Package ‘preciseTAD’

October 14, 2021

Type Package

Title preciseTAD: A machine learning framework for precise TAD boundary prediction

Version 1.2.0

Description preciseTAD provides functions to predict the location of boundaries of topologically associated domains (TADs) and chromatin loops at base-level resolution. As an input, it takes BED-formatted genomic coordinates of domain boundaries detected from low-resolution Hi-C data, and coordinates of high-resolution genomic annotations from ENCODE or other consortia. preciseTAD employs several feature engineering strategies and resampling techniques to address class imbalance, and trains an optimized random forest model for predicting low-resolution domain boundaries. Translated on a base-level, preciseTAD predicts the probability for each base to be a boundary. Density-based clustering and scalable partitioning techniques are used to detect precise boundary regions and summit points. Compared with low-resolution boundaries, preciseTAD boundaries are highly enriched for CTCF, RAD21, SMC3, and ZNF143 signal and more conserved across cell lines. The pre-trained model can accurately predict boundaries in another cell line using CTCF, RAD21, SMC3, and ZNF143 annotation data for this cell line.

License MIT + file LICENSE

Depends R (>= 4.0.0)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests knitr, rmarkdown, testthat, BiocCheck, BiocManager, BiocStyle

VignetteBuilder knitr

Imports S4Vectors, IRanges, GenomicRanges, randomForest, ModelMetrics, e1071, PRROC, pROC, caret, utils, cluster, dbscan, doSNOW, foreach, pbapply, stats, parallel, stats

biocViews Software, HiC, Sequencing, Clustering, Classification, FunctionalGenomics, FeatureExtraction

BugReports https://github.com/dozmorovlab/preciseTAD/issues

URL https://github.com/dozmorovlab/preciseTAD

git_url https://git.bioconductor.org/packages/preciseTAD

git_branch RELEASE_3_13

git_last_commit c908e5e

git_last_commit_date 2021-05-19

Date/Publication 2021-10-14

Author Spiro Stilianoudakis [aut, cre],
Mikhail Dozmorov [aut]

Maintainer Spiro Stilianoudakis <stilianoudasc@vcu.edu>

R topics documented:

`arrowhead_gm12878_5kb` *Domain data from ARROWHEAD TAD-caller for GM12878 at 5 kb*

Description

A data frame with 3 columns and 8409 rows

V1 The chromosome number

V2 The start coordinate of the TAD

V3 The end coordinate of the TAD

Usage

```
arrowhead_gm12878_5kb
```

Format

An object of class `data.frame` with 8409 rows and 3 columns.

Value

A data.frame

Source

Data from Rao SS, Huntley MH, Durand NC, Stamenova EK et al. A 3D map of the human genome at kilobase resolution reveals principles of chromatin looping. Cell 2014 Dec 18;159(7):1665-80. PMID: 25497547. Available at `https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE63525`

`bedToGRangesList` *Function to create a GRangesList object from functional genomic annotation data in the form of BED files*

Description

Function to create a GRangesList object from functional genomic annotation data in the form of BED files

Usage

```
bedToGRangesList(
  filepath,
  bedList = NULL,
  bedNames = NULL,
  pattern = "*.bed",
  signal = NULL
)
```

Arguments

`filepath`	Character describing the path to the folder containing the BED files of functional genomic annotations. This is ignored if bedList is specified.
`bedList`	A list object containing the bed data as data frames to be converted into a GRangesList. The data frames must include at least chromosome, start, and end coordinates as the first 3 columns. Default is NULL.
`bedNames`	A character vector to provide names to the GRangesList, should be in the order of bedList. Default is NULL.
`pattern`	Character describing the pattern of the files for the functional genomic annotations. Default is "*.bed".
`signal`	The column number in the BED files that denotes coverage strength. Must be the same for all files. Default is NULL indicating to no coverage value is to be used.

Value

A `GRangesList` object where each entry is a `GRanges` object specific to each BED file in the path provided

Examples

```
#set path
path <- system.file("extdata", package = "preciseTAD")
#contains 2 BED files representing YY1 and NFYA
#transcription factor binding sites for GM12878
tfbsList <- bedToGRangesList(filepath = path, bedList=NULL, bedNames=NULL,
pattern = "*.bed", signal=NULL)
```

`binary_func` *Helper function used to create binary overlap type feature space*

Description

Helper function used to create binary overlap type feature space

Usage

```
binary_func(binned_data_gr, annot_data_gr)
```

Arguments

`binned_data_gr`	A GRanges object
`annot_data_gr`	A Granges object

Value

An indicator vector for whether or not an overlap occured

`count_func` *Helper function used to create count overlap type feature space*

Description

Helper function used to create count overlap type feature space

Usage

```
count_func(binned_data_gr, annot_data_gr)
```

Arguments

`binned_data_gr`	A GRanges object
`annot_data_gr`	A Granges object

Value

A vector of counts enumerating the number of overlaps

`createTADdata` *Function to create a data matrix used for building a predictive model to classify boundary regions from functional genomic elements*

Description

Function to create a data matrix used for building a predictive model to classify boundary regions from functional genomic elements

Usage

```
createTADdata(
  bounds.GR,
  resolution,
  genomicElements.GR,
  featureType = "distance",
  resampling,
  trainCHR,
  predictCHR = NULL
)
```

Arguments

`bounds.GR`	a GRanges object with chromosomal coordinates of TAD boundaries used to identify positive cases (can be obtained using `extractBoundaries`). Required.
`resolution`	Numeric, the width to bin the genome at, should match the resolution that TADs were called at. Required.
`genomicElements.GR`	a GRangesList object containing GRanges objects for each ChIP-seq data to leverage in the random forest model (can be obtained using the `bedToGRangesList`). Required.
`featureType`	Character, controls how the feature space is constructed (one of either "binary" (overlap yes/no), "oc" (overlap counts, the number of overlaps), "op" (overlap percent, the percent of bin width covered by the genomic annotation), or "distance" (log2-transformed distance from the center of the nearest genomic annotation to the center of the bin); default is "distance"). Required.

`resampling`	Character, controls if and how the data should be resampled to create balanced classes of boundary vs. nonboundary regions (one of either "none" - no resampling, "ros" - Random Over-Sampling, "rus" - Random Under-Sampling. Required.
`trainCHR`	Character vector of chromosomes to use to build the binned data matrix for training. Required.
`predictCHR`	Character vector of chromosomes to use to build the binned data matrix for testing. Default in NULL, indicating no test data is created. If trainCHR=predictCHR then a 7:3 split is created.

Value

A list object containing two data.frames: 1) the training data, 2) the test data (only if predictCHR is not NULL, otherwise it is NA). "y" is an indicator whether the corresponding bin is a TAD boundary, and the subsequent columns have the association measures between bins and the genomic annotations

Examples

```
# Create training data for CHR21 and testing data for CHR22 with
# 5 kb binning, oc-type predictors from 26 different transcription factor
# binding sites from the GM12878 cell line, and random under-sampling

# Read in ARROWHEAD-called TADs at 5kb
data(arrowhead_gm12878_5kb)

#Extract unique boundaries
bounds.GR <- extractBoundaries(domains.mat = arrowhead_gm12878_5kb,
                               preprocess = FALSE,
                               CHR = c("CHR21", "CHR22"),
                               resolution = 5000)

# Read in GRangesList of 26 TFBS
data(tfbsList)

tadData <- createTADdata(bounds.GR = bounds.GR,
                         resolution = 5000,
                         genomicElements.GR = tfbsList,
                         featureType = "oc",
                         resampling = "rus",
                         trainCHR = "CHR21",
                         predictCHR = "CHR22")
```

`distance_func` *Helper function used to create (log2) distance type feature space*

Description

Helper function used to create (log2) distance type feature space

Usage

```
distance_func(binned_data_center_gr, annot_data_center_gr)
```

Arguments

`binned_data_center_gr`	A GRanges object of width 1
`annot_data_center_gr`	A GRanges object of width 1

Value

A vector of log2 distances to the nearest overlap

`extractBoundaries` *Function to extract using boundaries from domain data*

Description

Function to extract using boundaries from domain data

Usage

```
extractBoundaries(domains.mat, preprocess = FALSE, CHR, resolution)
```

Arguments

`domains.mat`	either a `matrix` or `data.frame` with at least 3 columns. First column is either of class "numeric" or "integer". The second and third columns are the start and end coordinates of domains, respectively. Required.
`preprocess`	logical, indicating whether or not domains exceeding 2mb in width or smaller than 2*(the specified resolution) should be filtered out (default is FALSE, all boundaries will be used). Required.
`CHR`	character, specifying which chromosome(s) to extract domain boundaries on (ex: "CHR22"). Required.
`resolution`	numeric, the Hi-C data resolution that domains were called at. Ignored if preprocess is FALSE.

Value

A `GRanges` object

Examples

```
#Read in domain data from ARROWHEAD at 5 kb for GM12878
data("arrowhead_gm12878_5kb")
#Extract unique boundaries for CHRs 1-8 and 10-22
bounds.GR <- extractBoundaries(domains.mat=arrowhead_gm12878_5kb,
                               preprocess=FALSE,
                               CHR=paste0("CHR",c(1:8,10:22)),
                               resolution=5000)
```

---

juicer_func — *Helper function for transforming a GRanges object into matrix form to be saved as .txt or .BED file and imported into juicer*

Description

Helper function for transforming a GRanges object into matrix form to be saved as .txt or .BED file and imported into juicer

Usage

```
juicer_func(grdat)
```

Arguments

`grdat`	A GRanges object representing boundary coordinates

Value

A dataframe that can be saved as a BED file to import into juicer

Examples

```
## Not run:
# Read in ARROWHEAD-called TADs at 5kb
data(arrowhead_gm12878_5kb)

# Extract unique boundaries
bounds.GR <- extractBoundaries(domains.mat = arrowhead_gm12878_5kb,
                               preprocess = FALSE,
                               CHR = c("CHR21", "CHR22"),
                               resolution = 5000)

# Read in GRangesList of 26 TFBS
data(tfbsList)

tfbsList_filt <- tfbsList[which(names(tfbsList) %in%
                                                   c("Gm12878-Ctcf-Broad",
                                                     "Gm12878-Rad21-Haib",
                                                     "Gm12878-Smc3-Sydh",
                                                  "Gm12878-Znf143-Sydh"))]

# Create the binned data matrix for CHR1 (training) and CHR22 (testing)
# using 5 kb binning, distance-type predictors from 26 different TFBS from
# the GM12878 cell line, and random under-sampling
tadData <- createTADdata(bounds.GR = bounds.GR,
                         resolution = 5000,
                         genomicElements.GR = tfbsList_filt,
                         featureType = "distance",
                         resampling = "rus",
                         trainCHR = "CHR21",
                         predictCHR = "CHR22")

# Perform random forest using TADrandomForest by tuning mtry over 10 values
# using 3-fold CV
tadModel <- TADrandomForest(trainData = tadData[[1]],
                            testData = tadData[[2]],
                            tuneParams = list(mtry = 2,
                                              ntree = 500,
                                              nodesize = 1),
                            cvFolds = 3,
                            cvMetric = "Accuracy",
                            verbose = TRUE,
                            model = TRUE,
                            importances = TRUE,
                            impMeasure = "MDA",
                            performances = TRUE)

# Apply preciseTAD on a specific 2mb section of CHR22:17000000-19000000
pt <- preciseTAD(genomicElements.GR = tfbsList_filt,
                 featureType = "distance",
                 CHR = "CHR22",
                 chromCoords = list(17000000, 19000000),
                 tadModel = tadModel[[1]],
                 threshold = 1.0,
                 verbose = TRUE,
                 parallel = TRUE,
                 cores = 2,
                 splits = 2,
                 DBSCAN_params = list(10000, 3),
                 flank = NULL)

# Transform into juicer format
juicer_func(pt[[2]])

## End(Not run)
```

`percent_func` *Helper function used to create percent overlap type feature space*

Description

Helper function used to create percent overlap type feature space

Usage

```
percent_func(binned_data_gr, annot_data_gr)
```

Arguments

`binned_data_gr`	A GRanges object
`annot_data_gr`	A Granges object

Value

A vector of proportions indicating the percentage of overlap

`preciseTAD` *Precise TAD boundary prediction at base-level resolution using density-based spatial clustering and partitioning techniques*

Description

Precise TAD boundary prediction at base-level resolution using density-based spatial clustering and partitioning techniques

Usage

```
preciseTAD(
  genomicElements.GR,
  featureType = "distance",
  CHR,
  chromCoords = NULL,
  tadModel,
  threshold = 1,
  verbose = TRUE,
  parallel = NULL,
  DBSCAN_params,
  flank,
  BaseProbs = FALSE
)
```

Arguments

`genomicElements.GR`	`GRangesList` object containing GRanges from each ChIP-seq BED file that was used to train a predictive model (can be obtained using the `bedToGRangesList`). Required.
`featureType`	Controls how the feature space is constructed (one of either "binary", "oc", "op", "signal, or "distance" (log2- transformed). Default is "distance".
`CHR`	Controls which chromosome to predict boundaries on at base-level resolution. Required.
`chromCoords`	List containing the starting bp coordinate and ending bp coordinate that defines the region of the linear genome to make predictions on. If chromCoords is not specified, then predictions will be made on the entire chromosome. Default is NULL.
`tadModel`	Model object used to obtain predicted probabilities at base-level resolution (examples include `gbm`, `glmnet`, `svm`, `glm`, etc). For a random forest model, can be obtained using `preciseTAD::randomForest`). Required.
`threshold`	Bases with predicted probabilities that are greater than or equal to this value are labeled as potential TAD boundaries. Values in the range of .95-1.0 are suggested. Default is 1. If a vector of different values is passed to the threshold paramenter then a grid is provided with eps (see DBSCAN_params parameter) threshold values and the optimal combination is chose as the combination that maximizes normalized enrichment – calculated as the number of peak regions that overlap with flanked predicted boundaries divided by the total number of predicted boundaries, averaged over all predictors (genomic elements).
`verbose`	Option to print progress. Default is TRUE.
`parallel`	Option to parallelise the process for obtaining predicted probabilities. Must be number to indicate the number of cores to use in parallel. Default is NULL.
`DBSCAN_params`	Parameters passed to `dbscan` in list form containing 1) eps and 2) MinPts. If a vector of different values is passed to eps then a grid is provided with the probability threshold (see threshold parameter) values and the optimal combination is chose as the combination that maximizes normalized enrichment – calculated as the number of peak regions that overlap with flanked predicted boundaries divided by the total number of predicted boundaries, averaged over all predictors (genomic elements). Required.
`flank`	Controls how much to flank the predicted TAD boundaries for calculating normalized enrichment. Required.
`BaseProbs`	Option to include the vector of probabilities for each base-level coordinate. Recommended to be used only when chromCoords is specified.

Value

A list containing 4 elements including: 1) data frame with average (and standard deviation) normalized enrichment (NE) values for each combination of t and eps (only if multiple values are provided for at least paramenter; all subsequent summaries are applied to optimal combination of (t, eps)), 2) the genomic coordinates spanning each preciseTAD predicted region (PTBR), 3) the genomic coordinates of preciseTAD predicted boundaries points (PTBP), 4) a named list including summary statistics of the following: PTBRWidth - PTBR width, PTBRCoverage - the proportion of bases within a PTBR with probabilities that equal to or exceed the threshold (t=1 by default), DistanceBetweenPTBR - the genomic distance between the end of the previous PTBR and the start of the subsequent PTBR, NumSubRegions - the number of the subregions in each PTBR cluster, SubRegionWidth - the width of the subregion forming each PTBR, DistBetweenSubRegions - the genomic distance between the end of the previous PTBR-specific subregion and the start of the subsequent PTBR-specific subregion, NormilizedEnrichment - the normalized enrichment of the genomic annotations used in the model around flanked PTBPs, and BaseProbs - a numeric vector of probabilities for each corresponding base coordinate.

Examples

```
# Read in ARROWHEAD-called TADs at 5kb
data(arrowhead_gm12878_5kb)

# Extract unique boundaries
bounds.GR <- extractBoundaries(domains.mat = arrowhead_gm12878_5kb,
                               preprocess = FALSE,
                               CHR = c("CHR21", "CHR22"),
                               resolution = 5000)

# Read in GRangesList of 26 TFBS and filter to include only CTCF, RAD21,
#SMC3, and ZNF143
data(tfbsList)

tfbsList_filt <- tfbsList[which(names(tfbsList) %in%
                                                 c("Gm12878-Ctcf-Broad",
                                                   "Gm12878-Rad21-Haib",
                                                   "Gm12878-Smc3-Sydh",
                                                   "Gm12878-Znf143-Sydh"))]

# Create the binned data matrix for CHR1 (training) and CHR22 (testing)
# using 5 kb binning, distance-type predictors from 4 TFBS from
# the GM12878 cell line, and random under-sampling
set.seed(123)
tadData <- createTADdata(bounds.GR = bounds.GR,
                         resolution = 5000,
                         genomicElements.GR = tfbsList_filt,
                         featureType = "distance",
                         resampling = "rus",
                         trainCHR = "CHR21",
                         predictCHR = "CHR22")

# Perform random forest using TADrandomForest by tuning mtry over 10 values
# using 3-fold CV
set.seed(123)
tadModel <- TADrandomForest(trainData = tadData[[1]],
                            testData = tadData[[2]],
                            tuneParams = list(mtry = 2,
                                              ntree = 500,
                                              nodesize = 1),
                            cvFolds = 3,
                            cvMetric = "Accuracy",
                            verbose = TRUE,
                            model = TRUE,
                            importances = TRUE,
                            impMeasure = "MDA",
                            performances = TRUE)

# Apply preciseTAD on a specific 2mb section of CHR22:17000000-18000000
set.seed(123)
pt <- preciseTAD(genomicElements.GR = tfbsList_filt,
                 featureType = "distance",
                 CHR = "CHR22",
                 chromCoords = list(17000000, 18000000),
                 tadModel = tadModel[[1]],
                 threshold = c(0.975, 0.99, 1.0),
                 verbose = TRUE,
                 parallel = NULL,
                 DBSCAN_params = list(c(5000, 10000,15000,20000,30000), 3),
                 flank = 5000)
```

---

signal_func *Helper function used to create signal type feature space*

Description

Helper function used to create signal type feature space

Usage

```
signal_func(binned_data_gr, annot_data_gr)
```

Arguments

binned_data_gr	A GRanges object
annot_data_gr	A GRanges object

Value

A vector of intensities indicating the signal strength within each overlap

`TADrandomForest`	*A wrapper function passed to* `caret::train` *to apply a random forest classification algorithm built and tested on user-defined binned domain data from* `createTADdata`.

Description

A wrapper function passed to `caret::train` to apply a random forest classification algorithm built and tested on user-defined binned domain data from `createTADdata`.

Usage

```
TADrandomForest(
  trainData,
  testData = NULL,
 tuneParams = list(mtry = ceiling(sqrt(ncol(trainData) - 1)), ntree = 500, nodesize =
    1),
  cvFolds = 3,
  cvMetric = "Accuracy",
  verbose = FALSE,
  model = TRUE,
  importances = TRUE,
  impMeasure = "MDA",
  performances = FALSE
)
```

Arguments

`trainData`	Data frame, the binned data matrix to built a random forest classifiers (can be obtained using `createTADdata`). Required.
`testData`	Data frame, the binned data matrix to test random forest classifiers (can be obtained using `createTADdata`). The first column must be a factor with positive class "Yes". Default is NULL in which case no performances are evaluated.
`tuneParams`	List, providing `mtry`, `ntree`, and `nodesize` parameters to feed into `randomForest`. Default is list(mtry = ceiling(sqrt(ncol(trainData) - 1)), ntree = 500, nodesize = 1). If multiple values are provided, then a grid search is performed to tune the model. Required.
`cvFolds`	Numeric, number of k-fold cross-validation to perform in order to tune the hyperparameters. Required.
`cvMetric`	Character, performance metric to use to choose optimal tuning parameters (one of either "Kappa", "Accuracy", "MCC", "ROC", "Sens", "Spec", "Pos Pred Value", "Neg Pred Value"). Default is "Accuracy".
`verbose`	Logical, controls whether or not details regarding modeling should be printed out. Default is TRUE.
`model`	Logical, whether to keep the model object. Default is TRUE.

`importances`	Logical, whether to extract variable importances. Default is TRUE.
`impMeasure`	Character, indicates the variable importance measure to use (one of either "MDA" (mean decrease in accuracy) or "MDG" (mean decrease in gini)). Ignored if importances = FALSE.
`performances`	Logical, indicates whether various performance metrics should be extracted when validating the model on the test data. Ignored if testData = NULL.

Value

A list containing: 1) a train object from `caret` with model information, 2) a data.frame of variable importance for each feature included in the model, and 3) a data.frame of various performance metrics

Examples

```
# Read in ARROWHEAD-called TADs at 5kb
data(arrowhead_gm12878_5kb)

# Extract unique boundaries
bounds.GR <- extractBoundaries(domains.mat = arrowhead_gm12878_5kb,
                               preprocess = FALSE,
                               CHR = c("CHR21", "CHR22"),
                               resolution = 5000)

# Read in GRangesList of 26 TFBS
data(tfbsList)

# Create the binned data matrix for CHR1 (training) and CHR22 (testing)
# using 5 kb binning, distance-type predictors from 26 different TFBS from
# the GM12878 cell line, and random under-sampling
tadData <- createTADdata(bounds.GR = bounds.GR,
                         resolution = 5000,
                         genomicElements.GR = tfbsList,
                         featureType = "distance",
                         resampling = "rus",
                         trainCHR = "CHR21",
                         predictCHR = "CHR22")

# Perform random forest using TADrandomForest by tuning mtry over 10 values
# using 3-fold CV
tadModel <- TADrandomForest(trainData = tadData[[1]],
                            testData = tadData[[2]],
                            tuneParams = list(mtry = c(2,5,8,10,13,16,18,21,24,26),
                                              ntree = 500,
                                              nodesize = 1),
                            cvFolds = 3,
                            cvMetric = "Accuracy",
                            verbose = TRUE,
                            model = TRUE,
                            importances = TRUE,
                            impMeasure = "MDA",
                            performances = TRUE)
```

`TADrfe`	*A wrapper function passed to* `caret::rfe` *to apply recursive feature elimination (RFE) on binned domain data as a feature reduction technique for random forests. Backward elimination is performed from p down to 2, by powers of 2, where p is the number of features in the data.*

Description

A wrapper function passed to `caret::rfe` to apply recursive feature elimination (RFE) on binned domain data as a feature reduction technique for random forests. Backward elimination is performed from p down to 2, by powers of 2, where p is the number of features in the data.

Usage

```
TADrfe(
  trainData,
  tuneParams = list(ntree = 500, nodesize = 1),
  cvFolds = 5,
  cvMetric = "Accuracy",
  verbose = FALSE
)
```

Arguments

`trainData`	Data frame, the binned data matrix to built a random forest classifiers (can be obtained using `createTADdata`). Required.
`tuneParams`	List, providing `ntree` and `nodesize` parameters to feed into `randomForest`. Required.
`cvFolds`	Numeric, number of k-fold cross-validation to perform. Required.
`cvMetric`	Character, performance metric to use to choose optimal tuning parameters (one of either "Kappa", "Accuracy", "MCC","ROC","Sens", "Spec", "Pos Pred Value", "Neg Pred Value"). Default is "Accuracy".
`verbose`	Logical, controls whether or not details regarding modeling should be printed out. Default is TRUE.

Value

A list containing: 1) the performances extracted at each of the k folds and, 2) Variable importances among the top features at each step of RFE. For 1) ‘Variables‘ - the best subset of features to consider at each iteration, ‘MCC‘ (Matthews Correlation Coefficient), ‘ROC‘ (Area under the receiver operating characteristic curve), ‘Sens‘ (Sensitivity), ‘Spec‘ (Specificity), ‘Pos Pred Value‘ (Positive predictive value), ‘Neg Pred Value‘ (Negative predictive value), ‘Accuracy‘, and the corresponding standard deviations across the cross-folds. For 2) ‘Overall‘ - the variable importance, ‘var‘ - the feature name, ‘Variables‘ - the number of features that were considered at each cross-fold, and ‘Resample‘ - the cross-fold

Examples

```
# Read in ARROWHEAD-called TADs at 5kb
data(arrowhead_gm12878_5kb)

#Extract unique boundaries
bounds.GR <- extractBoundaries(domains.mat = arrowhead_gm12878_5kb,
                               preprocess = FALSE,
                               CHR = "CHR22",
                               resolution = 5000)

# Read in GRangesList of 26 TFBS
data(tfbsList)

# Create the binned data matrix for CHR22 using:
# 5 kb binning,
# oc-type predictors from 26 different TFBS from the GM12878 cell line, and
# random under-sampling
tadData <- createTADdata(bounds.GR = bounds.GR,
                         resolution = 5000,
                         genomicElements.GR = tfbsList,
                         featureType = "oc",
                         resampling = "rus",
                         trainCHR = "CHR22",
                         predictCHR = NULL)

# Perform RFE for fully grown random forests with 100 trees using 5-fold CV
# Evaluate performances using accuracy
rfe_res <- TADrfe(trainData = tadData[[1]],
                  tuneParams = list(ntree = 100, nodesize = 1),
                  cvFolds = 5,
                  cvMetric = "Accuracy",
                  verbose = TRUE)
```

`tfbsList` *A list of the chromosomal coordinates for 26 transcription factor binding sites from the Gm12878 cell line*

Description

A GRangesList containing 26 GRanges objects each with the following elements

seqnames The chromosome number

ranges IRanges object with start and end coordinates for each TFBS

strand empty column

coverage a metadata column with peak strength values at each site

Usage

```
tfbsList
```

Format

An object of class `CompressedGRangesList` of length 26.

Value

A GrangeList

Source

Data obtained from ENCODE. ENCODE integrative analysis (PMID: 22955616; PMCID: PMC3439153) ENCODE portal (PMID: 29126249; PMCID: PMC5753278) Available at `https://www.encodeproject.org/`

Index