# Package 'ALPS'

October 16, 2020

**Title** AnaLysis routines for ePigenomicS data

**Version** 1.2.0

**Author** Venu Thatikonda, Natalie Jäger

**Maintainer** Venu Thatikonda <thatikonda92@gmail.com>

**Description**

The package provides analysis and publication quality visualization routines for genome-wide epigenomics data such as histone modification or transcription factor ChIP-seq, ATAC-seq, DNase-seq etc.
The functions in the package can be used with any type of data that can be represented with bigwig files
at any resolution. The goal of the ALPS is to provide analysis tools for most downstream analysis without
leaving the R environment and most tools in the package require a minimal input that can be prepared with
basic R, unix or excel skills.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** false

**RoxygenNote** 6.1.1

**Depends** R (>= 3.6)

**Imports** assertthat, BiocParallel, ChIPseeker, corrplot, data.table, dplyr, GenomicRanges, GGally, genefilter, gghalves, ggplot2, ggseqlogo, Gviz, magrittr, org.Hs.eg.db, plyr, reshape2, rtracklayer, stats, stringr, tibble, tidyr, TxDb.Hsapiens.UCSC.hg19.knownGene, TxDb.Hsapiens.UCSC.hg38.knownGene, utils

**URL** https://github.com/itsvenu/ALPS

**BugReports** https://github.com/itsvenu/ALPS/issues

**Suggests** knitr, rmarkdown, ComplexHeatmap, circlize, testthat

**biocViews** Epigenetics, Sequencing, ChIPSeq, ATACSeq, Visualization, Transcription, HistoneModification

**VignetteBuilder** knitr

**git_url** https://git.bioconductor.org/packages/ALPS

**git_branch** RELEASE_3_11

**git_last_commit** 3daf3ac
**git_last_commit_date** 2020-04-27
**Date/Publication** 2020-10-16

# R topics documented:



---

`get_genomic_annotations` *Annotate genomic regions*

---

## Description

annotate genomic regions by simultaneosuly merging overlaping regions and preparing consensus set of genomic regions from multiple files

## Usage

```
get_genomic_annotations(data_table, ref_gen = "hg38",
  tss_region = c(-1000, 1000), merge_level = "all")
```

## Arguments

| | |
|---|---|
| `data_table` | a data.frame of sample table, as is for `multiBigwig_summary` input table, default NULL |
| `ref_gen` | reference genome, either `hg38` or `hg19`, default `hg38` |
| `tss_region` | bp ± TSS to define promoter regions |
| `merge_level` | either `all`, `group_level` or `none`. `all` prepares a consensus set of peaks from all files present in `data_table`. `group_level` prepares consensus set of peaks from all samples present each group separately. `none` does not prepare any consensus peak set, annotates each peak file separately. Default `all` |

## Value

a data.frame of annotations and percentages

## Examples

```
chr21_data_table <- system.file('extdata/bw', 'ALPS_example_datatable.txt', package = 'ALPS', mustWork = TRUE)

## attach path to bw_path and bed_path
d_path <- dirname(chr21_data_table)

chr21_data_table <- read.delim(chr21_data_table, header = TRUE)
chr21_data_table$bw_path <- paste0(d_path, '/', chr21_data_table$bw_path)
chr21_data_table$bed_path <- paste0(d_path, '/', chr21_data_table$bed_path)

get_genomic_annotations(data_table = chr21_data_table,
merge_level = 'group_level')
```

---

# get_variable_regions *Get variable regions*

---

## Description

given a data.frame of genomic regions enrichments, the function returns the number of variable regions across all samples. The resulting matrix can be directly used with `PCAtools` or `ComplexHeatmap` for further downstream explorative analysis e.g. unsupervised clustering

## Usage

```
get_variable_regions(enrichments_df, log_transform = TRUE,
  scale = TRUE, num_regions = 500)
```

## Arguments

| | |
|---|---|
| `enrichments_df` | a data.frame o enrichments at genomic regions. Output of `multiBigwig_summary` or a similar format is compatible |
| `log_transform` | logical, whether to log2 transform the counts, default `TRUE` |
| `scale` | logical, whether to `scale` the variable regions before returning the results, default `TRUE` |
| `num_regions` | number of variable regions to return, default `500` |

## Value

a data.frame of scaled variable regions

## Examples

```
mat <- matrix(sample.int(15, 9*100, TRUE), 9, 100) %>% as.data.frame()
mat <- mat %>%
tibble::rowid_to_column(var = 'start') %>%
dplyr::mutate(end = start + 1000) %>%
dplyr::mutate(chr = 'chr1') %>%
dplyr::select(chr, start, end, dplyr::everything())

get_variable_regions(enrichments_df = mat, num_regions = 50)
```

## merge_GR *Merge genomic regions*

### Description

merge overlaping genomic regions from multiple peak/bed files

### Usage

```
merge_GR(x)
```

### Arguments

| | |
|---|---|
| x | a character vector of bed file paths |

### Value

GRanges object

### Examples

```
## load example data

chr21_data_table <- system.file('extdata/bw', 'ALPS_example_datatable.txt',
package = 'ALPS', mustWork = TRUE)

## attach path to bw_path and bed_path
d_path <- dirname(chr21_data_table)

chr21_data_table <- read.delim(chr21_data_table, header = TRUE)
chr21_data_table$bw_path <- paste0(d_path, '/', chr21_data_table$bw_path)
chr21_data_table$bed_path <- paste0(d_path, '/', chr21_data_table$bed_path)

x <- as.character(chr21_data_table$bed_path)

merge_GR(x = x)
```

## multiBigwig_summary *Enrichments at genomics regions*

### Description

`multiBigwig_summary` is a function to calculate enrichments from a set of given bigwig files and a set of genomics regions. This function is similar to `deeptools` `multiBigwigSummary` python package.

### Usage

```
multiBigwig_summary(data_table, summary_type = "mean", parallel = TRUE)
```

### Arguments

| | |
|---|---|
| data_table | a dataframe that contains bw_path, sample_id. sample_id ids will be used in the final result matrix |
| summary_type | whether to calculate mean, median, min or max for each genomic region in within consensus peak-set from the bigwig files. Default is mean |
| parallel | logical. Whether to parallelize the calculation process, default is TRUE |

### Value

data.frame of enrichments within given genomic regions

### Examples

```
## load example data

chr21_data_table <- system.file('extdata/bw', 'ALPS_example_datatable.txt', package = 'ALPS', mustWork = TRUE)

## attach path to bw_path and bed_path
d_path <- dirname(chr21_data_table)

chr21_data_table <- read.delim(chr21_data_table, header = TRUE)
chr21_data_table$bw_path <- paste0(d_path, '/', chr21_data_table$bw_path)
chr21_data_table$bed_path <- paste0(d_path, '/', chr21_data_table$bed_path)

enrichments <- multiBigwig_summary(data_table = chr21_data_table,
                                   summary_type = 'mean',
                                   parallel = FALSE)
```

---

## plot_browser_tracks *UCSC Genome browser like plots*

---

### Description

Function to plot genome browser like plots given a very minimal information such as a set of bigwig files and a genomics region. Tracks are arranged as they are in given input data.frame. Function uses highly customizable Gviz R/bioconductor package to plot browser like plots.

### Usage

```
plot_browser_tracks(data_table, gene_range = NULL, ref_gen = "hg38",
  cex.axis = 0.5, cex.title = 0.8, ...)
```

### Arguments

| | |
|---|---|
| data_table | a dataframe that contains bw_path, sample_id and color_code. Tracks are colored according to color_code. Default NULL |
| gene_range | genomic region for which browser-like plots needed in format chr:start-end. Default NULL |
| ref_gen | reference genome, to get gene annotations, currently supports hg19 and hg38. Default, hg38 |

| | |
|---|---|
| cex.axis | axis label size, default 0.5 |
| cex.title | axis title size, default 0.8 |
| ... | additional arguments to change the appearence of a plot. All arguments that can be passed to base R graphics are supported |

## Value

plot of genome browser tracks

## Examples

```
## load example data

chr21_data_table <- system.file('extdata/bw', 'ALPS_example_datatable.txt', package = 'ALPS', mustWork = TRUE)

## attach path to bw_path and bed_path
d_path <- dirname(chr21_data_table)

chr21_data_table <- read.delim(chr21_data_table, header = TRUE)
chr21_data_table$bw_path <- paste0(d_path, '/', chr21_data_table$bw_path)
chr21_data_table$bed_path <- paste0(d_path, '/', chr21_data_table$bed_path)

gene_range = 'chr21:45643725-45942454'

plot_browser_tracks(data_table = chr21_data_table,
gene_range = gene_range, ref_gen = 'hg38')
```

---

# plot_correlation *Correlations among replicates/groups*

---

## Description

Function to calculate correlations of ChIP/ATAC-seq enrichment among replicates/samples or groups. The function is compatible with the output of multiBigwig_summary or any custom data.frame with the similar format.

## Usage

```
plot_correlation(enrichments_df, log_transform = TRUE,
  method = "pearson", plot_type = "replicate_level", sample_metadata,
  ...)
```

## Arguments

| | |
|---|---|
| enrichments_df | dataframe of enrichments, usually in the form of the output from function multiBigwig_summary, default NULL |
| log_transform | logical, whether to log2 transform enrichments_df |
| method | method to calculate correlation coefficient, one of 'pearson' (default), 'spearman' or 'kendall' |

plot_type — whether to plot replicate_level correlations or group_level correlations. replicate_level plot represents the pairwise correlation values among all the samples/columns, where as the group_level plot is a paired plot of genomic regions after averaging of all samples in a group. Default replicate_level

sample_metadata — a data.frame required if plot_type = 'group_level'. The data.frame must contain columns sample_id and group

... — additional arguments either to corrplot::corrplot or GGally::ggpairs depending on arg plot_type

## Value

corrplot or ggplot2 object

## Examples

```
## load example data
## load example data

chr21_data_table <- system.file('extdata/bw', 'ALPS_example_datatable.txt', package = 'ALPS', mustWork = TRUE)

## attach path to bw_path and bed_path
d_path <- dirname(chr21_data_table)

chr21_data_table <- read.delim(chr21_data_table, header = TRUE)
chr21_data_table$bw_path <- paste0(d_path, '/', chr21_data_table$bw_path)
chr21_data_table$bed_path <- paste0(d_path, '/', chr21_data_table$bed_path)

enrichments <- multiBigwig_summary(data_table = chr21_data_table,
                                   summary_type = 'mean',
                                   parallel = TRUE)

## replicate_level correlation plot
plot_correlation(enrichments_df = enrichments,
log_transform = TRUE, plot_type = 'replicate_level',
sample_metadata = chr21_data_table)

## group_level correlation plot
plot_correlation(enrichments_df = enrichments,
log_transform = TRUE, plot_type = 'group_level',
sample_metadata = chr21_data_table)
```

---

# plot_enrichments *Enrichment plots*

---

## Description

Function to plot enrichments from ChIP-seq/ATAC-seq at genomics regions either as an individual groups or as paired condition e.g untreated-treated

## Usage

```
plot_enrichments(enrichments_df = NULL, log_transform = TRUE,
  plot_type = "separate", sample_metadata, box_alpha = 0.8,
  violin_alpha = 0.8, x_order = NULL, overlap_order = NULL)
```

## Arguments

| | |
|---|---|
| enrichments_df | enrichments at genomics regions from all samples, as in the format of output from multiBigwig_summary |
| log_transform | logical. Should the data be log2 transformed? Default is TRUE |
| plot_type | either separate or overlap |
| sample_metadata | metadata associated with the columns present in enrichments_df, information in this table will be used depending on the option in plot_type |
| box_alpha | alpha/transparency to use for box plots, default 0.8 |
| violin_alpha | alpha/transparecny to use for violin plots, default 0.8 |
| x_order | ordering of levels on x-axis in resulting plot, default NULL |
| overlap_order | ordering of overlaying if plot_type = 'overlap'. E.g. overlaying treatment data on top of untreated data, default NULL |

## Value

ggplot2 object

## Examples

```
## load example data
chr21_data_table <- system.file('extdata/bw', 'ALPS_example_datatable.txt', package = 'ALPS', mustWork = TRUE)

## attach path to bw_path and bed_path
d_path <- dirname(chr21_data_table)

chr21_data_table <- read.delim(chr21_data_table, header = TRUE)
chr21_data_table$bw_path <- paste0(d_path, '/', chr21_data_table$bw_path)
chr21_data_table$bed_path <- paste0(d_path, '/', chr21_data_table$bed_path)

enrichments <- multiBigwig_summary(data_table = chr21_data_table,
                                   summary_type = 'mean',
                                   parallel = TRUE)

## plot_type == 'separate'
plot_enrichments(enrichments_df = enrichments, log_transform = TRUE,
plot_type = 'separate', sample_metadata = chr21_data_table)

## plot_type == 'overlap'
enrichemnts_4_overlapviolins <- system.file('extdata/overlap_violins', 'enrichemnts_4_overlapviolins.txt', p
enrichemnts_4_overlapviolins <- read.delim(enrichemnts_4_overlapviolins, header = TRUE)

## metadata associated with above enrichments
data_table_4_overlapviolins <- system.file('extdata/overlap_violins', 'data_table_4_overlapviolins.txt', pac
data_table_4_overlapviolins <- read.delim(data_table_4_overlapviolins, header = TRUE)
```

```
plot_enrichments(enrichments_df = enrichemnts_4_overlapviolins, log_transform = FALSE,
plot_type = 'overlap', sample_metadata = data_table_4_overlapviolins)
```

---

plot_genomic_annotations *Plot genomic annotations*

---

## Description

plot the annotations of genomic regions either as stacked bar or heatmap. The function takes the output of get_genomic_annotations directly or it is also compatible with a similar data.frame.

## Usage

```
plot_genomic_annotations(annotations_df, plot_type = "bar", col = NULL)
```

## Arguments

| | |
|---|---|
| annotations_df | a data.frame of genomic annotations. It can either be of one sample or of multiple samples as a data.frame |
| plot_type | either bar plot or heatmap, default bar |
| col | vector of colors for each feature in annotations_df. If provided these colors are used, if not a custom set of distinct colors will be used. Default NULL |

## Value

ggplot2 plot

## Examples

```
## load example data

chr21_data_table <- system.file('extdata/bw', 'ALPS_example_datatable.txt', package = 'ALPS', mustWork = TRUE)

## attach path to bw_path and bed_path
d_path <- dirname(chr21_data_table)

chr21_data_table <- read.delim(chr21_data_table, header = TRUE)
chr21_data_table$bw_path <- paste0(d_path, '/', chr21_data_table$bw_path)
chr21_data_table$bed_path <- paste0(d_path, '/', chr21_data_table$bed_path)

g_annotations <- get_genomic_annotations(data_table = chr21_data_table,
merge_level = 'group_level')

plot_genomic_annotations(annotations_df = g_annotations, plot_type = 'heatmap')
```

## plot_motif_logo *Plot sequence motifs*

### Description

Function to plot transscription factor motif logos in two different styles, bar plot or logo plot. It supports motif formats from different databases e.g. JASPAR, MEME, TRANSFAC, HOMER and PFM

### Usage

```
plot_motif_logo(motif_path, database = NULL, plot_type = "bar")
```

### Arguments

| | |
|---|---|
| motif_path | path to motif file, default NULL |
| database | database name from which motif has taken, default NULL |
| plot_type | either bar or logo, default bar |

### Value

ggplot2 object

### Examples

```
## examplr motif file paths
myc_meme <- system.file('extdata/motifs', 'MA0147.2.meme', package = 'ALPS', mustWork = TRUE)
myc_jaspar <- system.file('extdata/motifs', 'MA0147.2.jaspar', package = 'ALPS', mustWork = TRUE)
myc_transfac <- system.file('extdata/motifs', 'MA0147.2.transfac', package = 'ALPS', mustWork = TRUE)
myc_homer <- system.file('extdata/motifs', 'cmyc.homer', package = 'ALPS', mustWork = TRUE)
myc_pfm <- system.file('extdata/motifs', 'MA0147.2.pfm', package = 'ALPS', mustWork = TRUE)

## plot motifs
plot_motif_logo(motif_path = myc_homer, database = 'homer', plot_type = 'logo')
```

## process_homer *Process homer format*

### Description

Process homer format

### Usage

```
process_homer(x)
```

### Arguments

| | |
|---|---|
| x | path to homer format file |

### Value

data.frame

### Examples

```
myc_homer <- system.file('extdata/motifs', 'cmyc.homer', package = 'ALPS', mustWork = TRUE)
myc_df <- process_homer(x = myc_homer)
```

---

## process_jaspar *Process jaspar format*

---

### Description

Process jaspar format

### Usage

```
process_jaspar(x)
```

### Arguments

| | |
|---|---|
| x | path to jaspar format file |

### Value

data.frame

### Examples

```
myc_jaspar <- system.file('extdata/motifs', 'MA0147.2.jaspar', package = 'ALPS', mustWork = TRUE)
myc_df <- process_jaspar(x = myc_jaspar)
```

---

## process_meme *Process meme format*

---

### Description

Process meme format

### Usage

```
process_meme(x)
```

### Arguments

| | |
|---|---|
| x | path to meme format file |

### Value

data.frame

## Examples

```
myc_meme <- system.file('extdata/motifs', 'MA0147.2.meme', package = 'ALPS', mustWork = TRUE)
myc_df <- process_meme(x = myc_meme)
```

---

`process_pfm` *Process PFM format*

---

## Description

Process PFM format

## Usage

```
process_pfm(x)
```

## Arguments

| | |
|---|---|
| x | path to PFM format file |

## Value

data.frame

## Examples

```
myc_pfm <- system.file('extdata/motifs', 'MA0147.2.pfm', package = 'ALPS', mustWork = TRUE)
myc_df <- process_pfm(x = myc_pfm)
```

---

`process_transfac` *Process transfac format*

---

## Description

Process transfac format

## Usage

```
process_transfac(x)
```

## Arguments

| | |
|---|---|
| x | path to transfac format file |

## Value

data.frame

## Examples

```
myc_transfac <- system.file('extdata/motifs', 'MA0147.2.transfac', package = 'ALPS', mustWork = TRUE)
my_df <- process_transfac(x = myc_transfac)
```

# Index