NucleoMiner2 Documentation

Release 2.3.45

Florent CHUFFART, Jean-Baptiste VEYRIERAS, Gael YVERT

July 18, 2014

CHAPTER

ONE

README / DOCUMENTATION FOR *NUCLEOMINER2*

NucleoMiner2 offers Python API and R package allowing to perform quantitative analysis of epigenetic marks on individual nucleosomes. It was developed to detect natural Single-Nucleosome Epi-Polymorphisms (SNEP) from MNase-seq and ChIP-seq data.

1.1 License

1.2 Installation Instructions

1.2.1 Links

NucleoMiner2 home page and documentation are available here:

- https://forge.cbp.ens-lyon.fr/redmine/projects/nucleominer

The Yvert lab web page is accessible here:

- http://www.ens-lyon.fr/LBMC/gisv/

1.2.2 Installation

The first installation step is to retrieve the source code of MyLabStocks. You can do this by typing the following command in a terminal.

```
git clone http://forge.cbp.ens-lyon.fr/git/nucleominer
```

Prerequisites

To work properly, NucleoMiner2 needs that the following free software are installed and made available on your system:

- Bowtie2 http://bowtie-bio.sourceforge.net/bowtie2
- SAMtools http://samtools.sourceforge.net
- bedtools http://code.google.com/p/bedtools/
- TemplateFilter http://compbio.cs.huji.ac.il/NucPosition/TemplateFiltering

It also requires the following R packages to be installed on your system:

- fork
- rjson
- seqinr
- cachecache https://forge.cbp.ens-lyon.fr/redmine/projects/cachecache
- bot https://forge.cbp.ens-lyon.fr/redmine/projects/bot
- nucleominer https://forge.cbp.ens-lyon.fr/redmine/projects/nucleominer

the three last packages are available in the git repository they could be install by typing the following command in your terminal:

```
cd nucleominer
R CMD INSTALL R CMD INSTALL doc/Chuffart_NM2_workdir/deps/bot_0.14.tar.gz\
    doc/Chuffart_NM2_workdir/deps/cachecache_0.1.tar.gz\
    build/nucleominer_2.XXX.tar.gz
```

CHAPTER

TWO

TUTORIAL

This tutorial describes steps allowing performing quantitative analysis of epigenetic marks on individual nucleosomes. We assume that files are organised according to a given hierarchy and that all command lines are launched from the project's root directory.

This tutorial is divided into two main parts. The first part covers the python script *wf.py* that aligns and converts short sequence reads. The second part covers the R scripts that extracts information (nucleosome position and indicators) from the dataset.

2.1 Experimental Dataset, Working Directory and Configuration File

2.1.1 Working Directory Organisation

The working directory...

$$$ TODO: Explain how and where retrieve the workdir

2.1.2 Retrieving Experimental Dataset

The MNase-seq and MN-ChIP-seq raw data are available at ArrayExpress (http://www.ebi.ac.uk/arrayexpress/) under accession number E-MTAB-2671.

$$$ TODO explain how organise Experimental Dataset into the *data* directory of the working directory.

We want to compare nucleosomes of 2 yeast strains: BY and RM. For each strain we performed Mnase-Seq and ChIP-Seq using an antibody recognizing the H3K14ac epigenetic mark.

The dataset is composed of 55 files organised as follows:

- 3 replicates for BY MNase Seq
  - sample 1 (5 fastq.gz files)
  - sample 2 (5 fastq.gz files)
  - sample 3 (4 fastq.gz files)
- 3 replicates for RM MNase Seq
  - sample 4 (4 fastq.gz files)
  - sample 5 (4 fastq.gz files)
  - sample 6 (5 fastq.gz files)
- 3 replicates for BY ChIP Seq H3K14ac

  - sample 36 (5 fastq.gz files)
  - sample 37 (5 fastq.gz files)
  - sample 53 (9 fastq.gz files)
- 2 replicates for RM ChIP Seq H3K14ac
  - sample 38 (5 fastq.gz files)
  - sample 39 (4 fastq.gz files)

2.1.3 Python and R Common Configuration File

First of all we define in one place some configuration variables that will be launched by python and R scripts. These variables are contained in file *configurator.py*. The execution of this python script dumps variables into the *nucleominer_config.json* file that will then be used by both R and python scripts.

To do this, go to the root directory of your project and run the following command:

```
python src/current/configurator.py
```

2.2 Preprocessing Illumina Fastq Reads for Each Sample

This preprocessing step consists of 4 main steps embedded in the *wf.py* script. They are described bellow. As a preamble, this script computes *samples*, *samples_mnase* and *strains* that will be used along the 4 steps.

`wf.`**`samples`** **= []**
: List of samples where a sample is identified by an id (key: *id*) and a strain name (key *strain*).

`wf.`**`samples_mnase`** **= []**
: List of Mnase samples.

`wf.`**`strains`** **= []**
: List of reference strains.

2.2.1 Creating Bowtie Index from each Reference Genome

For each strain, we need to create bowtie index. Bowtie index of a strain is a tree view of the genome of this strain. It will be used by bowtie to align reads. This step is performed by the following part of the *wf.py* script:

```
for strain in strains:
  per_strain_stats[strain] = create_bowtie_index(strain,
    config["FASTA_REFERENCE_GENOME_FILES"][strain], config["INDEX_DIR"],
    config["BOWTIE_BUILD_BIN"])
```

The following table summarizes the file sizes and process durations concerning this step.

strain	fasta genome file size	bowtie index file size	process duration
BY	12 Mo	25 Mo	11 s.
RM	12 Mo	24 Mo	9 s.

2.2.2 Aligning Reads to Reference Genome

Next, we launch bowtie to align reads to the reference genome. It produces a *.sam* file that we convert into a *.bed* file. Binaries for *bowtie*, *samtools* and *bedtools* are wrapped using python *subprocess* class. This step is performed by the following part of the *wf.py* script:

```
for sample in samples:
  per_sample_align_stats["sample_%s" % sample["id"]] = align_reads(sample,
    config["ALIGN_DIR"], config["LOG_DIR"], config["INDEX_DIR"],
    config["ILLUMINA_OUTPUTFILE_PREFIX"], config["BOWTIE2_BIN"],
    config["SAMTOOLS_BIN"], config["BEDTOOLS_BIN"])
```

2.2.3 Convert Aligned Reads into TemplateFilter Format

TemplateFilter uses particular input formats for reads, so it is necessary to convert the *.bed* files. TemplateFilter expect reads as follows: *chr*, *coord*, *strand* and *#read* where:

- *chr* is the number of the chromosome;
- *coord* is the coordinate of the reads;
- *strand* is *F* for forward and *R* for reverse;
- *#reads* the number of reads covering this position.

Each entry is *tab*-separated.

WARNING for reverse strands, bowtie returns the position of the first nucleotide on the left hand side, whereas TemplateFilter expects the first one on the right hand side. This step takes this into account by adding the read length (in our case 50) to the reverse reads coordinates.

This step is performed by the following part of the *wf.py* script:

```
for sample in samples:
  per_sample_convert_stats["sample_%s" % sample["id"]] = split_fr_4_TF(sample,
    config["ALIGN_DIR"], config["FASTA_INDEXES"], config["AREA_BLACK_LIST"],
    config["READ_LENGTH"],config["MAPQ_THRES"])
```

The following table summarises the number of reads, the involved file sizes and process durations concerning the two last steps. In our case, alignment process have been multithreaded over 3 cores.

id	Illumina reads	aligned and filtred reads	ratio	*.bed* file size	TF input file size	process duration
1	16436138	10199695	62,06%	1064 Mo	60 Mo	383 s.
2	16911132	12512727	73,99%	1298 Mo	64 Mo	437 s.
3	15946902	12340426	77,38%	1280 Mo	65 Mo	423 s.
4	13765584	10381903	75,42%	931 Mo	59 Mo	352 s.
5	15168268	11502855	75,83%	1031 Mo	64 Mo	386 s.
6	18850820	14024905	74,40%	1254 Mo	69 Mo	482 s.
36	17715118	14092985	79,55%	1404 Mo	68 Mo	483 s.
37	17288466	7402082	42,82%	741 Mo	48 Mo	339 s.
38	16116394	13178457	81,77%	1101 Mo	63 Mo	420 s.
39	14241106	10537228	73,99%	880 Mo	57 Mo	348 s.
53	40876476	33780065	82,64%	3316 Mo	103 Mo	1165 s.

2.2.4 Run TemplateFilter on Mnase Samples

Finally, for each sample we perform TemplateFilter analysis.

WARNING TemplateFilter returns a list of nucleosomes. Each nucleosome is define by its center and its width. An odd width leads us to consider non- integer lower and upper bound.

WARNING TemplateFilter is not designed to deal with replicates. So we recommend to keep a maximum of nucleosomes and filter the aberrant ones afterwards using the benefits of having replicates. To do this, we set a low correlation threshold parameter (0.5) and a particularly high value of overlap (300%).

This step is performed by the following part of the *wf.py* script:

```
for sample in samples_mnase:
  per_mnase_sample_stats["sample_%s" % sample["id"]] = template_filter(sample,
    config["ALIGN_DIR"], config["LOG_DIR"], config["TF_BIN"],
    config["TF_TEMPLATES_FILE"], config["TF_CORR"], config["TF_MINW"],
    config["TF_MAXW"], config["TF_OL"])
```

id	strain	found nucs	nuc file size	process duration
1	BY	96214	68 Mo	1022 s.
2	BY	91694	65 Mo	1038 s.
3	BY	91205	65 Mo	1036 s.
4	RM	88076	62 Mo	984 s.
5	RM	90141	64 Mo	967 s.
6	RM	87517	62 Mo	980 s.

2.3 Inferring Nucleosome Position and Extracting Read Counts

The second part of the tutorial uses R (http://http://www.r-project.org). It consists of a set of R scripts that will be sourced in an R from a console launched at the root of your project. These scripts are:

- headers.R
- extract_maps.R
- translate_common_wp.R
- split_samples.R
- count_reads.R
- get_size_factors
- launch_deseq.R

2.3.1 The Script headers.R

The script headers.R is included in each other scripts. It is in charge of:

- launching libraries used in the scripts
- launching configuration (design, strain, marker...)
- computing and caching CURs (caching means storing the information in the computer's memory)

Note that you can customize the function "translate". This function allows you to use the alignments between genomes when performing various tasks. You may be using NucleoMiner2 to analyse data of a single strain, or of several strains.

- All the data corresponds to the same strain (e.g. treatment/control, or only few mutations): Then in step 1), the regions to use are entire chromosomes. Instep 2) simply use the default translate function which is neutral.

- The data come from two or more strains: In this case, edit a list of regions and customize the translate function which performs the correspondence between the different genomes. How we did it: a .c2c file is obtained with NucleoMiner 1.0 (refer to the Appendice "Generate .c2c Files"), then use it to produce the list of regions and customise "translate".

In your R console, run the following command line:

```
source("src/current/headers.R")
```

2.3.2 The Script extract_maps.R

This script is in charge of extracting Maps for well-positioned and fuzzy nucleosomes. First of all, this script computes intra and inter-strain nucleosome maps for each CUR. This step is executed in parallel on many cores using the BoT library. Next, it collects results and produces well-positioned, fuzzy and UNR maps.

The well-positioned map for BY is collected in the result directory and is called *BY_wp.tab*. It is composed of following columns:

- chr, the number of the chromosome
- lower_bound, the lower bound of the nucleosome
- upper_bound, the upper bound of the nucleosome
- cur_index, index of the CUR
- index_nuc, the index of the nucleosome in the CUR
- wp, 1 if it is a well positioned nucleosome, 0 otherwise
- nb_reads, the number of reads that support this nucleosome
- nb_nucs, the number of TemplateFilter nucleosome across replicates (= the number of replicates in which it is a well-positioned nucleosome)
- llr_1, for a well-positioned nucleosome, it is the LLR1 (log-likelihood ratio) between the first and the second TemplateFilter nucleosome on the chain.
- llr_2, for a well-positioned nucleosome, it is the LLR1 between the second and the third TemplateFilter nucleosome on the chain.
- wp_llr, for a well-positioned nucleosome, it is the LLR2 that compares consistency of the positioning over all TemplateFilter nucleosomes.
- wp_pval, for a well-positioned nucleosome, it is the p-value chi square test obtained with the LLR2 (*1-pchisq(2.LLR2, df=4)*)
- dyad_shift, for a well-positioned nucleosome, it is the shift between the two extreme TemplateFilter nucleosome dyad positions.

The fuzzy map for BY is collected in the result directory and is called *BY_fuzzy.tab*. It is composed of following columns:

- chr, the number of the chromosome
- lower_bound, the lower bound of the nucleosome
- upper_bound, the upper bound of the nucleosome
- cur_index, index of the CUR

The map of common well-positioned nucleosomes aligned between the BY and RM strains is collected in the result directory and is called *BY_RM_common_wp.tab*. It is composed of following columns:

- cur_index, the index of the CUR
- index_nuc_BY, the index of the BY nucleosome in the CUR
- index_nuc_RM, the index of the RM nucleosome in the CUR
- llr_score, , the LLR3 score that estimates conservation between the positions in BY and RM
- common_wp_pval, the p-value chi square test obtained from LLR3 (*1-pchisq(2.LLR3, df=2)*)
- diff, the dyads shift between the positions in the two strains

The common UNR map for BY and RM strains is collected in the result directory and is called *BY_RM_common_unr.tab*. It is composed of the following columns:

- cur_index, the index of the CUR
- index_nuc_BY, the index of the BY nucleosome in the CUR
- index_nuc_RM,the index of the RM nucleosome in the CUR

To execute this script, run the following command in your R console:

```
source("src/current/extract_maps.R")
```

2.3.3 The Script translate_common_wp.R

This script is used to translate common well-positioned nucleosome maps from a strain to another strain and stores it into a table.

For example, the file *results/2014-04/RM_wp_tr_2_BY.tab* contains RM well-positioned nucleosome translated into the BY genome coordinates. It is composed of following columns:

- strain_ref, the reference genome (in which positioned are defined)
- begin, the translated lower bound of the nucleosome
- end, the translated upper bound of the nucleosome
- chr, the number of chromosomes for the reference genome (in which positioned are defined)
- length, the length of the nucleosome (could be negative)
- cur_index, the index of the CUR
- index_nuc, the index of the nucleosome in the CUR

To execute this script, run the following command in your R console:

```
source("src/current/translate_common_wp.R")
```

2.3.4 The Script split_samples.R

For memory space usage reasons, we split and compress TemplateFilter input files according to their corresponding chromosome. for example, *sample_1_TF.tab* will be split into :

- sample_1_chr_1_splited_sample.tab.gz
- sample_1_chr_2_splited_sample.tab.gz
- ...
- sample_1_chr_17_splited_sample.tab.gz

To execute this script, run the following command in your R console:

```
source("src/current/split_samples.R")
```

2.3.5 The Script count_reads.R

To associate a number of observations (read) to each nucleosome we run the script *count_reads.R*. It produces the files *BY_RM_H3K14ac_wp_and_nbreads.tab*, *BY_RM_H3K14ac_unr_and_nbreads.tab* *BY_RM_Mnase_Seq_wp_and_nbreads.tab* and *BY_RM_Mnase_Seq_unr_and_nbreads.tab* for H3K14ac common well-positioned nucleosomes, H3K14ac UNRs, Mnase common well-positioned nucleosomes and Mnase UNRs respectively.

For example, the file *BY_RM_H3K14ac_unr_and_nbreads.tab* contains counted reads for well-positioned nucleosomes with the experimental condition ChIP H3K14ac. It is composed of the following columns:

- chr_BY, the number of the chromosome for BY
- lower_bound_BY, the lower bound of the nucleosome for BY
- upper_bound_BY, the upper bound of the nucleosome for BY
- index_nuc_BY, the index of the BY nucleosome in the CUR for BY
- chr_RM, the number of the chromosome for RM
- lower_bound_RM, the lower bound of the nucleosome for RM
- upper_bound_RM, the upper bound of the nucleosome for RM
- index_nuc_RM,the index of the RM nucleosome in the CUR for RM
- cur_index, index of the CUR
- BY_H3K14ac_36, the number of reads for the current nucleosome for the sample 36
- BY_H3K14ac_37, #reads for sample 37
- BY_H3K14ac_53, #reads for sample 53
- RM_H3K14ac_38, #reads for sample 38
- RM_H3K14ac_39, #reads for sample 39

To execute this script, run the following command in your R console:

```
source("src/current/count_reads.R")
```

2.3.6 The Script get_size_factors.R

This script uses the DESeq function *estimateSizeFactors* to compute the size factor of each sample. It corresponds to normalisation of read counts from sample to sample, as determined by DESeq. When a sample has n reads for a nucleosome or a UNR, the normalised count is n/f where f is the factor contained in this file. The script dumps computed size factors into the file *size_factors.tab*. This file has the form:

sample_id	wp	unr	wpunr
1	0.87396	0.88097	0.87584
2	1.07890	1.07440	1.07760
3	1.06400	1.05890	1.06250
4	0.85782	0.87948	0.86305
5	0.97577	0.96590	0.97307
6	1.19630	1.18120	1.19190
36	0.93318	0.92762	0.93166
37	0.48315	0.48453	0.48350
38	1.11240	1.11210	1.11230
39	0.89897	0.89917	0.89903
53	2.22650	2.22700	2.22660

sample_id are given in file samples.csv

If you don't know which column to use, we recommend using wpunr.

If you want the very detailed factors produced by DESeq, here are the information:

- unr: factor computed from data of UNR regions. These regions are defined for every pairs of aligned genomes (e.g. BY_RM)
- wp: same, but for well-positioned nucleosomes.
- wpunr: both types of regions.

To execute this script, run the following command in your R console:

```
source("src/current/get_size_factors.R")
```

2.3.7 The Script launch_deseq.R

Finally, the script *launch_deseq.R* perform statistical analysis on each nucleosome using *DESeq*. It produces files:

- results/current/BY_RM_H3K14ac_wp_snep.tab
- results/current/BY_RM_H3K14ac_unr_snep.tab
- results/current/BY_RM_H3K14ac_wpunr_snep.tab
- results/current/BY_RM_H3K14ac_wp_mnase.tab
- results/current/BY_RM_H3K14ac_unr_mnase.tab
- results/current/BY_RM_H3K14ac_wpunr_mnase.tab

These files are organised with the following columns (see file *BY_RM_H3K14ac_wp_snep.tab* for an example):

- chr_BY, the number of the chromosome for BY
- lower_bound_BY, the lower bound of the nucleosome for BY
- upper_bound_BY, the upper bound of the nucleosome for BY
- index_nuc_BY, the index of the BY nucleosome in the CUR for BY
- chr_RM, the number of the chromosome for RM
- lower_bound_RM, the lower bound of the nucleosome for RM
- upper_bound_RM, the upper bound of the nucleosome for RM
- index_nuc_RM,the index of the RM nucleosome in the CUR for RM
- cur_index, index of the CUR

- form
- BY_Mnase_Seq_1, the number of reads for the current nucleosome for the sample 1

Next columns concern indicators for each sample:

- BY_Mnase_Seq_2, #reads for sample 2
- BY_Mnase_Seq_3, #reads for sample 3
- RM_Mnase_Seq_4, #reads for sample 4
- RM_Mnase_Seq_5, #reads for sample 5
- RM_Mnase_Seq_6, #reads for sample 6
- BY_H3K14ac_36, #reads for sample 36
- BY_H3K14ac_37, #reads for sample 37
- BY_H3K14ac_53, #reads for sample 53
- RM_H3K14ac_38, #reads for sample 38
- RM_H3K14ac_39, #reads for sample 39

The 5 last columns concern DESeq analysis:

- manip[a_manip] strain[a_strain] manip[a_strain]:strain[a_strain], the manip (marker) effect, the strain effect and the snep effect. These are the coefficients of the fitted generalized linear model.
- pvalsGLM, the pvalue resulting of the comparison of the GLM model considering or not the interaction term marker:strain. This is the statsitcial significance of the interaction term and therefore the statistical significance of the SNEP.
- snep_index, a boolean set to TRUE if the pvalueGLM value is under the threshold computed with FDR function with a rate set to 0.0001.

To execute this script, run the following command

To execute this script, run the following command in your R console:

```
source("src/current/launch_deseq.R")
```

2.4 Results: Number of SNEPs

Here are the number of computed SNEPs for each forms.

form	strains	#nucs	H3K14ac
wp	BY-RM	30464	3549
unr	BY-RM	9497	1559
wpunr	BY-RM	39961	5240

2.5 APPENDICE: Generate .c2c Files

The *.c2c* files is a simple table that describes how the genome sequence can be aligned. We generate it using NucleoMiner 1.0.

To install NucleoMiner 1.0 on your UNIX/LINUX computer you need first to install the Genetic Data analysis Library (GDL), which is a dynamic library of useful C functions derived from the GNU Scientific Library.

2.5.1 Installing the GDL library

Get the gdl-1.0.tar.gz archive on your computer (in the directory deps of your working directory). Copy it in a dedicated directory. Go into this directory using the cd command, and then unfold the archive by typing:

tar -xvzf gdl-1.0.tar.gz

This creates a directory called gdl-1.0. You now need to go into this directory and compile the library, by typing:

```
cd gdl-1.0
./configure
make
```

Now you need to install the library on your system. This needs root priviledges:

```
sudo make install
```

2.5.2 Installing NucleoMiner

Get the nucleominer-1.0.tar.gz archive on your computer. Copy it in a dedicated directory. Go into this directory using the cd command, and then unfold the archive by typing:

tar -xvzf nucleominer-1.0.tar.gz This creates a directory called nucleominer-1.0. You now need to go into this directory and compile the library, by typing:

```
cd nucleominer-1.0
./configure
make
```

You can then use the binaries dircetly from this folder (best then is to add the path to this folder in your PATH environment variable). If you want to install nucleominer at the system's level (useful if mutiple users will need it) then type, with root priviledges:

```
sudo make install
```

2.5.3 Generate .c2c Files

To generate .c2c files you need to type the following command in a terminal:

```
mkdir dir_4_c2c
NMgxcomp Data/BY_S288c/Sequence/Genome.fasta \
         Data/RM_11-1a/Sequence/Genome.fasta \
         dir_4_c2c/BY_RM 2>dir_4_c2c/BY_RM.log
```

After execution, the directory dir_4_c2c will hold the .c2c files.

CHAPTER

THREE

REFERENCES

3.1 Python Reference

configurator.**CSV_SAMPLE_FILE = None**
Path to cvs file that contains sample information.

configurator.**BOWTIE_BUILD_BIN = None**
Path for bowtie2 build bin.

configurator.**BOWTIE2_BIN = None**
Path for bowtie2 bin.

configurator.**SAMTOOLS_BIN = None**
Path for samtools bin.

configurator.**BEDTOOLS_BIN = None**
Path for bedtools bin.

configurator.**TF_BIN = None**
Path for TemplateFilter bin.

configurator.**TF_TEMPLATES_FILE = None**
Path for TemplateFilter templates file.

configurator.**ILLUMINA_OUTPUTFILE_PREFIX = None**
Prefix for Illumina fastq output files.

configurator.**INDEX_DIR = None**
Path for index dir.

configurator.**ALIGN_DIR = None**
Path for align dir.

configurator.**LOG_DIR = None**
Path for log dir

configurator.**CACHE_DIR = None**
Path for cache dir.

configurator.**RESULTS_DIR = None**
Path for results dir

configurator.**FASTA_REFERENCE_GENOME_FILES = None**
Dictionary where each fasta reference genomes is indexed by reference strain that it corresponds.

configurator.**AREA_BLACK_LIST = None**
Dictionary where keys are strain and values are black listed of geneome region.

configurator.**FASTA_INDEXES = None**
Dictionary of strain that indexes dictionaries where keys are chromosome reference from Fastq file and value are its correspondance for Templatefilter.

configurator.**C2C_FILES = None**
Dictionary where each strain combination indexes genome aligment.

configurator.**READ_LENGTH = None**
Length of Illumina reads.

configurator.**MAPQ_THRES = None**
Aligment quality thresold.

configurator.**TF_CORR = None**
TemplateFilter Template correlation threshold.

configurator.**TF_MINW = None**
TemplateFilter minimum width of a nucleosome.

configurator.**TF_MAXW = None**
TemplateFilter maximum width of a nucleosome.

configurator.**TF_OL = None**
TemplateFilter maximum allowed overlap for two nucleosomes.

wf.**json_conf_file = 'src/current/nucleominer_config.json'**
Path to the json configuration file.

wf.**samples = []**
List of samples where a sample is identified by an id (key: *id*) and a strain name (key *strain*).

wf.**samples_mnase = []**
List of Mnase samples.

wf.**strains = []**
List of reference strains.

libcoverage.**create_bowtie_index**(*strain*, *strain_fasta_ref*, *index_dir*, *bowtie_build_bin*)
Creates bowtie index for a strain *strain*.

Parameters

- **strain** – the strain reference.
- **strain_fasta_ref** – fasta reference genome.
- **index_dir** – directories where to put bowtie index.
- **bowtie_build_bin** – bowtie2 build binary.

libcoverage.**align_reads**(*sample*, *align_dir*, *log_dir*, *index_dir*, *illumina_outputfile_prefix*, *bowtie2_bin*, *samtools_bin*, *bedtools_bin*)
Aligns reads to reference genomes. It produces .sam files, that are converted to .bam, that are then converted to .bed.

Parameters

- **sample** – a dict that describe a sample.
- **align_dir** – directory where aligned reads will be stored.
- **log_dir** – directory where logs will be stored.
- **illumina_outputfile_prefix** – prefix of Illumina sequencer fastq.gz output files.
- **bowtie2_bin** – bowtie2 binary.

- **samtools_bin** – samtools binary.
- **bedtools_bin** – bedtools binary.
- **index_dir** – bowtie index directory.

libcoverage.**split_fr_4_TF**(*sample*, *align_dir*, *fasta_indexes*, *area_black_list*, *read_length*, *mapq_thres*)

Create TemplateFilter input files form bed files. This function appends in two times. First, it collects reads from bed files and feeds a datastructure

Parameters

- **sample** – a dict that describe a sample.
- **align_dir** – directory where aligned reads will be stored.
- **fasta_index** – the chr reference from the illumina output file.
- **area_black_list** – the description of genome that will be omit.
- **read_length** – Length of Illumina reads.
- **mapq_thres** – mapping quality criterion threshold, see MAPQ in BED/BAM file format.

libcoverage.**template_filter**(*sample*, *align_dir*, *log_dir*, *tf_bin*, *tf_templates_file*, *corr*, *minw*, *maxw*, *ol*)

Run TemplateFilter on a specific sample. It produces .tab file.

Parameters

- **sample** – a dict that describe a sample.
- **align_dir** – directory where aligned reads will be stored.
- **log_dir** – directory where logs will be stored.
- **tf_bin** – path to the TemplateFilter binary.
- **tf_templates_file** – path to the TemplateFilter templates file.
- **corr** – correlation threshold transmits to TemplateFilter.
- **minw** – minimum width of a nuc, transmits to TemplateFilter.
- **maxw** – maximum width of a nuc, transmits to TemplateFilter.
- **ol** – maximum overlaps for 2 nuc, transmits to TemplateFilter.

3.2 R Reference

3.2.1 Arabic to Roman pair list.

Description

Utility to convert Arabic numbers to Roman numbers

Usage

```
ARAB2ROM()
```

Author(s)

Florent Chuffart

R: False Discovery Rate

3.2.2 False Discovery Rate

Description

From a vector x of independent p-values, extract the cutoff corresponding to the specified FDR. See Benjamini & Hochberg 1995 paper

Usage

```
FDR(x, FDR)
```

Arguments

x

A vector x of independent p-values.

FDR

The specified FDR.

Value

Return the the corresponding cutoff.

Author(s)

Gael Yvert, Florent Chuffart

Examples

```
print("example")
```

R: Roman to Arabic pair list.

3.2.3 Roman to Arabic pair list.

Description

Utility to convert Roman numbers into Arabic numbers

Usage

```
ROM2ARAB()
```

Author(s)

Florent Chuffart

R: Aggregate replicated sample's nucleosomes.

3.2.4 Aggregate replicated sample's nucleosomes.

Description

This function aggregates nucleosomes from replicated samples. It uses TemplateFilter ouput of each sample as replicate. Each sample owns a set of nucleosomes computed using TemplateFilter and ordered by the position of their center (dyad). A chain of nucleosomes is builts across all replicates. Adjacent nucleosomes of the chain are compared two by two. Comparison is based on a log likelihood ratio (LLR1). depending on the LLR1 value nucleosomes are merged (low LLR) or separated (high LLR). Finally the function returns a list of clusters and all computed llr_scores. Each cluster ows an attribute wp for "well positioned". This attribute is set to TRUE if the cluster is composed of exactly one nucleosome of each sample.

Usage

```
aggregate_intra_strain_nucs(samples, llr_thres = 20, coord_max = 2e+07)
```

Arguments

`samples`

A list of samples. Each sample is a list like *sample = list(id=..., marker=..., strain=..., roi=..., inputs=..., outputs=...)* with *roi = list(name=..., begin=..., end=..., chr=..., genome=...)*.

`llr_thres`

Log likelihood ratio threshold to decide between merging and separating

`coord_max`

A too big value to be a coord for a nucleosome lower bound.

Value

Returns a list of clusterized nucleosomes, and all computed llr scores.

Author(s)

Florent Chuffart

Examples

```
# Dealing with a region of interest
roi =list(name="example", begin=1000,  end=1300, chr="1", genome=rep("A",301))
samples = list()
for (i in 1:3) {
    # Create TF output
    tf_nuc = list("chr"=paste("chr", roi$chr, sep=""), "center"=(roi$end + roi$begin)/2, "width"= 150
    outputs = dfadd(NULL,tf_nuc)
    outputs = filter_tf_outputs(outputs, roi$chr, roi$begin, roi$end)
    # Generate corresponding reads
    nb_reads = round(runif(1,170,230))
    reads = round(rnorm(nb_reads, tf_nuc$center,20))
    u_reads = sort(unique(reads))
    strands = sample(c(rep("R",ceiling(length(u_reads)/2)),rep("F",floor(length(u_reads)/2))))
    counts = apply(t(u_reads), 2, function(r) { sum(reads == r)})
    shifts = apply(t(strands), 2, function(s) { if (s == "F") return(-tf_nuc$width/2) else return(tf_
    u_reads = u_reads + shifts
    inputs = data.frame(list("V1" = rep(roi$chr, length(u_reads)),
                             "V2" = u_reads,
                                                    "V3" = strands,
                                                    "V4" = counts), stringsAsFactors=FALSE)
    samples[[length(samples) + 1]] = list(id=1, marker="Mnase_Seq", strain="strain_ex", total_reads =
}
print(aggregate_intra_strain_nucs(samples))
```

R: Aligns nucleosomes between 2 strains.

3.2.5 Aligns nucleosomes between 2 strains.

Description

This function aligns nucleosomes between two strains for a given genome region.

Usage

```
align_inter_strain_nucs(replicates, wp_nucs_strain_ref1 = NULL,
    wp_nucs_strain_ref2 = NULL, corr_thres = 0.5, llr_thres = 100,
    config = NULL, ...)
```

Arguments

`replicates`

Set of replicates, ideally 3 per strain.

`wp_nucs_strain_ref1`

List of aggregates nucleosome for strain 1. If it's NULL this list will be computed.

`wp_nucs_strain_ref2`

List of aggregates nucleosome for strain 2. If it's NULL this list will be computed.

`corr_thres`

Correlation threshold.

`llr_thres`

Log likelihood ratio threshold to decide between merging and separating

`config`

GLOBAL config variable

`...`

A list of parameters that will be passed to *aggregate_intra_strain_nucs* if needed.

Value

Returns a list of clusterized nucleosomes, and all computed llr scores.

Author(s)

Florent Chuffart

Examples

```
    # Define new translate_cur function...
    translate_cur = function(roi, strain2, big_cur=NULL, config=NULL) {
      return(roi)
    }
    # Binding it by uncomment follwing lines.
    unlockBinding("translate_cur", as.environment("package:nucleominer"))
    unlockBinding("translate_cur", getNamespace("nucleominer"))
    assign("translate_cur", translate_cur, "package:nucleominer")
    assign("translate_cur", translate_cur, getNamespace("nucleominer"))
    lockBinding("translate_cur", getNamespace("nucleominer"))
    lockBinding("translate_cur", as.environment("package:nucleominer"))

# Dealing with a region of interest
roi =list(name="example", begin=1000,  end=1300, chr="1", genome=rep("A",301), strain_ref1 = "STRAINE
roi2 = translate_cur(roi, roi$strain_ref1)
replicates = list()
for (j in 1:2) {
    samples = list()
    for (i in 1:3) {
        # Create TF output
        tf_nuc = list("chr"=paste("chr", roi$chr, sep=""), "center"=(roi$end + roi$begin)/2, "width"=
        outputs = dfadd(NULL,tf_nuc)
        outputs = filter_tf_outputs(outputs, roi$chr, roi$begin, roi$end)
        # Generate corresponding reads
        nb_reads = round(runif(1,170,230))
        reads = round(rnorm(nb_reads, tf_nuc$center,20))
        u_reads = sort(unique(reads))
        strands = sample(c(rep("R",ceiling(length(u_reads)/2)),rep("F",floor(length(u_reads)/2))))
        counts = apply(t(u_reads), 2, function(r) { sum(reads == r)})
        shifts = apply(t(strands), 2, function(s) { if (s == "F") return(-tf_nuc$width/2) else return
        u_reads = u_reads + shifts
        inputs = data.frame(list("V1" = rep(roi$chr, length(u_reads)),
                                 "V2" = u_reads,
                                                        "V3" = strands,
                                                        "V4" = counts), stringsAsFactors=FALSE)
```

```
        samples[[length(samples) + 1]] = list(id=1, marker="Mnase_Seq", strain=paste("strain_ex",j,se
    }
    replicates[[length(replicates) + 1]] = samples
}
print(align_inter_strain_nucs(replicates))
```

R: Launch DESeq methods.

3.2.6 Launch DESeq methods.

Description

This function is based on DESeq example. It normalizes data, fit data to GLM model with and without interaction term and compares the two models.

Usage

```
analyse_design(snep_design, reads)
```

Arguments

`snep_design`

The design to consider.

`reads`

The data to consider.

Author(s)

Florent Chuffart

R: Stage replicates data

3.2.7 Stage replicates data

Description

This function loads in memory the data corresponding to the given experiments.

Usage

```
build_replicates(expe, roi, only_fetch = FALSE, get_genome = FALSE,
    all_samples, config = NULL)
```

Arguments

`expe`

a list of vectors corresponding to replicates.

`roi`

the region that we are interested in.

`only_fetch`

filter or not inputs.

`get_genome`

Load or not corresponding genome.

`all_samples`

Global list of samples.

`config`

GLOBAL config variable.

Author(s)

Florent Chuffart

Examples

```
# library(rjson)
# library(nucleominer)
#
# # Read config file
# json_conf_file = "nucleominer_config.json"
# config = fromJSON(paste(readLines(json_conf_file), collapse=""))
# # Read sample file
# all_samples = get_content(config$CSV_SAMPLE_FILE, "cvs", sep=";", head=TRUE, stringsAsFactors=FALSE
# # here are the sample ids in a list
# expes = list(c(1))
# # here is the region that we wnt to see the coverage
# cur = list(chr="8", begin=472000, end=474000, strain_ref="BY")
# # it displays the corverage
# replicates = build_replicates(expes, cur, all_samples=all_samples, config=config)
# out = watch_samples(replicates, config$READ_LENGTH,
#       plot_coverage = TRUE,
#       plot_squared_reads = FALSE,
#       plot_ref_genome = FALSE,
#       plot_arrow_raw_reads = FALSE,
#       plot_arrow_nuc_reads = FALSE,
#       plot_gaussian_reads = FALSE,
#       plot_gaussian_unified_reads = FALSE,
#       plot_ellipse_nucs = FALSE,
#       plot_wp_nucs = FALSE,
#       plot_wp_nuc_model = FALSE,
#       plot_common_nucs = FALSE,
#       height = 50)
```

R: Extract a sub part of the corresponding c2c file

3.2.8 Extract a sub part of the corresponding c2c file

Description

This fonction allows to access to a specific part of the c2c file.

Usage

```
c2c_extraction(strain1, strain2, chr = NULL, lower_bound = NULL,
    upper_bound = NULL, config = NULL)
```

Arguments

`strain1`

the key strain

`strain2`

the target strain

`chr`

if defined, the c2c will be filtered according to the chromosome value

`lower_bound`

if defined, the c2c will be filtered for part of the genome upper than lower_bound

`upper_bound`

if defined, the c2c will be filtered for part of the genome lower than upper_bound

`config`

GLOBAL config variable

Author(s)

Florent Chuffart

R: reformat an “apply manipulated” list of regions

3.2.9 reformat an “apply manipulated” list of regions

Description

Utils to reformat an “apply manipulated” list of regions

Usage

```
collapse_regions(regions)
```

Arguments

regions	

Author(s)

Florent Chuffart

R: Compute Common Uninterrupted Regions (CUR)

3.2.10 Compute Common Uninterrupted Regions (CUR)

Description

CURs are regions that can be aligned between the genomes

Usage

```
compute_inter_all_strain_curs(diff_allowed = 30, min_cur_width = 4000,
    config = NULL)
```

Arguments

`diff_allowed`

the maximum indel width allowe din a CUR

`min_cur_width`

The minimum width of a CUR

`config`

GLOBAL config variable

Author(s)

Florent Chuffart

R: Crop bound of regions according to region of interest bound

3.2.11 Crop bound of regions according to region of interest bound

Description

The fucntion is no more necessary since we remove “big_cur” bug in translate_cur function.

Usage

```
crop_fuzzy(tmp_fuzzy_nucs, roi, strain, config = NULL)
```

Arguments

`tmp_fuzzy_nucs`

the regiuons to be croped.

`roi`

The region of interest.

`strain`

The strain to consider.

`config`

GLOBAL config variable

Author(s)

Florent Chuffart

R: Adding list to a dataframe.

3.2.12 Adding list to a dataframe.

Description

Add a list *l* to a dataframe *df*. Create it if *df* is *NULL*. Return the dataframe *df*.

Usage

```
dfadd(df, l)
```

Arguments

`df`

A dataframe

`l`

A list

Value

Return the dataframe *df*.

Author(s)

Florent Chuffart

Examples

```
## Here dataframe is NULL
print(df)
df = NULL

# Initialize df
df = dfadd(df, list(key1 = "value1", key2 = "value2"))
print(df)

# Adding elements to df
df = dfadd(df, list(key1 = "value1'", key2 = "value2'"))
print(df)
```

R: Prefetch data

3.2.13 Prefetch data

Description

Fetch and filter inputs and outpouts per region of interest. Organize it per replicates.

Usage

```
fetch_mnase_replicates(strain, roi, all_samples, config = NULL,
    only_fetch = FALSE, get_genome = FALSE, get_ouputs = TRUE)
```

Arguments

`strain`

The strain we want mnase replicatesList of replicates. Each replicates is a vector of sample ids.

`roi`

Region of interest.

`all_samples`

Global list of samples.

`config`

GLOBAL config variable

`only_fetch`

If TRUE, only fetch and not filtering. It is used tio load sample files into memory before forking.

`get_genome`

If TRUE, load corresponding genome sequence.

`get_ouputs`

If TRUE, get also ouput corresponding TF output files.

Author(s)

Florent Chuffart

R: Filter TemplateFilter inputs

3.2.14 Filter TemplateFilter inputs

Description

This function filters TemplateFilter inputs according genome area observed properties. It takes into account reads that are at the frontier of this area and the strand of these reads.

Usage

```
filter_tf_inputs(inputs, chr, x_min, x_max, nuc_width = 160,
    only_f = FALSE, only_r = FALSE, filter_for_coverage = FALSE)
```

Arguments

`inputs`

TF inputs to be filtered.

`chr`

Chromosome observed, here chr is an integer.

`x_min`

Coordinate of the first bp observed.

`x_max`

Coordinate of the last bp observed.

`nuc_width`

Nucleosome width.

`only_f`

Filter only F reads.

`only_r`

Filter only R reads.

`filter_for_coverage`

Does it filter for plot coverage?

Value

Returns filtred inputs.

Author(s)

Florent Chuffart

R: Filter TemplateFilter outputs

3.2.15 Filter TemplateFilter outputs

Description

This function filters TemplateFilter outputs according, not only genome area observerved properties, but also correlation and overlapping threshold.

Usage

```
filter_tf_outputs(tf_outputs, chr, x_min, x_max, nuc_width = 160,
    ol_bp = 59, corr_thres = 0.5)
```

Arguments

`tf_outputs`

TemplateFilter outputs.

`chr`

Chromosome observed, here chr is an integer.

`x_min`

Coordinate of the first bp observed.

`x_max`

Coordinate of the last bp observed.

`nuc_width`

Nucleosome width.

`ol_bp`

Overlap Threshold.

`corr_thres`

Correlation threshold.

Value

Returns filtered TemplateFilter Outputs

Author(s)

Florent Chuffart

R: to flat aggregate_intra_strain_nucs function output

3.2.16 to flat aggregate_intra_strain_nucs function output

Description

This function builds a dataframe of all clusters obtain from aggregate_intra_strain_nucs function.

Usage

```
flat_aggregated_intra_strain_nucs(partial_strain_maps, cur_index)
```

Arguments

`partial_strain_maps`

the output of aggregate_intra_strain_nucs function

`cur_index`

the index of the roi involved

Value

Returns a dataframe of all clusters obtain from aggregate_intra_strain_nucs function.

Author(s)

Florent Chuffart

R: flat reads

3.2.17 flat reads

Description

Extract reads coordinates from TempleteFilter input sequence

Usage

```
flat_reads(reads, nuc_width)
```

Arguments

`reads`

TemplateFilter input reads

`nuc_width`

Width used to shift F and R reads.

Value

Returns a list of F reads, R reads and joint/shifted F and R reads.

Author(s)

Florent Chuffart

R: Retrieve Reads

3.2.18 Retrieve Reads

Description

Retrieve reads for a given marker, combi, form.

Usage

```
get_all_reads(marker, combi, form = "wp", config = NULL)
```

Arguments

`marker`

The marker to considere.

`combi`

The starin combination to considere.

`form`

The nuc form to considere.

`config`

GLOBAL config variable

Author(s)

Florent Chuffart

R: get comp strand

3.2.19 get comp strand

Description

Compute the complementatry strand.

Usage

```
get_comp_strand(strand)
```

Arguments

`strand`

The original strand.

Value

Returns the complementatry strand.

Author(s)

Florent Chuffart

R: Build the design for DESeq

3.2.20 Build the design for DESeq

Description

This function build the design according sample properties.

Usage

```
get_design(marker, combi, all_samples)
```

Arguments

`marker`

The marker to considere.

`combi`

The starin combination to considere.

`all_samples`

Global list of samples.

Author(s)

Florent Chuffart

R: Compute the fuzzy list for a given strain.

3.2.21 Compute the fuzzy list for a given strain.

Description

This function grabs the nucleosomes detxted by template_filter that have been rejected bt aggregate_intra_strain_nucs as well positions.

Usage

```
get_intra_strain_fuzzy(wp_map, roi, strain, config = NULL)
```

Arguments

`wp_map`

Well positionned nucleosomes map.

`roi`

The region of interest.

`strain`

The strain we want to extracvt the fuzzy map.

`config`

GLOBAL config variable.

Author(s)

Florent Chuffart

R: Compute the list of SNEPs for a given set of marker, strain...

3.2.22 Compute the list of SNEPs for a given set of marker, strain combination and nuc form.

Description

This function uses

Usage

```
get_sneps(marker, combi, form, all_samples, FDR = 1e-04, config = NULL)
```

Arguments

`marker`

The marker involved.

`combi`

The strain combination involved.

`form`

the nuc form involved.

`all_samples`

Global list of samples.

`FDR`

`config`

GLOBAL config variable

Author(s)

Florent Chuffart

Examples

```
marker = "H3K4me1"
combi = c("BY", "YJM")
form = "wpunr" # "wp" | "unr" | "wpunr"
# foo = get_sneps(marker, combi, form)
# foo = get_sneps("H4K12ac", c("BY", "RM"), "wp")
```

R: Compute the unaligned nucleosomal regions (UNRs).

3.2.23 Compute the unaligned nucleosomal regions (UNRs).

Description

This function aggregate non common wp nucs for each strain and substract common wp nucs. It does not take care about the size of the resulting UNR. It will be take into account in the count read part og the pipeline.

Usage

```
get_unrs(combi, roi, cur_index, wp_maps, fuzzy_maps, common_nuc_results,
    config = NULL)
```

Arguments

`combi`

The strain combination to consider.

`roi`

The region of interest.

`cur_index`

The region of interest index.

`wp_maps`

Well positionned nucleosomes maps.

`fuzzy_maps`

Fuzzy nucleosomes maps.

`common_nuc_results`

Common wp nuc maps

`config`

GLOBAL config variable

Author(s)

Florent Chuffart

R: Returns the intersection of 2 list on regions.

3.2.24 Returns the intersection of 2 list on regions.

Description

This function...

Usage

```
intersect_region(region1, region2)
```

Arguments

`region1`

Original regions.

`region2`

Regions to intersect.

Author(s)

Florent Chuffart

R: Likelihood ratio

3.2.25 Likelihood ratio

Description

Compute the log likelihood ratio of two or more set of value.

Usage

```
llr_score_nvecs(xs)
```

Arguments

`xs`

list of vectors.

Value

Returns the log likelihood ratio.

Author(s)

Florent Chuffart

Examples

```
# LLR score for 2 set of values
mean1=5; sd1=2; card2 = 250
mean2=6; sd2=3; card1 = 200
x1 = rnorm(card1, mean1, sd1)
x2 = rnorm(card2, mean2, sd2)
min = floor(min(c(x1,x2)))
max = ceiling(max(c(x1,x2)))
hist(c(x1,x2), xlim=c(min, max), breaks=min:max)
lines(min:max,dnorm(min:max,mean1,sd1)*card1,col=2)
lines(min:max,dnorm(min:max,mean2,sd2)*card2,col=3)
lines(min:max,dnorm(min:max,mean(c(x1,x2)),sd(c(x1,x2)))*card2,col=4)
llr_score_nvecs(list(x1,x2))
```

R: nm

3.2.26 nm

Description

It provides a set of useful functions allowing to perform quantitative analysis of nucleosomal epigenome.

Details

Package:	nucleominer
Maintainer:	Florent Chuffart <florent.chuffart@ens-lyon.fr>
Author:	Florent Chuffart
Version:	2.3.45
License:	CeCILL
Title:	nm
Depends:	seqinr, plotrix, DESeq, cachecache

Author(s)

Florent Chuffart

R: Plot the distribution of reads.

3.2.27 Plot the distribution of reads.

Description

This fuxntion use the DESeq nomalization feature to compare qualitatively the distribution.

Usage

```
plot_dist_samples(strain, marker, res, all_samples, NEWPLOT = TRUE)
```

Arguments

`strain`

The strain to considere.

`marker`

The marker to considere.

`res`

Data

`all_samples`

Global list of samples.

`NEWPLOT`

If FALSE the curve will be add to the current plot.

Author(s)

Florent Chuffart

R: sign from strand

3.2.28 sign from strand

Description

Get the sign of strand

Usage

```
sign_from_strand(strands)
```

Arguments

strands	

Value

If strand in forward then returns 1 else returns -1

Author(s)

Florent Chuffart

R: Substract to a list of regions an other list of regions that...

3.2.29 Substract to a list of regions an other list of regions that intersect it.

Description

This fucntion embed a recursive part. It occurs when a substracted region split an original region on two.

Usage

```
substract_region(region1, region2)
```

Arguments

`region1`

Original regions.

`region2`

Regions to substract.

Author(s)

Florent Chuffart

R: Switch a pairlist

3.2.30 Switch a pairlist

Description

Take a pairlist key:value and return the switched pairlist value:key.

Usage

```
switch_pairlist(l)
```

Arguments

`l`

The pairlist to switch.

Value

The switched pairlist.

Author(s)

Florent Chuffart

Examples

```
l = list(key1 = "value1", key2 = "value2")
print(switch_pairlist(l))
```

R: Translate coords of a genome region.

3.2.31 Translate coords of a genome region.

Description

This function is used in the examples, usualy you have to define your own translation function and overwrite this one using *unlockBinding* features. Please, refer to the example.

Usage

```
translate_cur(roi, strain2, config = NULL, big_cur = NULL)
```

Arguments

`roi`

Original genome region of interest.

`strain2`

The strain in wich you want the genome region of interest.

`config`

GLOBAL config variable

`big_cur`

A largest region than roi use to filter c2c if it is needed.

Author(s)

Florent Chuffart

Examples

```
# Define new translate_cur function...
translate_cur = function(roi, strain2, config) {
    strain1 = roi$strain_ref
    if (strain1 == strain2) {
        return(roi)
    } else {
      stop("Here is my new translate_cur function...")
    }
}
# Binding it by uncomment follwing lines.
# unlockBinding("translate_cur", as.environment("package:nm"))
# unlockBinding("translate_cur", getNamespace("nm"))
# assign("translate_cur", translate_cur, "package:nm")
# assign("translate_cur", translate_cur, getNamespace("nm"))
# lockBinding("translate_cur", getNamespace("nm"))
# lockBinding("translate_cur", as.environment("package:nm"))
```

R: Translate a list of regions from a strain ref to another.

3.2.32 Translate a list of regions from a strain ref to another.

Description

This function is an elaborated call to translate_cur.

Usage

```
translate_regions(regions, combi, cur_index, config = NULL, roi)
```

Arguments

`regions`

Regions to be translated.

`combi`

Combination of strains.

`cur_index`

The region of interest index.

`config`

GLOBAL config variable

`roi`

The region of interest.

Author(s)

Florent Chuffart

R: Aggregate regions that intersect themselves.

3.2.33 Aggregate regions that intersect themselves.

Description

This function is based on sort of lower bounds to detect regions that intersect. We compare lower bound and upper bound of the porevious item. This function embed a while loop and break break regions list become stable.

Usage

```
union_regions(regions)
```

Arguments

`regions`

The Regions to be aggregated

Author(s)

Florent Chuffart

R: Watching analysis of samples

3.2.34 Watching analysis of samples

Description

This function allows to view analysis for a particuler region of the genome.

Usage

```
watch_samples(replicates, read_length, plot_ref_genome = TRUE,
    plot_arrow_raw_reads = TRUE, plot_arrow_nuc_reads = TRUE,
    plot_squared_reads = TRUE, plot_coverage = FALSE, plot_gaussian_reads = TRUE,
    plot_gaussian_unified_reads = TRUE, plot_ellipse_nucs = TRUE,
    change_col = TRUE, plot_wp_nucs = TRUE, plot_fuzzy_nucs = TRUE,
    plot_wp_nuc_model = TRUE, plot_common_nucs = FALSE, plot_common_unrs = FALSE,
    plot_wp_nucs_4_nonmnase = FALSE, plot_chain = FALSE, plot_sample_id = FALSE,
    aggregated_intra_strain_nucs = NULL, aligned_inter_strain_nucs = NULL,
    height = 10, main = NULL, xlab = NULL, ylab = "#reads (per million reads)",
    config = NULL)
```

Arguments

`replicates`

replicates under the form...

`read_length`

length of the reads

`plot_ref_genome`

Plot (or not) reference genome.

`plot_arrow_raw_reads`

Plot (or not) arrows for raw reads.

`plot_arrow_nuc_reads`

Plot (or not) arrows for reads aasiocied to a nucleosome.

`plot_squared_reads`

Plot (or not) reads in the square fashion.

`plot_coverage`

Plot (or not) reads in the covergae fashion. fashion.

`plot_gaussian_reads`

Plot (or not) gaussian model of a F anf R reads.

`plot_gaussian_unified_reads`

Plot (or not) gaussian model of a nuc.

`plot_ellipse_nucs`

Plot (or not) ellipse for a nuc.

`change_col`

Change the color of each nucleosome.

`plot_wp_nucs`

Plot (or not) cluster of nucs

`plot_fuzzy_nucs`

Plot (or not) cluster of fuzzy

`plot_wp_nuc_model`

Plot (or not) gaussian model for a cluster of nucs

`plot_common_nucs`

Plot (or not) aligned reads.

`plot_common_unrs`

Plot (or not) unaligned nucleosomal refgions (UNRs).

`plot_wp_nucs_4_nonmnase`

Plot (or not) clusters for non inputs samples.

`plot_chain`

Plot (or not) clusterised nuceosomes between mnase samples.

`plot_sample_id`

Plot (or not) the sample id for each sample.

`aggregated_intra_strain_nucs`

list of aggregated intra strain nucs. If NULL, it will be computed.

`aligned_inter_strain_nucs`

list of aligned inter strain nucs. If NULL, it will be computed.

`height`

Number of reads in per million read for each sample, graphical parametre for the y axis.

`main`

main title of the produced plot

`xlab`

xlab of the produced plot

`ylab`

ylab of the produced plot

`config`

GLOBAL config variable

Author(s)

Florent Chuffart