Package ‘dREG’

January 18, 2018

Version 1.2.0

Date 2017-12-19

Title Detection of Regulatory DNA using PRO-seq, GRO-seq Data(dREG)

Author Charles G. Danko <dankoc@gmail.com>

Maintainer Charles G. Danko <dankoc@gmail.com>

Depends
R (>= 2.14), bigWig (>= 0.2-9), e1071, rphast, snowfall, data.table, rmutil, mvtnorm, methods

LinkingTo

Suggests data.table, Rgtsvm

Description This package is an analysis pipeline for the analysis of GRO-seq data.

License GPL-3

biocViews Sequencing, Analysis

LazyLoad yes

R topics documented:

combine.roc *Combines ROC plots*

Description

Combines ROC plots, interpolating and weighting by nTP.

Usage

```
combine.roc(list.roc,
      weight = rep(1, NROW(list.roc)),
      interp.corners = FALSE,
      use.max = FALSE,
      nvals = 100)
```

Arguments

`list.roc`	List including multiple ROC data frame
`weight`	Weight vector for each ROC dataframe
`interp.corners`	Logical value indicating if the header(1,1) and tail values(0,0) are interpolated to each ROC data frame.
`use.max`	Logical value indicating if maximum value of muliple ROCs at same point are used as TPF values.
`nvals`	Integer value indicating interval number for ROC plot.

Value

A data frame with 2 columns is returned

`FPR`	False Positive Rate
`TPR`	True Positive Rate

References

Danko, C. G., Hyland, S. L., Core, L. J., Martins, A. L., Waters, C. T., Lee, H. W., ... & Siepel, A. (2015). Identification of active transcriptional regulatory elements from GRO-seq data. Nature methods, 12(5), 433-438.

See Also

`roc.calc`, `logreg.roc.calc`, `roc.auc`, `roc.plot`

Examples

```
list.roc<-list();

true <- c(rep(1, 100), rep(0, 100));
scores <- c( rnorm(100, 1, 1 ), rnorm(100, 0, 1 ) );
list.roc[[1]] <- logreg.roc.calc( true, scores );
```

```
true <- c(rep(1, 120), rep(0, 110));
scores <- c( rnorm(120, 1, 0.8 ), rnorm(110, 0, 1.2 ) );
list.roc[[2]] <- logreg.roc.calc( true, scores );

r <- combine.roc(list.roc);
roc.plot(r)
```

`eval_reg_svm` *Evaluates a set of genomic coordinates for regulatory potential using P/GRO-seq data*

Description

Evaluates a set of genomic coordinates for regulatory potential using P/GRO-seq data

Usage

```
eval_reg_svm(gdm,
      asvm,
      positions,
      bw_plus_path,
      bw_minus_path,
      batch_size = 50000,
      ncores = 3,
      use_rgtsvm = FALSE,
      debug = TRUE)
```

Arguments

`gdm` Genomic data model return by `genomic_data_model`.

`asvm` A pre-trained SVM model from the e1071 package returned by `regulatory_svm`.

`positions` Data frame with 2 columns indicating the universe of positions to test and evaluate(chrom,chromCenter). It can be returned by `get_informative_positions`.

`bw_plus_path` String value indicating file path to bigWig file representing the plus strand.

`bw_minus_path` String value indicating file path to bigWig file representing the minus strand

`batch_size` Number of positions to evaluate at once (more might be faster, but takes more memory).

`ncores` Number of CPU cores in parallel computing

`use_rgtsvm` Indictating whether the predict will be performed on GPU through the Rgtsvm package.

`debug` Logical value indicating the process detail is outputted.

Value

Returns the value of the SVM for each genomic coordinate specified.

References

Danko, C. G., Hyland, S. L., Core, L. J., Martins, A. L., Waters, C. T., Lee, H. W., ... & Siepel, A. (2015). Identification of active transcriptional regulatory elements from GRO-seq data. Nature methods, 12(5), 433-438.

See Also

get_informative_positions, get_test_set, read_genomic_data, regulatory_svm

Examples

```
## The following codes cannot run without the bigWig files

# ps_plus_path  <- "bigwig.plus.bw"
# ps_minus_path <- "bigwig.minus.bw"

## Now scan all positions in the genome ...
# positions <- get_informative_positions(ps_plus_path, ps_minus_path,
# depth= 0, step=50, use_ANDOR=TRUE, use_OR=FALSE);

# pred_val<- eval_reg_svm( gdm, asvm, inf_positions, ps_plus_path, ps_minus_path, batch_size=50000)
# write.table( data.frame(inf_positions, pred_val), file="eval.tab",
# row.names=FALSE, col.names=FALSE, quote=FALSE, sep="\t")
```

genomic_data_model *Creates a genome data model.*

Description

Creates a genome data model.

Usage

```
genomic_data_model(window_sizes, half_nWindows)
```

Arguments

window_sizes	Number indicating the width of genomic window.
half_nWindows	Number indicating the count of genomic window at each side(left side or right side).

Value

A s4 object is returned with

n_zooms	Number indicating zoom ratio.
window_sizes	Vector indicating window sizes.
half_nWindows	Vector indicating number of half windows.

References

Danko, C. G., Hyland, S. L., Core, L. J., Martins, A. L., Waters, C. T., Lee, H. W., ... & Siepel, A. (2015). Identification of active transcriptional regulatory elements from GRO-seq data. Nature methods, 12(5), 433-438.

See Also

`read_genomic_data`, `regulatory_svm`, `eval_reg_svm`

Examples

```
gdm <- genomic_data_model( c(10,20,30), c(10, 10, 10) )
```

`get_informative_positions` *Gets center positions that pass a minimum depth filter*

Description

Returns a data frame with center positions that pass a minimum depth filter

Usage

```
get_informative_positions(bw_path,
      bw_minus_path = NULL,
      depth = 0,
      window = 400,
      step = 50,
      use_OR = TRUE,
      use_ANDOR = TRUE,
      debug = TRUE)
```

Arguments

`bw_path`	String indicating file path to bigwig file representing the plus strand.
`bw_minus_path`	String indicating file path to bigwig file representing the minus strand, If specified, takes the windows that pass the step in both bigWig files.(intersection)
`depth`	Integer value indicating minimum number of reads to return.
`window`	Integer value indicating window distance between to search for #depth reads [bp].
`step`	Integer value indicating step distance for window list.
`use_OR`	Logical value indicating if the center positions in minus bigwig file are merged into the results. If false, the intersection operation will be performed to the center positions of plus bigwig and from minus bigwig.
`use_ANDOR`	Logical value indicating if the center positions will be merged from the two results. a) Intersection operation with the conditions: window interval=1000 depth>=0. b) Union operation with with the conditions: window interval=100 depth >=2.
`debug`	Logical value indication the process detail is outputted.

Details

The use_ANDOR and use_OR parameter are applied to two Bigwig files as following logical:

```
if(use_ANDOR){
    v1 <- get_window_Or  (window=1000, depth=0);
    v2 <- get_window_and (window=100,  depth=2);
    vals <- c(v1,v2);
}
else {
  if(use_OR){
     vals <- get_window_Or( window=window, depth=depth);
  }
  else {
     vals <- get_window_and( window=window, depth=depth);
  }
}
```

Value

A BED-style data frame will be returned with 3 columns

chrom	Chromosome information
chromStart	Start position
chromEnds	End position

References

Danko, C. G., Hyland, S. L., Core, L. J., Martins, A. L., Waters, C. T., Lee, H. W., ... & Siepel, A. (2015). Identification of active transcriptional regulatory elements from GRO-seq data. Nature methods, 12(5), 433-438.

See Also

get_test_set, read_genomic_data, regulatory_svm, eval_reg_svm

---

get_test_set *Returns a genome loci of positive set and negative set for SVM training purpose.*

Description

Returns a genome loci of positive set and negative set for SVM training purpose

Usage

```
get_test_set(positions,
      positive,
      n_samp,
      allow = NULL,
      enrich_negative_near_pos = 0.15,
      extra_enrich_bed = NULL,
      extra_enrich_frac = 0.1,
      avoid_dist = 100)
```

Arguments

`positions` Bed-style data frame indicating the universe of positions to test and evaluate (chrom,chromCenter).

`positive` Bed-style data frame containing positive positions (chrom,chromStart,chromEnd).

`n_samp` Number of training examples

`allow` Bed-style data frame containing inverse negative set of positions (chrom,chromStart,chromEnd).

`enrich_negative_near_pos`
Fraction of training examples chosen to be nearby (<=5kb) a positive example [0,1].

`extra_enrich_bed`
Bed-style data frame indicating extra bed file to enrich near.

`extra_enrich_frac`
Fraction of final positions sampled in the negative set which are in the bed file. Unused if extra_enrich_bed is NULL.

`avoid_dist` Integer value indicating how long extend avoiding genomic loci.

Details

(1). The parameter of `positions` can be obtained by `get_informative_positions`.

Value

Returns a data frame including double number of the _train set(2*n_samp), each sample includes 4 items.

`chrom`

`chromStart`

`chromEnd`

`status` 1 for positive and 0 for negative.

References

Danko, C. G., Hyland, S. L., Core, L. J., Martins, A. L., Waters, C. T., Lee, H. W., ... & Siepel, A. (2015). Identification of active transcriptional regulatory elements from GRO-seq data. Nature methods, 12(5), 433-438.

See Also

`get_informative_positions`, `read_genomic_data`, `regulatory_svm`, `eval_reg_svm`

`logreg.roc.calc` *Calculates the TPR and FPR for a ROC plot.*

Description

Calculates the TPR and FPR for a ROC plot from the status and score vector.

Usage

```
logreg.roc.calc(true, scores)
```

Arguments

`true`	Vector indicating the two status, 1 and 0.
`scores`	Vector indicating the scores for each status calculated by the predict function.

Details

The function of `roc.calc` calculates a ROC matrix for the genomic loci, whereas the function of `logreg.roc.calc` calculates for a status vector.

Value

A data frame with 3 columns is returned, which is same as `roc.calc`.

`FPR`	False Positive Rate.
`TPR`	True Positive Rate.
`threshold`	Threshold based on the score parameter.

References

Danko, C. G., Hyland, S. L., Core, L. J., Martins, A. L., Waters, C. T., Lee, H. W., ... & Siepel, A. (2015). Identification of active transcriptional regulatory elements from GRO-seq data. Nature methods, 12(5), 433-438.

See Also

`roc.calc`, `combine.roc`, `roc.auc`,`roc.plot`

Examples

```
true <- c(rep(1, 100), rep(0, 100));
scores <- c( rnorm(100, 1, 1 ), rnorm(100, 0, 1 ) );
roc_mat <- logreg.roc.calc( true, scores );
AUC<- roc.auc(roc_mat);
roc.plot(roc_mat, main=AUC );
```

`peak_calling` *Peak calling based on dREG prediction*

Description

This procedure calls SVR prediction for paired bigWig files using pre-trained SVM model and detects divergent peaks based on the predicted score.

Usage

```
peak_calling(asvm, gdm,
      bw_plus_path, bw_minus_path,
      infp_bed = NULL,
      use_rgtsvm = TRUE,
      min_score = NULL,
      pv_adjust="fdr",
      pv_threshold=0.05,
      smoothwidth = 4,
      cpu_cores=1,
      gpu_cores=1)
```

Arguments

`asvm`	SVR model pre-trained for dREG package, which can be downloaded from the dREG page in Github.
`gdm`	Genomic data model return by `genomic_data_model`. This data is binding with SVR model.
`bw_plus_path`	File name indicating file path to bigWig file representing the plus strand.
`bw_minus_path`	File name indicating file path to bigWig file representing the minus strand.
`infp_bed`	A BED data frame indicating informative sites and scores returned by `eval_reg_svm`. If NULL is specified, the peak calling starts from the informative sites finding and predicting.
`use_rgtsvm`	Logical value indictating whether the predict will be performed on GPU through the Rgtsvm package.
`min_score`	Numerical value indicating the minimum dREG score applied to the peak calling procedure. If NULL is specified, this value is calculated based on the predicted scores.
`pv_adjust`	String value indictating which correction method is used to do multiple comparison, see details in `p.adjust`, default is 'fdr'.
`pv_threshold`	Numerical value indicating the threshold is used to report the dREG peaks.
`smoothwidth`	Numerical value indicating the parameter of curve smooth in the moving average.
`cpu_cores`	Number of CPU cores in parallel computing.
`gpu_cores`	Number of GPU cores in parallel computing if **Rgtsvm** is used.

Value

This function returns a list containing 6 items, including:

1) dREG peaks: `peak_bed`

`chr`	Chromosome
`start`	Start position
`end`	End position
`score`	Maxmimum score in the peak region
`prob`	Probability of multivariate Laplace distribution indicating the probability of the peak points belonging to negative set (No divergent peak).
`center`	the center position in original peak

2) Informative sites with score `infp_bed`

`chr`	Chromosome
`start`	Start position
`end`	End position
`score`	predicted score
`infp`	indicating the informative site or dense site

3) Broad peak regions `peak_broad`

`chr`	Chromosome
`start`	Start position
`end`	End position
`no`	index
`min`	minimum score in this region
`max`	maximum score in this region
`mean`	score mean in this region
`sum`	score um in this region
`stdev`	standard deviation of scores in this region
`count`	informative site in this region

4) Threshold of dREG score `min_score`

5) Raw results of peak calling `raw_peak`

6) Summary of peak calling `peak_sum`

`adjust.none.0.05`	The count of dREG peak withou p-value correction.
`adjust.fdr.0.05`	The count of dREG peak adjusted by the 'fdr' method.
`adjust.BH.0.05`	The count of dREG peak adjusted by the 'BH' method.
`adjust.bonferroni.0.05`	The count of dREG peak adjusted by the 'bonferroni' method.

`adjust.holm.0.05`
The count of dREG peak adjusted by the 'holm' method.

`adjust.hochberg.0.05`
The count of dREG peak adjusted by the 'hochberg' method.

`adjust.BY.0.05` The count of dREG peak adjusted by the 'BY' method.

`peak.sig.score` The score range of significant dREG peaks.

`peak.narrow100` The ignored narrow peaks which length are less than 100.

`peak.narrow100.sig`
The ignored narrow peaks which may be significant based on 'peak.sig.score'.

`peak.narrow100.score`
The score range of the ignored narrow peaks.

Examples

```
# load("../asvm.6.6M.20170828.rdata");
# gdm <- genomic_data_model(window_sizes= c(10, 25, 50, 500, 5000), half_nWindows= c(10, 10, 30, 20, 20) )
# bw_plus_path <- "K562.chr21.plus.bw"
# bw_minus_path <- "K562.chr21.minus.bw"
# x <- peak_calling( svm, gdm, bw_plus_path, bw_minus_path, cpu_cores=12, use_rgtsvm=T)
# show(x$peak_bed);
```

`read_genomic_data` *Gets read data from the specified genomic position.*

Description

Gets read data from the specified genomic position.

Usage

```
read_genomic_data( gdm,
      bed,
      file_bigwig_plus,
      file_bigwig_minus,
      as_matrix = TRUE,
      scale.method = c("logistic", "linear"),
      batch_size = 50000,
      ncores = 1 )
```

Arguments

`gdm` Genomic data model return by `genomic_data_model`.

`bed` bed-style data frame of genomic regions.(at least 3 columns including chrom, start, end).

`file_bigwig_plus`
String value indicating file path to bigwig file representing GRO-seq/ PRO-seq reads on the plus strand.

`file_bigwig_minus`
String value indicating file path to bigwig file representing GRO-seq/ PRO-seq reads on the minus strand.

`as_matrix` Logical type,if true, returns a matrix object, otherwise returns a list() object, where each element in the list is the zoom data.

`scale.method` String value indicating the normalize method of read counts. Two options are available, "logistic" or "linear", default value is logistic. See details

`batch_size` Number of genomic positions to evaluate at once (more might be faster, but takes more memory)

`ncores` Number of CPU cores in parallel computing

Details

Data normalize method:
(1): Logistic function: F(x) = 1/(1+exp(-a*(x-b))
(2): Linear function: F(x) = x / tootal_reads

Value

A matrix of normalized read count, the columns are windows list specified by `gdm` object.

References

Danko, C. G., Hyland, S. L., Core, L. J., Martins, A. L., Waters, C. T., Lee, H. W., ... & Siepel, A. (2015). Identification of active transcriptional regulatory elements from GRO-seq data. Nature methods, 12(5), 433-438.

See Also

`get_informative_positions`, `get_test_set`, `regulatory_svm`, `eval_reg_svm`

Examples

```
file_bigwig_plus <- "";
file_bigwig_minus <- "";
gdm <- genomic_data_model(20, 10);
#mat <- read_genomic_data(gdm, bed, file_bigwig_plus, file_bigwig_minus);
#summary(mat);
```

regulatory_svm *Trains a SVM to recognize a certain pattern of regulatory positions*

Description

Trains a SVM to recognize a certain pattern of regulatory positions.

Usage

```
regulatory_svm(gdm,
      bw_plus_path,
      bw_minus_path,
      positions, positive,
      allow = NULL,
      n_train = 25000,
      n_eval = 1000,
      pdf_path = "roc_plot.pdf",
      plot_raw_data = TRUE,
      extra_enrich_bed = NULL,
      extra_enrich_frac = 0.1,
      enrich_negative_near_pos = 0.15,
      use_rgtsvm = FALSE,
      svm_type = "SVR",
      ncores = 1,
      ...,
      debug = TRUE)
```

Arguments

`gdm` Genomic data model returned by `genomic_data_model`.

`bw_plus_path` String indicating file path to bigWig file representing the plus strand.

`bw_minus_path` String indicating file path to bigWig file representing the minus strand.

`positions` Data frame with two columns(chrom,chromCenter), indicating the universe of positions to test and evaluate. It can be generated by `get_informative_positions`.

`positive` Bed-style data frame containing positive positions(chrom,chromStart,chromEnd).

`allow` Bed-style data frame containing positions to avoid in the negative set(chrom,chromStart,chromEnd).

`n_train` Number of training examples.

`n_eval` Number of examples on which to test performance.

`pdf_path` String value indicating a PDF file. Set to NULL if no PDF should be printed.

`plot_raw_data` If TRUE (default), and if a PDF file is specified, plots the raw data used to train the model.

`extra_enrich_bed`
Bed-style data frame indicating extra bed file to enrich near. Used by `get_test_set`.

`extra_enrich_frac`
Fraction of final positions sampled in the negative set which are in the bed file. Unused if extra_enrich_bed is NULL. Used by `get_test_set`.

`enrich_negative_near_pos`
Fraction of training examples chosen to be nearby (<=5kb) a positive example [0,1].

`use_rgtsvm` Indictating whether the predict will be performed on GPU through the Rgtsvm package.

`svm_type` Two options, "SVR" for support vecctor regression (epsilon-regression). "P_SVM" for probabilistic SVM (C-classification).

`ncores` Integer indicating how many cores are used to improve the performance.

`...` The parameters for `plot` function

`debug` Logical value indication the process detail is outputted.

Value

A svm model trained by `svm` function in e1071 package.

References

Danko, C. G., Hyland, S. L., Core, L. J., Martins, A. L., Waters, C. T., Lee, H. W., ... & Siepel, A. (2015). Identification of active transcriptional regulatory elements from GRO-seq data. Nature methods, 12(5), 433-438.

See Also

`get_informative_positions`, `get_test_set`, `read_genomic_data`, `eval_reg_svm`

`roc.auc` *Computes the AUC of a ROC plot.*

Description

Computes the AUC of a ROC plot.

Usage

```
roc.auc(ROC)
```

Arguments

`ROC`	A matrix with 3 columns (FPR, TPR and threshold) calculated by `logreg.roc.calc`.

Details

The parameter of ROC is a matrix or data frame including 3 columns, FPR(False Positive Rate), TPR(True Positive Rate) and threshold.

Value

AUC value is returned.

References

Danko, C. G., Hyland, S. L., Core, L. J., Martins, A. L., Waters, C. T., Lee, H. W., ... & Siepel, A. (2015). Identification of active transcriptional regulatory elements from GRO-seq data. Nature methods, 12(5), 433-438.

See Also

`roc.calc`, `logreg.roc.calc`, `combine.roc`, `roc.plot`

Examples

```
roc_mat <- data.frame( FPR=c(0, 0.25, 0.5, 0.75, 1),
TPR=c(0, 0.5, 0.8, 0.95, 1),
threshold=c(1, 1, 1, 1, 1) );
AUC<- roc.auc( roc_mat );
roc.plot( roc_mat, main=AUC );
```

`roc.calc` *Calculates the TPR and FPR for a ROC plot.*

Description

Calculates the TPR and FPR for a ROC plot.

Usage

```
roc.calc(true,
      possible,
      scores,
      filterPossible = TRUE,
      n_points = 100)
```

Arguments

`true` Bed-style data frame, a set of 'true' genomic intervals (e.g. ChIP-seq peaks).

`possible` Bed-style data frame, A set of 'possible' genomic intervals (e.g. DNAse-1 peaks).

`scores` Vector indicating the scores for each possibe genomic interval in parameter of `possible`.

`filterPossible` Vector indicating indexes which be removed.

`n_points` Integer indicating how many points for the ROC plot.

Value

A data frame with 3 columns is returned

`FPR` False Positive Rate

`TPR` True Positive Rate

`threshold` Threshold based on the score parameter.

References

Danko, C. G., Hyland, S. L., Core, L. J., Martins, A. L., Waters, C. T., Lee, H. W., ... & Siepel, A. (2015). Identification of active transcriptional regulatory elements from GRO-seq data. Nature methods, 12(5), 433-438.

See Also

`logreg.roc.calc`, `combine.roc`, `roc.auc`, `roc.plot` `roc.plot` *Draws a ROC figure.*

Description

Draws a ROC figure.

Usage

```
roc.plot(ROC, ...)
```

Arguments

ROC	Matrix or data frame with 3 columns, FPR, TPR and threshold.
...	The parameters for `plot` function

Value

None

References

Danko, C. G., Hyland, S. L., Core, L. J., Martins, A. L., Waters, C. T., Lee, H. W., ... & Siepel, A. (2015). Identification of active transcriptional regulatory elements from GRO-seq data. Nature methods, 12(5), 433-438.

See Also

`roc.calc`, `logreg.roc.calc`, `combine.roc`, `roc.auc`

Examples

```
true <- c(rep(1, 100), rep(0, 100));
scores <- c( rnorm(100, 1, 1 ), rnorm(100, 0, 1 ) );
roc_mat <- logreg.roc.calc( true, scores );
AUC<- roc.auc(roc_mat);
roc.plot(roc_mat, main=AUC );
```

Index