpackages

dgenies.lib package

Submodules

dgenies.lib.crons module

class dgenies.lib.crons.**Crons**(*base_dir*, *debug*)

Bases: object

Manage crontab jobs (webserver mode)

Parameters

- **base_dir** (*str*) – software base directory path
- **debug** (*bool*) – True to enable debug mode

clear(*kill_scheduler=True*, *remove_pid_file=True*)

Clear all crons

Parameters

- **kill_scheduler** (*bool*) – if True, kill local scheduler currently running
- **remove_pid_file** (*bool*) – if True, remove pid file if local scheduler was killed successfully

init_clean_cron()

Initialize clean cron: will clear old jobs. Clean cron is launched at 1h00am each day

init_launch_local_cron()

Try to launch local scheduler (if not already launched)

start_all()

Start all crons

dgenies.lib.decorators module

class dgenies.lib.decorators.**Singleton**(*klass*)

Bases: object

Define a singleton (design pattern)

dgenies.lib.drmaasession module

dgenies.lib.fasta module

class dgenies.lib.fasta.**Fasta**(*name*, *path*, *type_f*, *example=False*)

Bases: object

Defines a fasta file: name of the sample, path to the fasta file, type of file (URL or local file), …

Parameters

- **name** (*str*) – sample name
- **path** (*str*) – fasta file path
- **type_f** (*str*) – type of file (local file or URL)
- **example** (*bool*) – is an example job

get_name()

Get sample name

Returns sample name

Return type str

get_path()

Get path of the fasta file

Returns fasta path

Return type str

get_type()

Get type: URL or local file

Returns type

Return type str

is_example()

Return if current sample is an example data

Returns current sample is an example data

Return type bool

set_name(*name*)

Set sample name

Parameters **name** (*str*) – new sample name

set_path(*path*)

Set path to the fasta file

Parameters **path** (*str*) – new path

dgenies.lib.functions module

class dgenies.lib.functions.**Functions**

Bases: object

General functions

static allowed_file(*filename*, *file_formats=('fasta',)*)

Check whether a file has a valid format

Parameters

- **filename** – file path
- **file_formats** – accepted file formats

Returns True if valid format, else False

static compress(*filename*)

Compress a file with gzip

Parameters **filename** (*str*) – file to compress

Returns path of the compressed file

Return type str

static compress_and_send_mail(*job_name*, *fasta_file*, *index_file*, *lock_file*, *mailer*)

Compress fasta file and the send mail with its link to the client

Parameters

- **job_name** (*str*) – job id
- **fasta_file** (*str*) – fasta file path
- **index_file** (*str*) – index file path
- **lock_file** (*str*) – lock file path
- **mailer** (Mailer) – mailer object (to send mail)

config = <dgenies.config_reader.AppConfigReader object>

static get_fasta_file(*res_dir*, *type_f*, *is_sorted*)

Get fasta file path

Parameters

- **res_dir** (*str*) – job results directory
- **type_f** (*str*) – type of file (query or target)
- **is_sorted** (*bool*) – is fasta sorted

Returns fasta file path

Return type str

`static get_gallery_items()`

Get list of items from the gallery

Returns

list of item of the gallery. Each item is a dict with 7 keys:

- *name* : name of the job
- *id_job* : id of the job
- *picture* : illustrating picture filename (located in gallery folder of the data folder)
- *query* : query specie name
- *target* : target specie name
- *mem_peak* : max memory used for the run (human readable)
- *time_elapsed* : time elapsed for the run (human readable)

Return type list of dict

`static get_list_all_jobs(mode='webserver')`

Get list of all jobs

Parameters **mode** (*str*) – webserver or standalone

Returns list of all jobs in standalone mode. Empty list in webserver mode

Return type list

`static get_mail_for_job(id_job)`

Retrieve associated mail for a job

Parameters **id_job** (*int*) – job id

Returns associated mail address

Return type str

`static get_readable_size(size, nb_after_coma=1, base='B')`

Get human readable size from a given size in bytes

Parameters

- **size** (*int*) – size in bytes
- **nb_after_coma** (*str*) – number of digits after coma
- **base** – base unit of size, must be either “B”, “KiB”, “MiB” or “GiB”

Returns size, human readable

Return type str

`static get_readable_time(seconds)`

Get human readable time

Parameters **seconds** (*int*) – time in seconds

Returns time, human readable

Return type str

static get_valid_uploaded_filename(*filename*, *folder*)

Check whether uploaded file already exists. If yes, rename it

Parameters

- **filename** (*str*) – uploaded file
- **folder** (*str*) – folder into save the file

Returns unique filename

Return type str

static is_in_gallery(*id_job*, *mode='webserver'*)

Check whether a job is in the gallery

Parameters

- **id_job** (*str*) – job id
- **mode** (*str*) – webserver or standalone

Returns True if job is in the gallery, else False

Return type bool

static query_fasta_file_exists(*res_dir*)

Check if a fasta file exists

Parameters **res_dir** (*str*) – job result directory

Returns True if file exists and is a regular file, else False

Return type bool

static random_string(*s_len*)

Generate a random string

Parameters **s_len** (*int*) – length of the string to generate

Returns the random string

Return type str

static read_index(*index_file*)

Load index of query or target

Parameters **index_file** (*str*) – index file path

Returns

- [0] index (size of each chromosome) {dict}
- [1] sample name {str}

Return type (dict, str)

static send_fasta_ready(*mailer*, *job_name*, *sample_name*, *compressed=False*, *path='fasta-query'*, *status='success'*, *ext='fasta'*)

Send link to fasta file when treatment ended

Parameters

- **mailer** (Mailer) – mailer object
- **job_name** (*str*) – job id

- **sample_name** (*str*) – sample name
- **compressed** (*bool*) – is a compressed fasta file
- **path** (*str*) – fasta path
- **status** (*str*) – treatment status
- **ext** (*str*) – file extension

static sort_fasta(*job_name*, *fasta_file*, *index_file*, *lock_file*, *compress=False*, *mailer=None*, *mode='webserver'*)

Sort fasta file according to the sorted index file

Parameters

- **job_name** (*str*) – job id
- **fasta_file** (*str*) – fasta file path
- **index_file** (*str*) – index file path
- **lock_file** (*str*) – lock file path
- **compress** (*bool*) – compress result fasta file
- **mailer** (Mailer) – mailer object (to send mail)
- **mode** (*str*) – webserver or standalone

static uncompress(*filename*)

Uncompress a gzipped file

Parameters **filename** (*str*) – gzipped file

Returns path of the uncompressed file

Return type str

dgenies.lib.job_manager module

class dgenies.lib.job_manager.**JobManager**(*id_job*, *email=None*, *query: Optional[*dgenies.lib.fasta.Fasta*] = None*, *target: Optional[*dgenies.lib.fasta.Fasta*] = None*, *mailer=None*, *tool='minimap2'*, *align: Optional[*dgenies.lib.fasta.Fasta*] = None*, *backup: Optional[*dgenies.lib.fasta.Fasta*] = None*, *options=None*)

Bases: object

Jobs management

Parameters

- **id_job** (*str*) – job id
- **email** (*str*) – email from user
- **query** (Fasta) – query fasta
- **target** (Fasta) – target fasta
- **mailer** (Mailer) – mailer object (to send mail throw flask app)
- **tool** (*str*) – tool to use for mapping (choice from tools config)
- **align** (Fasta) – alignment file (PAF, MAF, ...) as a fasta object

- **backup** (Fasta) – backup TAR file
- **options** (*list*) – list of str containing options for the chosen tool

check_file(*input_type*, *should_be_local*, *max_upload_size_readable*)

Check if file is correct: format, size, valid gzip

Parameters

- **input_type** – query or target
- **should_be_local** – True if job should be treated locally
- **max_upload_size_readable** – max upload size human readable

Returns (True if correct, True if error set [for fail], True if should be local)

check_job_status_sge()

Check status of a SGE job run

Returns True if the job jas successfully ended, else False

check_job_status_slurm()

Check status of a SLURM job run

Returns True if the job has successfully ended, else False

check_job_success()

Check if a job succeed

Returns status of a job: succeed, no-match or fail

Return type str

clear()

Remove job dir

delete()

Remove a job

Returns

- [0] Success of the deletion
- [1] Error message, if any (else empty string)

Return type (bool, str)

do_align()

Check if we have to make alignment

Returns True if the job is launched with an alignment file

download_files_with_pending(*files_to_download*, *should_be_local*, *max_upload_size_readable*)

Download files from URLs, with pending (according to the max number of concurrent downloads)

Parameters

- **files_to_download** (*list of list*) – files to download. For each item of the list, it's a list with 2 elements: first one is the Fasta object, second one the input type (query or target)
- **should_be_local** (*bool*) – True if the job should be run locally (according to input file sizes), else False

- **max_upload_size_readable** (*str*) – Human readable max upload size (to show on errors)

static find_error_in_log(*log_file*)

Find error in log (for cluster run)

Parameters **log_file** – log file of the job

Returns error (empty if no error)

Return type str

get_file_size(*filepath: str*)

Get file size

Parameters **filepath** (*str*) – file path

Returns file size (bytes)

Return type int

get_mail_content(*status*, *target_name*, *query_name=None*)

Build mail content for status mail

Parameters

- **status** (*str*) – job status
- **target_name** (*str*) – name of target
- **query_name** (*str*) – name of query

Returns mail content

Return type str

get_mail_content_html(*status*, *target_name*, *query_name=None*)

Build mail content as HTML

Parameters

- **status** (*str*) – job status
- **target_name** (*str*) – name of target
- **query_name** (*str*) – name of query

Returns mail content (html)

Return type str

get_mail_subject(*status*)

Build mail subject

Parameters **status** (*str*) – job status

Returns mail subject

Return type str

static get_pending_local_number()

Get number of of jobs running or waiting for a run

Returns number of jobs

Return type int

`get_query_split`()

Get query split fasta file

Returns split query fasta file

Return type str

`get_status_standalone`(*with_error=False*)

Get job status in standalone mode

Parameters **with_error** – get also the error

Returns status (and error, if with_error=True)

Return type str or tuple (if with_error=True)

`getting_files`()

Get files for the job

Returns

- [0] True if getting files succeed, False else
- [1] If error happenned, True if error already saved for the job, False else (error will be saved later)
- [2] True if no data must be downloaded (will be downloaded with pending if True)

Return type tuple

`static is_gz_file`(*filepath*)

Check if a file is gzipped

Parameters **filepath** (*str*) – file to check

Returns True if gzipped, else False

`is_query_filtered`()

Check if query has been filtered

Returns True if filtered, else False

`is_target_filtered`()

Check if target has been filtered

Returns True if filtered, else False

Returns

`launch`()

Launch a job in webserver mode (asynchronously in a new thread)

`launch_standalone`()

Launch a job in standalone mode (asynchronously in a new thread)

`launch_to_cluster`(*step*, *batch_system_type*, *command*, *args*, *log_out*, *log_err*)

Launch a program to the cluster

Parameters

- **step** (*str*) – step (prepare, start)
- **batch_system_type** (*str*) – slurm or sge
- **command** (*str*) – program to launch (without arguments)

- **args** (*list*) – arguments to use for the program
- **log_out** (*str*) – log file for stdout
- **log_err** (*str*) – log file for stderr

Returns True if succeed, else False

Return type bool

prepare_data()

Launch preparation of data

prepare_data_cluster(*batch_system_type*)

Launch of prepare data on a cluster

Parameters **batch_system_type** (*str*) – slurm or sge

Returns True if succeed, else False

Return type bool

prepare_data_in_thread()

Prepare data in a new thread

prepare_data_local()

Prepare data locally. On standalone mode, launch job after, if success. :return: True if job succeed, else False :rtype: bool

prepare_dotplot_cluster(*batch_system_type*)

Prepare data if alignment already done: just index the fasta (if index not given), then parse the alignment

Parameters **batch_system_type** (*str*) – type of cluster (slurm or sge)

prepare_dotplot_local()

Prepare data if alignment already done: just index the fasta (if index not given), then parse the alignment file and sort it.

run_job(*batch_system_type*)

Run of a job (mapping step)

Parameters **batch_system_type** (*str*) – type of cluster (slurm or sge)

run_job_in_thread(*batch_system_type='local'*)

Run a job asynchronously into a new thread

Parameters **batch_system_type** (*str*) – slurm or sge

search_error()

Search for an error in the log file (for local runs). If no error found, returns a generic error message

Returns error message to give to the user

Return type str

send_mail()

Send mail

send_mail_post()

Send mail using POST url (if there is no access to mailer)

set_inputs_from_res_dir()

Sets inputs (query, target, ...) from job dir

set_job_status(*status*, *error=''*)

Change status of a job

Parameters

- **status** (*str*) – new job status
- **error** (*str*) – error description (if any)

set_status_standalone(*status*, *error=''*)

Change job status in standalone mode

Parameters

- **status** (*str*) – new status
- **error** (*str*) – error description (if any)

start_job()

Start job: download, check and parse input files

status()

Get job status and error. In webserver mode, get also mem peak and time elapsed

Returns status and other information

Return type dict

unpack_backup()

Untar backup file

update_job_status(*status*, *id_process=None*)

Update job status

Parameters

- **status** – new status
- **id_process** – system process id

dgenies.lib.latest module

class dgenies.lib.latest.**Latest**

Bases: object

Search latest version

load()

Load latest version: use cached version (if any) and then sync with Github

update()

Get latest version from Github

update_async()

Update latest version asynchronously

dgenies.lib.mailer module

class dgenies.lib.mailer.**Mailer**(*app*)

Bases: object

Send mail throw flask app

Parameters **app** (*Flask*) – Flask app object

send_mail(*recipients*, *subject*, *message*, *message_html=None*)

Send mail

Parameters

- **recipients** (*list*) – list of recipients
- **subject** (*str*) – mail subject
- **message** (*str*) – message (text)
- **message_html** (*str*) – message (html)

dgenies.lib.paf module

class dgenies.lib.paf.**Paf**(*paf: str*, *idx_q: str*, *idx_t: str*, *auto_parse: bool = True*, *mailer=None*, *id_job=None*)

Bases: object

Functions applied to PAF files

Parameters

- **paf** (*str*) – PAF file path
- **idx_q** (*str*) – query index file path
- **idx_t** (*str*) – target index file path
- **auto_parse** (*bool*) – if True, parse PAF file at initialisation
- **mailer** (Mailer) – mailer object, to send mails
- **id_job** (*str*) – job id

build_list_no_assoc(*to*)

Build list of queries that match with None target, or the opposite

Parameters **to** – query or target

Returns content of the file

build_query_chr_as_reference()

Assemble query contigs like reference chromosomes

Returns path of the fasta file

build_query_on_target_association_file()

For each query, get the best matching chromosome and save it to a CSV file. Use the order of queries

Returns content of the file

build_summary_stats(*status_file*)

Get summary of identity

Returns table with percents by category

compute_gravity_contigs()

Compute gravity for each contig on each chromosome (how many big matches they have). Will be used to find which chromosome has the highest value for each contig

Returns

- **[0] gravity for each contig and each chromosome:** {contig1: {chr1: value, chr2: value, ...}, contig2: ...}
- **[1] For each block save lines inside:** [median_on_query, squared length, median_on_target, x1, x2, y1, y2, length] (x : on target, y: on query)

config = <dgenies.config_reader.AppConfigReader object>

get_d3js_data()

Build data for D3.js client

Returns

data for d3.js:

- y_len: length of query (Bp)
- x_len: length of target (Bp)
- min_idy: minimum of identity (float)
- max_idy: maximum of identity (float)
- lines: matches lines, by class of identity (dict)
- y_contigs: query contigs definitions (dict)
- y_order: query contigs order (list)
- x_contigs: target contigs definitions (dict)
- x_order: target contigs order (list)
- name_y: name of the query (str)
- name_x: name of the target (str)
- limit_idy: limit for each class of identities (list)

Return type dict

get_queries_on_target_association()

For each target, get the list of queries associated to it

Returns list of queries associated to each target

Return type dict

get_query_on_target_association(*with_coords=True*)

For each query, get the best matching chromosome

Returns query on target association

Return type dict

get_summary_stats()

Load summary statistics from file

Returns summary object or None if summary not already built

Return type dict

is_contig_well_oriented(*lines*, *contig*, *chrom*)

Returns True if the contig is well oriented. A well oriented contig must have y increased when x increased. We check that only for highest matches (small matches must be ignored)

Parameters

- **lines** (*list*) – lines inside the contig
- **contig** (*str*) – query contig name
- **chrom** (*str*) – target chromosome name

Returns True if well oriented, False else

Return type bool

keyerror_message(*exception*, *type_f*)

Build message if contig not found in query or target

Parameters

- **exception** (*KeyError*) – exception object
- **type_f** (*str*) – type of data (query or target)

Returns error message

Return type str

limit_idy = [0.25, 0.5, 0.75]

max_nb_lines = 100000

parse_index(*index_o: list*, *index_c: dict*, *full_len: int*)

Parse index and merge too small contigs together

Parameters

- **index_o** (*list*) – index contigs order
- **index_c** (*dict*) – index contigs def
- **full_len** (*int*) – length of the sequence

Returns (new contigs def, new contigs order)

Return type (dict, list)

parse_paf(*merge_index=True*, *noise=True*)

Parse PAF file

Parameters

- **merge_index** (*bool*) – if True, merge too small contigs in index
- **noise** (*bool*) – if True, remove noise

static remove_noise(*lines*, *noise_limit*)

Remove noise from the dot plot

Parameters

- **lines** (*dict*) – lines of the dot plot, by class
- **noise_limit** (*float*) – line length limit

Returns kept lines, by class

Return type dict

reorient_contigs_in_paf(*contigs*)

Reorient contigs in the PAF file

Parameters **contigs** – contigs to be reoriented

reverse_contig(*contig_name*)

Reverse contig

Parameters **contig_name** (*str*) – contig name

save_json(*out*)

Save D3.js data to json

Parameters **out** (*str*) – output file path

set_sorted(*is_sorted*)

Change sorted status

Parameters **is_sorted** (*bool*) – new sorted status

sort()

Sort contigs according to reference target and reorient them if needed

dgenies.lib.parsers module

Define tools parsers here

Each parser (main function) must have 2 and only 2 arguments: - First argument: input file which is the tool raw output - Second argument: finale PAF file

Returns True if parse succeed, else False

dgenies.lib.parsers.**maf**(*in_maf*, *out_paf*)

Maf parser

Parameters

- **in_maf** (*str*) – input maf file path
- **out_paf** (*str*) – output paf file path

Returns True if success, else False

dgenies.lib.parsers.**mashmap2paf**(*in_paf*, *out_paf*)

dgenies.lib.upload_file module

class dgenies.lib.upload_file.**UploadFile**(*name*, *type_f=None*, *size=None*, *not_allowed_msg=''*)

Bases: object

Manage uploaded files

Parameters

- **name** (*str*) – File name
- **type_f** (*str*) – file MIME type
- **size** (*int*) – file size in bytes
- **not_allowed_msg** (*str*) – error to add for not allowed file

get_file()

Get file object

Returns file object

Return type dict

dgenies.lib.validators module

Define formats validators here (for alignment files)

Each validator (main function) has a name which is exactly the name of the format in the aln-formats.yaml file. Only 1 argument to this function: - Input file to check

Secondary functions must start with _

Validators for non-mapping files must start with "v_"

Returns True if file is valid, else False

dgenies.lib.validators.**maf**(*in_file*)

Maf validator

Parameters **in_file** (*str*) – maf file to test

Returns True if valid, else False

Return type bool

dgenies.lib.validators.**paf**(*in_file*, *n_max=None*)

Paf validator

Parameters

- **in_file** (*str*) – paf file to test
- **n_max** (*int*) – number of lines to test (default: None for all)

Returns True if valid, else False

Return type bool

dgenies.lib.validators.**v_idx**(*in_file*)

Index file validator

Parameters **in_file** (*str*) – index file to test

Returns True if valid, else False

Return type bool

Module contents

dgenies.bin package

Submodules

dgenies.bin.clean_jobs module

dgenies.bin.clean_jobs.**parse_data_folders**(*app_data*, *gallery_jobs*, *now*, *max_age*, *fake=False*)

Parse data folder and remove too old jobs

Parameters

- **app_data** – folder where jobs are stored
- **gallery_jobs** (*list*) – id of jobs which are inside the gallery
- **now** (*float*) – current timestamp
- **max_age** (*dict*) – remove all files & folders older than this age. Define it for each category (uploads, data, error, ...)
- **fake** (*bool*) – if True, just print files to delete, without delete them

Returns

dgenies.bin.clean_jobs.**parse_database**(*app_data*, *max_age*, *fake=False*)

Parse database and remove too old jobs (from database and from disk)

Parameters

- **app_data** (*str*) – folder where jobs are stored
- **max_age** (*dict*) – remove all files & folders older than this age. Define it for each category (uploads, data, error, ...)
- **fake** (*bool*) – if True, just print files to delete, without delete them

Returns id jobs which are in the gallery (not removed independently of their age)

Return type list

dgenies.bin.clean_jobs.**parse_upload_folders**(*upload_folder*, *now*, *max_age*, *fake=False*)

Parse upload folders and remove too old files and folders

Parameters

- **upload_folder** (*str*) – upload folder path
- **now** (*float*) – current timestamp
- **max_age** (*dict*) – remove all files & folders older than this age. Define it for each category (uploads, data, error, ...)
- **fake** (*bool*) – if True, just print files to delete, without delete them

dgenies.bin.filter_contigs module

class dgenies.bin.filter_contigs.**Filter**(*fasta*, *index_file*, *type_f*, *min_filtered=0*, *split=False*, *out_fasta=None*, *replace_fa=False*)

Bases: object

Filter of a fasta file: remove too small contigs

Parameters

- **fasta** (*str*) – fasta file path
- **index_file** (*str*) – index file path
- **type_f** (*str*) – type of sample (query or target)
- **min_filtered** (*int*) – minimum number of large contigs to allow filtering
- **split** (*bool*) – are contigs split
- **out_fasta** (*str*) – output fasta file path
- **replace_fa** (*bool*) – if True, replace fasta file

filter()

Run filter of contigs

Returns True if success, else False

Return type bool

dgenies.bin.index module

class dgenies.bin.index.**Index**

Bases: object

Manage Fasta Index

static load(*index_file*, *merge_splits=False*)

Load index

Parameters

- **index_file** – index file path
- **merge_splits** (*bool*) – if True, merge split contigs together

Returns

- [0] sample name
- [1] contigs order
- [2] contigs size
- [3] reversed status for each contig
- [4] absolute start position for each contig
- [5] total len of the sample

Return type (str, list, dict, dict, dict, int)

static save(*index_file*, *name*, *contigs*, *order*, *reversed_c*)

Save index

Parameters

- **index_file** (*str*) – index file path
- **name** (*str*) – sample name
- **contigs** (*dict*) – contigs size
- **order** (*list*) – contifs order
- **reversed_c** (*dict*) – reversed status for each contig

dgenies.bin.index.**index_file**(*fasta_path*, *fasta_name*, *out*, *write_fa=None*)

Index fasta file

Parameters

- **fasta_path** (*str*) – fasta file path
- **fasta_name** (*str*) – sample name
- **out** (*str*) – output index file
- **write_fa** (*str*) – file path of the new fasta file to write, None to don't save fasta in a new file

Returns

- [0] True if success, else False
- [1] Number of contigs
- [2] Error message

Return type (bool, int, str)

dgenies.bin.local_scheduler module

dgenies.bin.local_scheduler.**cleaner**()

Exit DRMAA session at program exit

dgenies.bin.local_scheduler.**get_prep_scheduled_jobs**()

Get list of jobs ready to be prepared (all data is downloaded and parsed)

Returns list of jobs

Return type list

dgenies.bin.local_scheduler.**get_preparing_jobs_cluster_nb**()

Get number of jobs in preparation step (for cluster runs)

Returns number of jobs

Return type int

dgenies.bin.local_scheduler.**get_preparing_jobs_nb**()

Get number of jobs in preparation step (for local runs)

Returns number of jobs

Return type int

dgenies.bin.local_scheduler.**get_scheduled_cluster_jobs**()

Get list of jobs ready to be started (for cluster runs)

Returns list of jobs

Return type list

dgenies.bin.local_scheduler.**get_scheduled_local_jobs**()

Get list of jobs ready to be started (for local runs)

Returns list of jobs

Return type list

dgenies.bin.local_scheduler.**move_job_to_cluster**(*id_job*)

Change local job to be run on the cluster

Parameters **id_job** –

Returns

dgenies.bin.local_scheduler.**parse_args**()

Parse command line arguments and define DEBUG and LOG_FILE constants

dgenies.bin.local_scheduler.**parse_started_jobs**()

Parse all started jobs: check all is OK, change jobs status if needed. Look for died jobs

Returns (list of id of jobs started locally, list of id of jobs started on cluster)

Return type (list, list)

dgenies.bin.local_scheduler.**parse_uploads_asks**()

Parse asks for an upload: allow new uploads when other end, remove expired sessions, ...

dgenies.bin.local_scheduler.**prepare_job**(*id_job*)

Launch job preparation of data

Parameters **id_job** (*str*) – job id

dgenies.bin.local_scheduler.**start_job**(*id_job*, *batch_system_type='local'*)

Start a job (mapping step)

Parameters

- **id_job** (*str*) – job id
- **batch_system_type** (*str*) – local, slurm or sge

dgenies.bin.merge_splitted_chrms module

class dgenies.bin.merge_splitted_chrms.**Merger**(*paf_in*, *paf_out*, *query_in*, *query_out*, *debug=False*)

Bases: object

Merge splitted contigs together in PAF file

Parameters

- **paf_in** (*str*) – input PAF file path
- **paf_out** (*str*) – output PAF file path
- **query_in** (*str*) – input query index file path

- **query_out** (*str*) – output query index file path
- **debug** (*bool*) – True to enable debug mode

load_query_index(*index*)

Load query index

Parameters **index** (*str*) – index file path

Returns

- [0] contigs length
- [1] splitted contigs length
- [2] sample name

Return type (dict, dict, str)

merge()

Launch the merge

static merge_paf(*paf_in*, *paf_out*, *contigs*, *contigs_split*)

Do merge PAF staff

Parameters

- **paf_in** (*str*) – path of input PAF with split contigs
- **paf_out** (*str*) – path of output PAF where split contigs are now merged together
- **contigs** (*dict*) – contigs size
- **contigs_split** (*dict*) – split contigs size

static write_query_index(*index*, *contigs*, *q_name*)

Save new query index

Parameters

- **index** (*str*) – index file path
- **contigs** (*dict*) – contigs size
- **q_name** (*str*) – sample name

dgenies.bin.merge_splitted_chrms.**parse_args**()

Parse command line arguments

Returns arguments

Return type argparse.Namespace

dgenies.bin.sort_paf module

class dgenies.bin.sort_paf.**Sorter**(*input_f*, *output_f*)

Bases: object

Sort PAF file by match size

Parameters

- **input_f** (*str*) – input fasta file path
- **output_f** (*str*) – output fasta file path

sort()

Launch sort staff

dgenies.bin.split_fa module

class dgenies.bin.split_fa.**Splitter**(*input_f*, *name_f*, *output_f*, *size_c=10000000*, *query_index='query_split.idx'*, *debug=False*)

Bases: object

Split large contigs in smaller ones

Parameters

- **input_f** (*str*) – input fasta file path
- **name_f** (*str*) – sample name
- **output_f** (*str*) – output fasta file path
- **size_c** (*int*) – size of split contigs
- **query_index** (*str*) – index file path for query
- **debug** (*bool*) – True to enable debug mode

flush_contig(*fasta_str*, *chr_name*, *size_c*, *enc*, *index_f*)

split()

Split contigs in smaller ones staff

Returns True if the input Fasta is correct, else False

static split_contig(*name*, *sequence*, *block_sizes*)

static write_contig(*name*, *fasta*, *o_file*)

dgenies.bin.split_fa.**parse_args**()

Module contents

Submodules

dgenies.config_reader module

dgenies.database module

class dgenies.database.**BaseModel**(**args*, ***kwargs*)

Bases: peewee.Model

DoesNotExist

alias of dgenies.database.BaseModelDoesNotExist

classmethod connect()

id = <AutoField: BaseModel.id>

class dgenies.database.**Database**

Bases: object

nb_open = 0

class dgenies.database.**Gallery**(**args*, ***kwargs*)

Bases: *dgenies.database.BaseModel*

DoesNotExist

alias of dgenies.database.GalleryDoesNotExist

id = <AutoField: Gallery.id>

job = <ForeignKeyField: Gallery.job>

job_id = <ForeignKeyField: Gallery.job>

name = <CharField: Gallery.name>

picture = <CharField: Gallery.picture>

query = <CharField: Gallery.query>

target = <CharField: Gallery.target>

class dgenies.database.**Job**(**args*, ***kwargs*)

Bases: *dgenies.database.BaseModel*

DoesNotExist

alias of dgenies.database.JobDoesNotExist

batch_type = <CharField: Job.batch_type>

date_created = <DateTimeField: Job.date_created>

email = <CharField: Job.email>

error = <CharField: Job.error>

gallery_set

id = <AutoField: Job.id>

id_job = <CharField: Job.id_job>

id_process = <IntegerField: Job.id_process>

mem_peak = <IntegerField: Job.mem_peak>

options = <CharField: Job.options>

status = <CharField: Job.status>

time_elapsed = <IntegerField: Job.time_elapsed>

tool = <CharField: Job.tool>

class dgenies.database.**MyRetryDB**(*database*, *thread_safe=True*, *autorollback=False*, *field_types=None*, *operations=None*, *autocommit=None*, *autoconnect=True*, ***kwargs*)

Bases: *dgenies.database.RetryOperationalError*, peewee.MySQLDatabase

class dgenies.database.**RetryOperationalError**

Bases: object

execute_sql(*sql*, *params=None*, *commit=True*)

class dgenies.database.**Session**(**args*, ***kwargs*)

Bases: *dgenies.database.BaseModel*

DoesNotExist

alias of dgenies.database.SessionDoesNotExist

ask_for_upload(*change_status=False*)

date_created = <DateTimeField: Session.date_created>

id = <AutoField: Session.id>

keep_active = <BooleanField: Session.keep_active>

last_ping = <DateTimeField: Session.last_ping>

classmethod new(*keep_active=False*)

ping()

s_id = <CharField: Session.s_id>

status = <CharField: Session.status>

upload_folder = <CharField: Session.upload_folder>

dgenies.tools module

class dgenies.tools.**Tool**(*name*, *exec*, *command_line*, *all_vs_all*, *max_memory*, *label=None*, *threads=1*, *exec_cluster=None*, *threads_cluster=None*, *parser=None*, *split_before=False*, *help=None*, *order=None*, *options=None*)

Bases: object

Create a new tool

Parameters

- **command_line** – command line to launch the tool
- **all_vs_all** – command line in all_vs_all mode (None if not available for the tool)
- **max_memory** – max memory the tool is supposed to use (ex: 40G) - for cluster submissions
- **label** – Name to display for user
- **parser** – name of the function in dgenies.lib.functions to launch after mapping to have a correct PAF out file
- **split_before** (*bool*) – True to split contigs before mapping
- **help** – help message to show in run form
- **order** – order to show in run mode
- **options** – list of options for the tool

dgenies.views module

dgenies.views.**ask_upload**()

Ask for upload: to keep a max number of concurrent uploads

dgenies.views.**build_fasta**(*id_res*)

Generate the fasta file of query

Parameters **id_res** (*str*) – job id

dgenies.views.**build_query_as_reference**(*id_res*)

Build fasta of query with contigs order like reference

Parameters **id_res** (*str*) – job id

dgenies.views.**contact**()

Contact page

dgenies.views.**delete_job**(*id_res*)

Delete a job

Parameters **id_res** (*str*) – job id

dgenies.views.**dl_fasta**(*id_res*, *filename*)

Download fasta file

Parameters

- **id_res** (*str*) – job id
- **filename** (*str*) – file name (not used, but can be in the URL to define download filename to the browser)

dgenies.views.**documentation_dotplot**()

Documentation dotplot page

dgenies.views.**documentation_formats**()

Documentation formats page

dgenies.views.**documentation_result**()

Documentation result page

dgenies.views.**documentation_run**()

Documentation run page

dgenies.views.**download_file**(*id_res*, *filename*)

Download a file from a job

Parameters

- **id_res** (*str*) – job id
- **filename** (*str*) – file name

dgenies.views.**download_paf**(*id_res*)

Download PAF file of a job

Parameters **id_res** (*str*) – job id

dgenies.views.**free_noise**(*id_res*)

Remove noise from the dot plot

Parameters **id_res** (*str*) – job id

dgenies.views.**gallery**()

Gallery page

dgenies.views.**gallery_file**(*filename*)

Getting gallery illustration

Parameters **filename** – filename of the PNG file

dgenies.views.**get_backup_file**(*id_res*)

Download archive backup file of a job

Parameters **id_res** (*str*) – job id

dgenies.views.**get_file**(*file*, *gzip=False*)

Download a file

Parameters

- **file** (*str*) – filename
- **gzip** (*bool*) – is file gzipped?

dgenies.views.**get_filter_out**(*id_res*, *type_f*)

Download filter fasta, when it has been filtered before job run

Parameters

- **id_res** (*str*) – job id
- **type_f** (*str*) – type of fasta (query or target)

dgenies.views.**get_filter_out_query**(*id_res*)

Download query filtered fasta, when it has been filtered before job run

Parameters **id_res** (*str*) – job id

dgenies.views.**get_filter_out_target**(*id_res*)

Download target filtered fasta, when it has been filtered before job run

Parameters **id_res** (*str*) – job id

dgenies.views.**get_graph**()

Get dot plot data for a job

dgenies.views.**get_query_as_reference**(*id_res*)

Get fasta of query with contigs order like reference

Parameters **id_res** (*str*) – job id

dgenies.views.**get_tools_options**(*tool_name*, *chosen_options*)

Transform options chosen in javascript into parameters

Returns True is chosen options are valid + a string containing optional parameters to use with tool

Return type boolean, str

dgenies.views.**get_viewer_html**(*id_res*)

Get HTML file with offline interactive viewer inside

Parameters **id_res** (*str*) – job id

dgenies.views.**global_templates_variables**()

Global variables used for any view

dgenies.views.**install**()

Documentation: how to install? page

dgenies.views.**launch_analysis**()

Launch the job

dgenies.views.**legal**(*page*)

Display legal things

dgenies.views.**main**()

Index page

dgenies.views.**no_assoc**(*id_res*)

Get contigs that match with None target

Parameters **id_res** (*str*) – job id

dgenies.views.**ping_upload**()

When upload waiting, ping to be kept in the waiting line

dgenies.views.**post_query_as_reference**(*id_res*)

Launch build fasta of query with contigs order like reference

Parameters **id_res** (*str*) – job id

dgenies.views.**qt_assoc**(*id_res*)

Query - Target association TSV file

Parameters **id_res** –

Returns

dgenies.views.**result**(*id_res*)

Result page

Parameters **id_res** (*str*) – job id

dgenies.views.**reverse_contig**(*id_res*)

Reverse contig order

Parameters **id_res** (*str*) – job id

dgenies.views.**run**()

Run page

dgenies.views.**run_test**()

Run test page (used to simulate a real client run)

dgenies.views.**send_mail**(*id_res*)

Send mail

Parameters **id_res** (*str*) – job id

dgenies.views.**sort_graph**(*id_res*)

Sort dot plot to reference

Parameters **id_res** (*str*) – job id

dgenies.views.**status**(*id_job*)

Status page

Parameters **id_job** (*str*) – job id

dgenies.views.**summary**(*id_res*)

Get Dot plot summary data

Parameters **id_res** (*str*) – job id

dgenies.views.**upload**()

Do upload of a file

Module contents

dgenies

dgenies.**launch**(*mode='webserver'*, *debug=False*)

Launch the application

Parameters

- **mode** (*str*) – webserver or standalone
- **debug** (*bool*) – True to enable debug mode

Returns flask app object

Return type Flask

4.3 Javascript client part

4.3.1 Javascript client functions

dgenies

dgenies.**init**(*all_jobs*, *mode*)

Initialise dgenies client app

Arguments

- **all_jobs** (`array()`) – list of user jobs (in standalone mode, empty in other modes)
- **mode** (`string()`) – server mode (standalone or webserver)

dgenies.**ajax**(*url*, *data*, *success*, *error*, *method*)

Ajax server call

Arguments

- **url** – url to call
- **data** – data to send

- **success** – success function
- **error** – error function
- **method** – method (GET, POST, ...)

dgenies.**fill_select_zones**(*x_targets*, *y_contigs*)

Fill list of zones on select boxes (contigs and chromosomes)

Arguments

- **x_targets** (array()) – list of chromosomes of target
- **y_contigs** (array()) – list of contigs of query

dgenies.**get**(*url*, *data*, *success*, *error*)

Get server call

Arguments

- **url** – url to call
- **data** – data to send
- **success** – success function
- **error** – error function

dgenies.**hide_loading**()

Hide loading popup

dgenies.**notify**(*text*, *type*, *delay*)

Show new notification

Arguments

- **text** (string()) – notification text
- **type** (string()) – notification type (danger, warning, info, success) according to Bootstrap Notify library
- **delay** (int()) – time before hide notification

dgenies.**numberWithCommas**(*x*)

Show human readable number higher than 1000: 1000 -> 1,000

Arguments

- **x** (int()) – number

Returns **string** – human readable number

dgenies.**post**(*url*, *data*, *success*, *error*, *async*)

Post server call

Arguments

- **url** – url to call
- **data** – data to send
- **success** – success function
- **error** – error function
- **async** – make call asynchronous

dgenies.**reset_loading_message**()

Reset loading message to its default value

dgenies.**save_cookies**(*cookies*)

Save cookie on the browser

Arguments

- **cookies** (array()) – list of jobs

dgenies.**set_loading_message**(*message*)

Change loading message on current popup

Arguments

- **message** (string()) – new message

dgenies.**show_loading**(*message*, *width*)

Show loading popup

Arguments

- **message** (string()) – loading message
- **width** (int()) – popup width

dgenies.**update_results**(*results:*)

Update list of jobs

Arguments

- **results:** (array()) – new list of jobs

dgenies.run

dgenies.run.**init**(*s_id*, *allowed_ext*, *max_upload_file_size*, *target_example*, *query_example*, *tool_has_ava*)

Initialise app for run page

Arguments

- **s_id** (string()) – session id
- **allowed_ext** (object()) –
- **max_upload_file_size** (int()) – maximum upload file size
- **target_example** (string()) – target example pseudo path
- **query_example** (string()) – query example pseudo path
- **tool_has_ava** (object()) – defines if each available tool has an all-vs-all mode

dgenies.run.**add_error**(*error*)

Add an error to the form

Arguments

- **error** (string()) – error message to display

dgenies.run.**allowed_file**(*filename*, *formats*)

Check if a file has a valid format

Arguments

- **filename** (string()) – filename
- **formats** (array()) – expected file format

Returns **boolean** – true if valid, else false

dgenies.run.**ask_for_upload**()

Ask server to start uploads

dgenies.run.**change_fasta_type**(*fasta*, *type*, *keep_url*)

Change source of fasta (local or url)

Arguments

- **fasta** (string()) – type of fasta (query, target, ...)
- **type** (string()) – source of fasta (local or url)
- **keep_url** (boolean()) – if true, keep url in form, else empty it

dgenies.run.**check_url**(*url*)

Check if an URL is valid

Arguments

- **url** (string()) – the url to check

Returns **boolean** – true if valid, else false

dgenies.run.**disable_form**()

Disable run form

dgenies.run.**do_submit**()

Do form submit staff (done once all uploads are done successfully)

dgenies.run.**enable_form**()

Enable run form

dgenies.run.**fill_examples**()

Fill inputs with example data

dgenies.run.**get_file_size_str**(*size*)

Get file size (human readable)

Arguments

- **size** (int()) – file size in bytes

Returns **string** – human readable size

dgenies.run.**hide_loading**(*fasta*)

Hide loading for a fasta uploaded file

Arguments

- **fasta** (string()) – uploaded file type (query, target, ...)

dgenies.run.**hide_success**(*fasta*)

Hide success on a file

Arguments

- **fasta** (string()) – type of file (query, target, ...)

dgenies.run.**init_fileuploads**()

Init file upload forms

dgenies.run.**ping_upload**()

Ping server: we still upload or wait for upload

dgenies.run.**reset_errors**()

Remove all errors displayed

dgenies.run.**reset_file_form**(*tab*, *except_backup*)

Reset all inputs in the given tab

Arguments

- **tab** (string()) – tab name
- **except_backup** (boolean()) – if true, don't reset backup input

dgenies.run.**reset_file_input**(*inp_name*)

Reset file input

Arguments

- **inp_name** (string()) – type of fasta (query, target, ...)

dgenies.run.**restore_form**()

Restore run form

dgenies.run.**set_events**()

Initialise events

dgenies.run.**set_filename**(*name*, *fasta*)

Set filename for input fasta

Arguments

- **name** (string()) – filename
- **fasta** (string()) – type of fasta (query, target, ...)

dgenies.run.**show_global_loading**()

Show global loading

dgenies.run.**show_loading**(*fasta*)

Show loading for a fasta uploading file

Arguments

- **fasta** (string()) – uploading file type (query, target, ...)

dgenies.run.**show_success**(*fasta*)

Show success: file uploaded successfully

Arguments

- **fasta** (string()) – uploaded type of file (query, target, ...)

dgenies.run.**show_tab**(*tab*)

Change displayed tab

Arguments

- **tab** (string()) – id of the tab to show

dgenies.run.**start_uploads**()

Launch upload of files

dgenies.run.**submit**()

Submit form

dgenies.run.**upload_next**()

Upload next file

Returns **boolean** – true if there is a next upload, else false and run submit

dgenies.run.**__upload_server_error**(*fasta*, *data*)

Notify and reanable form on upload server error

Arguments

- **fasta** (string()) – fasta file (name) which fails
- **data** – data from server call

dgenies.run.**_init_fileupload**(*ftype*, *formats*, *position*)

Init file upload forms staff

Arguments

- **ftype** (string()) – type of file (query, target, ...)
- **formats** (array()) – valid formats
- **position** (int()) – position of file in the upload queue

dgenies.run.**_set_file_event**(*ftype*)

Initialise file change events

Arguments

- **ftype** (string()) – type of file (query, target, ...)

dgenies.run.**_set_file_select_event**(*ftype*)

Initialise change source of file (local, url) event

Arguments

- **ftype** (string()) – type of file (query, target, ...)

dgenies.run.**_start_upload**(*ftype*, *fname*)

Start upload staff

Arguments

- **ftype** – type of file (query, target, ...)
- **fname** – fasta name

Returns **boolean** – true if has uploads

dgenies.documentation

dgenies.documentation.**init**()
Initialise app for documentation page

dgenies.documentation.**fix_links_headers**()
Fix link in headers behavior (due to top bar fixed position - CSS)

dgenies.documentation.**goto**(*elem*)
Scroll to a JQuery element

Arguments

- **elem** – JQuery element

dgenies.result

dgenies.result.**init**(*id_res*)
Initialise app for result app

Arguments

- **id_res** (string()) – job id

dgenies.result.**add_to_list**()
Update list of results from cookie

dgenies.result.**remove_job_from_cookie**(*job*)
Remove a job in cookie

Arguments

- **job** (string()) – job id to remove

dgenies.result.controls

dgenies.result.controls.**init**()
Initialise controls of the result page

dgenies.result.controls.**delete_job**()
Ask confirm for delete current job

dgenies.result.controls.**do_delete_job**()
Delete current job (confirmed)

dgenies.result.controls.**launch_hide_noise**()
Hide noise

dgenies.result.controls.**launch_reverse_contig**()
Build reverse of a contig

dgenies.result.controls.**launch_sort_contigs**()
Build contigs sort

dgenies.result.controls.**select_zone**()
Select zone with select boxes

dgenies.result.controls.**summary**()
Build summary

dgenies.result.export

dgenies.result.export.**ask_export_fasta**()
Show export dialog

dgenies.result.export.**dl_fasta**(*gzip*)
Download query fasta file

Arguments

- **gzip** (boolean()) – if true, gzip the file

dgenies.result.export.**export**()
Manage exports

dgenies.result.export.**export_association_table**()
Download association table between queries and targets

dgenies.result.export.**export_backup_file**()
Download backup file of the project

dgenies.result.export.**export_fasta**(*compress*)
Export fasta file

Arguments

- **compress** (boolean()) – if true compress (gzip) the file

dgenies.result.export.**export_no_association_file**(*to:*)
Export list of contigs with no association with any target or any query

Arguments

- **to:** (string()) – query or target

dgenies.result.export.**export_offline_viewer**()
Download offline viewer

dgenies.result.export.**export_paf**()
Download PAF alignment file

dgenies.result.export.**export_png**()
Export dot plot as PNG

dgenies.result.export.**export_query_as_reference_fasta_standalone**()
Export query like reference fasta file (standalone mode)

dgenies.result.export.**export_query_as_reference_fasta_webserver**()
Export query like reference fasta file (webserver mode)

dgenies.result.export.**export_svg**()
Export dot plot as SVG

dgenies.result.export.**get_svg**(*width*)

Build SVG tag and content

Arguments

- **width** (string()) – svg width size (with unit [px])

Returns **string** – svg tag and content

dgenies.result.export.**save_file**(*blob*, *format*)

Save file

Arguments

- **blob** (Blob()) – file content to save
- **format** (string()) – file format

dgenies.result.summary

dgenies.result.summary.**export_png**()

Export summary to png

dgenies.result.summary.**export_svg**()

Export summary to svg

dgenies.result.summary.**export_tsv**()

Export summary to tsv

dgenies.result.summary.**get_svg**()

Get SVG picture of the summary

Returns **string** – svg picture

dgenies.result.summary.**save_file**(*blob*, *format*)

Save to a file

Arguments

- **blob** – data to save
- **format** (string()) – file format

dgenies.result.summary.**show**(*percents:*)

Show summary window

Arguments

- **percents:** (object()) – percents for each identity category

dgenies.result.summary.**_get_label**(*percent_class*)

Get label of the percent class

Arguments

- **percent_class** (string()) – percent class

Returns **string** – percent class label

dgenies.status

dgenies.status.**init**(*status*, *mode*)

initialise the app for status page

Arguments

- **status** (string()) – job status
- **mode** (string()) – server mode (standalone or webserver)

dgenies.status.**autoreload**()

Page autoreload periodically

d3.boxplot

d3.boxplot.**init**(*id_res*, *from_file*)

Initialize dotplot

Arguments

- **id_res** (string()) – job id
- **from_file** (boolean()) – true to load data from a file (default: false, load from server)

d3.boxplot.**change_color_theme**(*theme*)

Change color theme to the given one

Arguments

- **theme** (string()) – theme name

d3.boxplot.**draw**(*x_contigs*, *x_order*, *y_contigs*, *y_order*)

Draw dot plot

Arguments

- **x_contigs** (object()) – length associated to each contig of the query
- **x_order** (array()) – order of query contigs
- **y_contigs** (object()) – length associated to each chromosome of the target
- **y_order** (array()) – order of target chromosomes

d3.boxplot.**draw_axis_bckgd**()

Draw backgrounds of all axis

d3.boxplot.**draw_bottom_axis**(*x_max*, *x_min*)

Draw bottom axis

Arguments

- **x_max** (int()) – max value of x on the X axis
- **x_min** (int()) – min value of x on the X axis

d3.boxplot.**draw_left_axis**(*y_max*, *y_min*)

Draw left axis

Arguments

- **y_max** (int()) – max value of y on the Y axis

- **y_min** (int()) – min value of y on the Y axis

d3.boxplot.**draw_legend**()

Draw legend

d3.boxplot.**draw_lines**(*lines*, *x_len*, *y_len*)

Draw matches on dot plot

Arguments

- **lines** (object()) – matches definition
- **x_len** (number()) – total len of target
- **y_len** (number()) – total len of query

d3.boxplot.**draw_right_axis**(*y_zones*)

Draw right axis

Arguments

- **y_zones** (object()) – name of contigs of the query

d3.boxplot.**draw_top_axis**(*x_zones:*)

Draw top axis

Arguments

- **x_zones:** (object()) – name of chromosomes of the target

d3.boxplot.**get_human_readable_size**(*nbases*, *precision*, *space*)

Get human readable size in Kb or Mb for a number in bases

Arguments

- **nbases** (int()) – size in bases
- **precision** (int()) – unit to use (auto: select according to number size)
- **space** (string()) – space before unit (space or non-breaking space for example)

Returns **string** – human readable size

d3.boxplot.**launch**(*res*, *update*, *noise_change*)

Launch draw of dot plot

Arguments

- **res** (string()) –
- **update** (boolean()) – if true, just update the existing dot plot (don't initialize events)
- **noise_change** (boolean()) – if false, set noise to true

d3.boxplot.**select_query**(*y*)

Find query contig where the user click

Arguments

- **y** (float()) – coordinate on Y axis

Returns **string|null** – contig name

d3.boxplot.**select_target**(*x*)

Find target chromosome where the user click

Arguments

- **x** (float()) – coordinate on X axis

Returns **string|null** – chromosome name

d3.boxplot.**select_zone**(*x*, *y*, *x_zone*, *y_zone*, *force*)

Find zone (query contig and target chromosome) based on coordinates

Arguments

- **x** (float()) – coordinate on X axis
- **y** (float()) – coordinate on Y axis
- **x_zone** (string()) – selected chromosome on X axis (target)
- **y_zone** (string()) – selected contig on Y axis (query)
- **force** (boolean()) – if true, select zone even if a zone is already selected

d3.boxplot.**switch_color_theme**()

Switch to next color theme

d3.boxplot.**zoom_bottom_axis**()

Zoom on bottom axis

d3.boxplot.**zoom_left_axis**()

Zoom on left axis

d3.boxplot.**__draw_idy_lines**(*idy*, *lines*, *x_len*, *y_len*)

Draw matches on dot plot for the given identity class

Arguments

- **idy** (string()) – identity class of matches to draw
- **lines** (object()) – matches definitions
- **x_len** (number()) – total length of target
- **y_len** (number()) – total length of query

d3.boxplot.**__lineFunction**(*d*, *min_size*, *max_size*, *x_len*, *y_len*)

Build line data for D3.js

Arguments

- **d** (object()) – data object of the line
- **min_size** (int()) – min size of line. Beside it, don't draw the line
- **max_size** (int|null()) – max size of line. Over it, don't draw the line
- **x_len** (number()) – length of x (target)
- **y_len** (number()) – length of y (query)

Returns **string** – path object

d3.boxplot.**_get_line_len**(*line*)

Get length of a given line

Arguments

- **line** (array()) – line object

Returns **number** – line length

d3.boxplot.**_sort_color_idy**(*a*, *b*)

Sort function key for color identity

Arguments

- **a** –
- **b** –

Returns **number** –

d3.boxplot.**_sort_lines**(*l1*, *l2*)

Sort lines with their length (DESC)

Arguments

- **l1** (array()) – line object
- **l2** (array()) – line object

Returns **number** –

d3.boxplot.**_sort_lines_by_idy**(*l1*, *l2*)

Sort lines with their identity (DESC)

Arguments

- **l1** (array()) – line object
- **l2** (array()) – line object

Returns **number** –

d3.boxplot.events

d3.boxplot.events.**init**()

Initialise events

d3.boxplot.events.**filter_identity**(*min_idy*)

Remove low identity matches

Arguments

- **min_idy** (number()) – minimum of identity. Beside it, hide matches

d3.boxplot.events.**filter_size**(*min_size*)

Remove too small matches

Arguments

- **min_size** (number()) – minimum size. Beside it, hide matches

d3.boxplot.events.**init_context_menu**()

Initialise context menu

d3.boxplot.events.**set_break_lines_visibility**(*value:*)

Set break lines visibility: color and thickness, or hidden

Arguments

- **value:** (string()) – visibility value: "0"-> hidden to "5" -> max visibility value

d3.boxplot.events.**stroke_linecap**(*rounded*)

If stroke precision checked, strole-linecap is set to "butt". Else "round" to improve visibility of matches

Arguments

- **rounded** (boolean()) – if true, improve bisibility by add round cap to lines

d3.boxplot.events.**stroke_width**(*width*)

Change matches lines stroke width

Arguments

- **width** (string()) – new width class ("1", "2", or "3")

d3.boxplot.mousetip

$.fn.**mousetip**(*my_tip*, *relative_to*, *x*, *y*)

Mouse tip basis

Arguments

- **my_tip** –
- **relative_to** –
- **x** (int()) –
- **y** (int()) –

d3.boxplot.mousetip.**init**()

Initialise tooltip

d3.boxplot.mousetip.**getColorByBgColor**(*bgColor*)

Get color (black/white) depending on bgColor so it would be clearly seen.

Arguments

- **bgColor** –

Returns **string** –

d3.boxplot.mousetip.**get_label**(*label*)

get label to show

Arguments

- **label** (string()) – initial label

Returns **string** – new label

d3.boxplot.mousetip.**get_match**(*e*)

Get match override by mouse cursor

Arguments

- **e** – mouse event

Returns **Object** –

d3.boxplot.mousetip.**hide**()

Hide tooltip

d3.boxplot.zoom

d3.boxplot.zoom.**init**()

Initialize zoom.init module

d3.boxplot.zoom.**click**()

Click event action

d3.boxplot.zoom.**mousedown**()

Mousedown event action

d3.boxplot.zoom.**mouseup**()

Mouseup event action

d3.boxplot.zoom.**reset_scale**(*temp*, *after*, *force*)

Reset scale

Arguments

- **temp** (boolean()) – if true, reset it temporarily
- **after** (function()) – function to launch after staff
- **force** (boolean()) – do it even if events are disabled

Returns **boolean** – true if done, else false

d3.boxplot.zoom.**restore_scale**(*transform:*)

Restore previous scale

Arguments

- **transform:** – transform object

d3.boxplot.zoom.**translate**()

Translate event action

d3.boxplot.zoom.**zoom**()

Zoom staff

d3.boxplot.zoom.**_cursor_pos**(*rect*)

Get cursor position

Arguments

- **rect** (DOMRect()) – if given, dont get it from DOM

Returns

PYTHON MODULE INDEX

b

d

INDEX

Symbols

A

B

C

D

E

F

G

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W