

# Package ‘SVM2CRM’

October 18, 2017

**Type** Package

**Title** SVM2CRM: support vector machine for cis-regulatory elements detections

**Version** 1.8.0

**Date** 2014-03-30

**Description** Detection of cis-regulatory elements using svm implemented in LiblineaR.

**Author** Guidantonio Malagoli Tagliacruzchi  
<malagoli@ingm.org>

**License** GPL-3

**Depends** R (>= 3.2.0), LiblineaR, SVM2CRMdata

**Imports** AnnotationDbi, mclust, GenomicRanges, IRanges, zoo, squash, pls, rtracklayer, ROCR, verification

**biocViews** ChIPSeq, SupportVectorMachine, Software, Preprocessing, ChipOnChip

**Maintainer** Guidantonio Malagoli Tagliacruzchi  
<guidantonio.malagolitagliacruzchi@unimore.it>

**NeedsCompilation** no

## R topics documented:

cisREfindbed . . . . .	2
createBed . . . . .	3
createSVMinput . . . . .	5
featSelectionWithKmeans . . . . .	6
findFeatureOverlap . . . . .	9
frequencyHM . . . . .	10
getSignal . . . . .	12
performanceSVM . . . . .	13
plotFscore . . . . .	15
plotROC . . . . .	17
predictionGW . . . . .	19
smoothInputFS . . . . .	22
tuningParametersCombROC . . . . .	23

<b>Index</b>	<b>26</b>
--------------	-----------

## Description

This function require as input a bed file in format chr, start, end, signalNorm. The genomic coordinates must be sort. The data should be normalized using others methods. For e.g the ChIP-seq data should be partitioned in 100bp and then normalize by the library size. This function simply load each bed file of one histone mark and then, partion the genome in n not overlapping windows of size (w). In particular, build a matrix where the number of columns for each histone mark depends on the size of the window and the bin size used during the preprocessing of the bed file. This function allow to smooth the signal of the bed file every n bin. The default function of smoothing is median. This is the suggested function to model the signal. Other function have not been tested.

## Usage

```
cisREfindbed(list_file,chr="chr1", bin.size, windows, window.smooth=200,smoothing="FALSE",function.smoothing)
```

## Arguments

list_file	The list of bed files (e.g. character)
chr	A vector containing the list of chromosomes to consider during the analysis
bin.size	The size of bin used to preprocess the bed files
windows	The size of the window (default=5000bp)
window.smooth	The size of the window to smooth the bin (default=200)
smoothing	logical, default is FALSE, if TRUE smooth the signal
function.smoothing	Set the function to smooth the signal (default is median)

## Details

Some detailed description

## Value

A data.frame where the number of columns depends on the windows size, bin.size and the number of histone marks considered for the prediction.

## Author(s)

Guidantonio Malagoli Tagliazucchi [guidantonio.malagolitagliazucchi@unimore.it](mailto:guidantonio.malagolitagliazucchi@unimore.it)

## See Also

`cisREfindbed`, `mclapply`

**Examples**

```

library(rtracklayer)
chr<-"chr1"
#the values use to binarized the genome (in bp)
bin.size<-100
#the size of the windows (in bp)
windows<-500
#the step of smoothing (in bp): e.g. smooth the signal every 200bp
window.smooth<-200
#do you want apply a smoothing
smoothing<-"FALSE"
#what kind of function do you want use to smooth the signal
function.smoothing<-"median"
#list of file: format bed
list_file<-"CD4-H3K14ac.norm.w100.bed"

#completeTABLE<-cisREfindbed(list_file[1],chr=chr,bin.size=bin.size,windows=windows>window.smooth=window.s

```

---

createBed

*Create bed file of predictions using svm.*


---

**Description**

With this function the user can create a bed files with the regions of predicted cis-regulatory elements. Internal function.

**Usage**

```
createBed(test_set, label1, pred, outputfile)
```

**Arguments**

test_set	test_set produced for svm model
label1	class1
pred	p\$prediction object
outputfile	name of bed file

**Details**

Some detailed description

**Value**

A bed files with the genomic coordinates of the cis-regulatory elements predicted using SVM2CRM.

**Author(s)**

Guidantonio Malagoli Tagliazucchi [guidantonio.malagolitagliazucchi@unimore.it](mailto:guidantonio.malagolitagliazucchi@unimore.it)

**See Also**

cisREfindbed

**Examples**

```

library("GenomicRanges")
library("SVM2CRMdata")

setwd(system.file("data", package="SVM2CRMdata"))
load("CD4_matrixInputSVBbin100window1000.rda")
completeTABLE<-CD4_matrixInputSVBbin100window1000

new.strings<-gsub(x=colnames(completeTABLE[,c(6:ncol(completeTABLE))]),pattern="CD4.",replacement="")
new.strings<-gsub(new.strings,pattern=".norm.w100.bed",replacement="")
colnames(completeTABLE)[c(6:ncol(completeTABLE))]<-new.strings

#list_file<-grep(dir(),pattern=".sort.txt",value=TRUE)

#train_positive<-getSignal(list_file,chr="chr1",reference="p300.distal.fromTSS.txt",win.size=500,bin.si
#train_negative<-getSignal(list_file,chr="chr1",reference="random.region.hg18.nop300.txt",win.size=500,
setwd(system.file("data", package="SVM2CRMdata"))
load("train_positive.rda")
load("train_negative.rda")

training_set<-rbind(train_positive,train_negative)
#the colnames of the training set should be the same of data_enhancer_svm
colnames(training_set)[c(5:ncol(training_set))]<-gsub(x=gsub(x=colnames(training_set[,c(5:ncol(training
setwd(system.file("extdata", package = "SVM2CRMdata"))
data_level2 <- read.table(file = "GSM393946.distal.p300fromTSS.txt",sep = "\t", stringsAsFactors = FALSE)
data_level2<-data_level2[data_level2[,1]=="chr1",]

DB <- data_level2[, c(1:3)]
colnames(DB)<-c("chromosome", "start", "end")

label <- "p300"

table.final.overlap<-findFeatureOverlap(query=completeTABLE, subject=DB, select="all")

data_enhancer_svm<-createSVMinput(inputpos=table.final.overlap, inputfull=completeTABLE, label1="enhancer
colnames(data_enhancer_svm)[c(5:ncol(data_enhancer_svm))]<-gsub(gsub(x=colnames(data_enhancer_svm[,c(5:
listcolnames<-c("H2AK5ac", "H2AK9ac", "H3K23ac", "H3K27ac", "H3K27me3", "H3K4me1", "H3K4me3")

dftotann<-smoothInputFS(train_positive[,c(6:ncol(train_positive))],listcolnames,k=20)

results<-featSelectionWithKmeans(dftotann,5)

resultsFS<-results[[7]]

resultsFSfilter<-resultsFS[which(resultsFS[,2]>median(resultsFS[,2])),]

resultsFSfilterICRR<-resultsFSfilter[which(resultsFSfilter[,3]<0.50),]

listHM<-resultsFSfilterICRR[,1]
listHM<-gsub(gsub(listHM,pattern="_.",replacement=""),pattern="CD4.",replacement="")

```

```

selectFeature<-grep(x=colnames(training_set[,c(6:ncol(training_set))]),pattern=paste(listHM,collapse="|"))

colSelect<-c("chromosome","start","end","label",selectFeature)
training_set<-training_set[,colSelect]

vecS <- c(2:length(listHM))
typeSVM <- c(0, 6, 7)[1]
costV <- c(0.001, 0.01, 0.1, 1, 10, 100, 1000)[6]
wlabel <- c("not_enhancer", "enhancer")
infofile<-data.frame(a=c(paste(listHM,"signal",sep=".")))
infofile[,1]<-gsub(gsub(x=infofile[,1],pattern="CD4.",replacement=""),pattern=".sort.bed",replacement="")

tuningTAB <- tuningParametersCombROC(training_set = training_set, typeSVM = typeSVM, costV = costV,differ

tuningTABfilter<-tuningTAB[tuningTAB$fscore<0.95,]
#row_max_fscore<-which.max(tuningTABfilter[tuningTABfilter$nHM >2,"fscore"])
row_max_fscore<-which.max(tuningTABfilter[, "fscore"])
listHM_prediction<-gsub(tuningTABfilter[row_max_fscore,4],pattern="//",replacement="|")

columnPR<-grep(colnames(training_set),pattern=paste(listHM_prediction,collapse="|"),value=TRUE)

predictionGW(training_set=training_set,data_enhancer_svm=data_enhancer_svm, listHM=columnPR,pcClass.str:

```

---

createSVMinput

*Take the output of findFeatureOverlap and then create a positive and negative set of cis-regulatory elements*

---

## Description

Take the output of findFeatureOverlap and then create a positive and negative set of cis-regulatory elements

## Usage

```
createSVMinput(inputpos,inputfull,label1,label2)
```

## Arguments

inputpos	output of findFeatureOverlap (see documentation findFeatureOverlap)
inputfull	output of cisREfindbed (see documentation cisREfindbed)
label1	a string to define the first class (e.g. enhancers)
label2	a string to define the second class (e.g. not_enhancers)

## Details

Some detailed description

## Value

A data.frame with the signals of the histone modifications for positive (e.g. enhancers) and negative (e.g. not\_enhancers) examples.

**Author(s)**

Guidantonio Malagoli Tagliazucchi [guidantonio.malagolitagliazucchi@unimore.it](mailto:guidantonio.malagolitagliazucchi@unimore.it)

**See Also**

`findFeatureOverlap`, `cisREfindbed`

**Examples**

```
library("SVM2CRMdata")
library("GenomicRanges")

setwd(system.file("data", package="SVM2CRMdata"))
load("CD4_matrixInputSVMbin100window1000.rda")
completeTABLE<-CD4_matrixInputSVMbin100window1000

new.strings<-gsub(x=colnames(completeTABLE[,c(6:ncol(completeTABLE))]),pattern="CD4.",replacement="")
new.strings<-gsub(new.strings,pattern=".norm.w100.bed",replacement="")
colnames(completeTABLE)[c(6:ncol(completeTABLE))]<-new.strings

#list_file<-grep(dir(),pattern=".sort.txt",value=TRUE)
#train_positive<-getSignal(list_file,chr="chr1",reference="p300.distal.fromTSS.txt",win.size=500,bin.size=500)
#train_negative<-getSignal(list_file,chr="chr1",reference="random.region.hg18.nop300.txt",win.size=500,bin.size=500)
setwd(system.file("data", package="SVM2CRMdata"))
load("train_positive.rda")
load("train_negative.rda")

training_set<-rbind(train_positive,train_negative)
colnames(training_set)[c(5:ncol(training_set))]<-gsub(x=gsub(x=colnames(training_set[,c(5:ncol(training_set))]),pattern="p300.distal.fromTSS.txt",replacement=""),pattern="random.region.hg18.nop300.txt",replacement="")

setwd(system.file("extdata", package="SVM2CRMdata"))
data_level2 <- read.table(file = "GSM393946.distal.p300fromTSS.txt", sep = "\t", stringsAsFactors = FALSE)
data_level2<-data_level2[data_level2[,1]=="chr1",]

DB <- data_level2[, c(1:3)]
colnames(DB)<-c("chromosome", "start", "end")

label <- "p300"

table.final.overlap<-findFeatureOverlap(query=completeTABLE,subject=DB)
data_enhancer_svm<-createSVMinput(inputpos=table.final.overlap,inputfull=completeTABLE,label1="enhancer")
```

---

featSelectionWithKmeans

*This function select the most meaningful variables in a matrix of ChIP-seq data using k-means and ICRR.*

---

**Description**

In the research of promoters and enhancers the users start with large dataset of histone modification. The number of histone marks is a variable that influence the prediction of enhancers. Several papers investigated what is the best combination of histone marks to predict enhancers. There are not a

consensus about the optimal number of histone mark in the prediction of cis-regulatory elements. Moreover, in the genome there are a number of histone marks that is greater than 50. The biological roles of all of these is not already clear. However from the computational aspect the admit of a huge number of variables (histone marks) can insert redundant information (signal of histone marks) during the step of prediction. Therefore here we introduced a step of features selection. The step of variable selection consists of two steps. In the first one the function performs a smoothing of the signals of each histone mark. For example, if the data were binned every 100bp, a window size of smoothing equal to 2 means that the signal of the histone mark is smoothed every 200 bp. In the second step, the function using a  $k$  number of clusters defined by the user gathers the signal of each histone mark and estimates the mean of signals inside each group. Finally, it estimates the median of all means above computed and all histone marks with a signal less than this are filtered out. We introduced also the concept of index coverage of the regulatory regions (ICRR). In the first step the function measures the total coverage of the cis-regulatory elements in the list of positive class and considering  $w$ , the window size used to model the signal of the histone marks around a particular feature (positive class, negative class). Then it inspects in those enhancers the signals ( $S$ ) of a particular histone mark is less or greater than the global mean ( $M$ ) of the signal ( $S$ ). Next the function computes the coverage only for these two fractions of enhancers and estimates the ICRR. This value assumes values from 0 to 1. A value close to 0 means that the difference between the coverage of the two classes of enhancers is little, in contrast, if this value is close to 1 this means that there is a diversity between the two fractions of the cis-regulatory elements. The ICRR values can be used to select the histone marks to use during the analysis. Next using `tuningParametersCombROC` the user can tune the best optimal set of svm parameters and histone marks. The function `featSelectionWithKmeans` returns a list containing the results of this analysis. The first element contains a matrix with the results of the selection analysis, the parameters used during the analysis and the filtered histone marks. Finally this function creates a report with three plots. In the first one the x-axis contains the labels of the histone marks while in y-axis reports the median of mean of each group. The second plot represents the same thing but using a scatterplot. The third page contains the same graph in the first page and a plot that contains the index of coverage between enhancers and non-enhancer regions.

## Usage

```
featSelectionWithKmeans(dfTotann, nk, autoK="no", w=1000, gmax=7, outputplot="feature.selection.pdf")
```

## Arguments

<code>dfTotann</code>	a data.frame created with <code>smoothInputFS</code>
<code>nk</code>	number of cluster to use for k-means algorithm
<code>autoK</code>	logical, if <code>autoK=yes</code> , the function estimates automatically the number of cluster to use during k-means ( <code>autoK="no"</code> )
<code>w</code>	the window size used to build the model (default=1000bp)
<code>gmax</code>	the parameter <code>gmax</code> used during the analysis ( <code>gmax=7</code> )
<code>outputplot</code>	the name of the report pdf file (default="feature.selection.pdf")

## Details

input: the results of `smoothInputFS`. This function clusters every column (histone marks) using a  $K$  number of clusters user defined. Next the mean of the signals in each group are estimated. The function `featSelectionWithKmeans` returns a list containing the results of this analysis. The first element contains a matrix with the results of the selection analysis, the parameters used during the analysis and the filtered histone marks. Finally this function creates a report with three plots. In the first one the x-axis contains the labels of the histone marks while in y-axis reports the median

of mean of each group. The second plot represent the same thing but using a scatterplot. The third page contain the same graph in the first page and a plot that contain the index of coverage between enhancers and not enhancers regions. Definition of k-clusters: The user can set manually the parameter k otherwise featSelectionWithKmeans implement automatically the investigation of k using a Bayesian Information Criterion for EM initialized.

### Value

- a list where the first element is the matrix of output of feature selection analysis - mom: the vertical mean of the matrix in the element 1 - nk: number of clusters used during the clustering - gmax: the parameter gmax used during the analysis - which histone marks have a the signal that is greater than the median global signal - which histone marks have a signal where the histone are greater than the mean of the global signal. - a data.frame where the first column contain the results of feature selection analysis with k-means and in the second column the results of ICRR (index coverage regulatory regions).

### Author(s)

Guidantonio Malagoli Tagliacruzchi [guidantonio.malagolitagliacruzchi@unimore.it](mailto:guidantonio.malagolitagliacruzchi@unimore.it)

### See Also

`cisREfindbed`, `smoothinputFS`, `tuningParametersCombROC`

### Examples

```
library("SVM2CRMdata")
library("GenomicRanges")

setwd(system.file("data", package="SVM2CRMdata"))
load("CD4_matrixInputSVMbin100window1000.rda")
completeTABLE<-CD4_matrixInputSVMbin100window1000

new.strings<-gsub(x=colnames(completeTABLE[,c(6:ncol(completeTABLE))]),pattern="CD4.",replacement="")
new.strings<-gsub(new.strings,pattern=".norm.w100.bed",replacement="")
colnames(completeTABLE)[c(6:ncol(completeTABLE))]<-new.strings

#list_file<-grep(dir(),pattern=".sort.txt",value=TRUE)
#train_positive<-getSignal(list_file,chr="chr1",reference="p300.distal.fromTSS.txt",win.size=500,bin.size=100)
#train_negative<-getSignal(list_file,chr="chr1",reference="random.region.hg18.nop300.txt",win.size=500,bin.size=100)
load("train_positive.rda")
load("train_negative.rda")

training_set<-rbind(train_positive,train_negative)

training_set<-rbind(train_positive,train_negative)
colnames(training_set)[c(5:ncol(training_set))]<-gsub(x=gsub(x=colnames(training_set[,c(5:ncol(training_set))]),pattern="CD4.",replacement=""),pattern=".norm.w100.bed",replacement="")

setwd(system.file("extdata", package = "SVM2CRMdata"))
data_level2 <- read.table(file = "GSM393946.distal.p300fromTSS.txt", sep = "\t", stringsAsFactors = FALSE)
data_level2<-data_level2[data_level2[,1]=="chr1",]

DB <- data_level2[, c(1:3)]
colnames(DB)<-c("chromosome", "start", "end")
```



```

label <- "p300"

table.final.overlap<-findFeatureOverlap(query=completeTABLE,subject=DB,select="all")

data_enhancer_svm<-createSVMinput(inputpos=table.final.overlap,inputfull=completeTABLE,label1="enhancer",
colnames(data_enhancer_svm)[c(5:ncol(data_enhancer_svm))]<-gsub(gsub(x=colnames(data_enhancer_svm[,c(5:
ncol(data_enhancer_svm)]),c(5:ncol(data_enhancer_svm)]),c(5:ncol(data_enhancer_svm)]),c(5:ncol(data_enhancer_svm)]),c(5:ncol(data_enhancer_svm)]))

listcolnames<-c("H2AK5ac","H2AK9ac","H3K23ac","H3K27ac","H3K27me3","H3K4me1","H3K4me3")

dftotann<-smoothInputFS(train_positive[,c(6:ncol(train_positive))],listcolnames,k=20)

results<-featSelectionWithKmeans(dftotann,5)

```

---

findFeatureOverlap	<i>Find the overlap of genomic regions between the output of cisREfind and a database of validated cis-regulatory elements.</i>
--------------------	---

---

### Description

Find the overlap of genomic regions between the output of cisREfind and a database of validated cis-regulatory elements.

### Usage

```
findFeatureOverlap(query, subject, select)
```

### Arguments

query	database of validated cis-regulatory elements (e.g. p300 binding sites), GRanges class.
subject	output cisREfind function, GRanges class.
select	parameters of findOverlaps

### Details

See documentation of GenomicRanges for details.

### Value

A data frame with the genomic regions that overlap

### Author(s)

Guidantonio Malagoli Tagliazucchi [guidantonio.malagolitagliazucchi@unimore.it](mailto:guidantonio.malagolitagliazucchi@unimore.it)

### See Also

cisREfind, findOverlaps, GenomicRanges

**Examples**

```

library("SVM2CRMdata")
library("GenomicRanges")

setwd(system.file("data", package="SVM2CRMdata"))
load("CD4_matrixInputSVBbin100window1000.rda")
completeTABLE<-CD4_matrixInputSVBbin100window1000
new.strings<-gsub(x=colnames(completeTABLE[,c(6:ncol(completeTABLE))]),pattern="CD4.",replacement="")
new.strings<-gsub(new.strings,pattern=".norm.w100.bed",replacement="")
colnames(completeTABLE)[c(6:ncol(completeTABLE))]<-new.strings

setwd(system.file("extdata", package="SVM2CRMdata"))
data_level2 <- read.table(file = "GSM393946.distal.p300fromTSS.txt", sep = "\t", stringsAsFactors = FALSE)
data_level2<-data_level2[data_level2[,1]=="chr1",]

DB <- data_level2[, c(1:3)]
colnames(DB)<-c("chromosome", "start", "end")

label <- "p300"

table.final.overlap<-findFeatureOverlap(query=completeTABLE, subject=DB)

```

---

frequencyHM

*frequencyHM*


---

**Description**

Compute the frequency of the histone marks after tuning of parameters (tuningParametersComb)

**Usage**

```
frequencyHM(tuningTABfilter)
```

**Arguments**

tuningTABfilter

A data.frame from tuningParametersComb

**Details**

Some detailed description

**Value**

A vector containing the frequencies of the histone marks

**Author(s)**

Guidantonio Malagoli Tagliazucchi [guidantonio.malagolitagliazucchi@unimore.it](mailto:guidantonio.malagolitagliazucchi@unimore.it)

**See Also**

cisREfind, mclapply

**Examples**

```

library("GenomicRanges")
library("SVM2CRMdata")

setwd(system.file("data", package="SVM2CRMdata"))
load("CD4_matrixInputSVMbin100window1000.rda")
completeTABLE<-CD4_matrixInputSVMbin100window1000

new.strings<-gsub(x=colnames(completeTABLE[,c(6:ncol(completeTABLE))]),pattern="CD4.",replacement="")
new.strings<-gsub(new.strings,pattern=".norm.w100.bed",replacement="")
colnames(completeTABLE)[c(6:ncol(completeTABLE))]<-new.strings

#list_file<-grep(dir(),pattern=".sort.txt",value=TRUE)
#train_positive<-getSignal(list_file,chr="chr1",reference="p300.distal.fromTSS.txt",win.size=500,bin.size=100)
#train_negative<-getSignal(list_file,chr="chr1",reference="random.region.hg18.nop300.txt",win.size=500,bin.size=100)
setwd(system.file("data", package="SVM2CRMdata"))
load("train_positive.rda")
load("train_negative.rda")

training_set<-rbind(train_positive,train_negative)
#the colnames of the training set should be the same of data_enhancer_svm
colnames(training_set)[c(5:ncol(training_set))]<-gsub(x=gsub(x=colnames(training_set[,c(5:ncol(training_set))]),pattern="p300.",replacement=""),pattern="p300.",replacement="")

setwd(system.file("extdata", package = "SVM2CRMdata"))
data_level2 <- read.table(file = "GSM393946.distal.p300fromTSS.txt",sep = "\t", stringsAsFactors = FALSE)
data_level2<-data_level2[data_level2[,1]=="chr1",]

DB <- data_level2[, c(1:3)]
colnames(DB)<-c("chromosome","start","end")

label <- "p300"

table.final.overlap<-findFeatureOverlap(query=completeTABLE,subject=DB,select="all")

data_enhancer_svm<-createSVMinput(inputpos=table.final.overlap,inputfull=completeTABLE,label1="enhancers")
colnames(data_enhancer_svm)[c(5:ncol(data_enhancer_svm))]<-gsub(gsub(x=colnames(data_enhancer_svm[,c(5:ncol(data_enhancer_svm))]),pattern="p300.",replacement=""),pattern="p300.",replacement="")

listcolnames<-c("H2AK5ac","H2AK9ac","H3K23ac","H3K27ac","H3K27me3","H3K4me1","H3K4me3")

dftotann<-smoothInputFS(train_positive[,c(6:ncol(train_positive))],listcolnames,k=20)

results<-featSelectionWithKmeans(dftotann,5)

resultsFS<-results[[7]]

resultsFSfilter<-resultsFS[which(resultsFS[,2]>median(resultsFS[,2])),]

resultsFSfilterICRR<-resultsFSfilter[which(resultsFSfilter[,3]<0.50),]

```

```

listHM<-resultsFSfilterICRR[,1]
listHM<-gsub(gsub(listHM,pattern="_.",replacement=""),pattern="CD4.",replacement="")

selectFeature<-grep(x=colnames(training_set[,c(6:ncol(training_set))]),pattern=paste(listHM,collapse="|"))

colSelect<-c("chromosome","start","end","label",selectFeature)
training_set<-training_set[,colSelect]

vecS <- c(2:length(listHM))
typeSVM <- c(0, 6, 7)[1]
costV <- c(0.001, 0.01, 0.1, 1, 10, 100, 1000)[6]
wlabel <- c("not_enhancer", "enhancer")
infofile<-data.frame(a=c(paste(listHM,"signal",sep=".")))
infofile[,1]<-gsub(gsub(x=infofile[,1],pattern="CD4.",replacement=""),pattern=".sort.bed",replacement="|")

tuningTAB <- tuningParametersCombROC(training_set = training_set, typeSVM = typeSVM, costV = costV,difference=0.05)

tuningTABfilter<-tuningTAB[tuningTAB$fscore<0.95,]

frequencyHM<-frequencyHM(tuningTABfilter)

```

---

getSignal

*Model the signals of each histone marks around genomic features (e.g. enhancers, not\_enhancers).*


---

## Description

This function simply model the signal of each histone marks around the features used in the input files and considering the bin.size and window size defined during the pre-processing step.

## Usage

```
getSignal.bedfilelist,chr,reference,win.size,bin.size,label1="enhancers")
```

## Arguments

bedfilelist	test_set produced for svm model
chr	a vector containin the list of chromosome that you want use during the analysis (e.g."chr1")
reference	file with the reference position of the features. The genomic coordinates of positive and negative examples (e.g. enhancers, not_enhancers)
win.size	windows size used to smooth the signal
bin.size	original bin size used
label1	class of reference (e.g. enhancers or not_enhancers)

## Details

Some detailed description

**Value**

A data.frame with the signals where in the column there are the signals of the histone marks and in the rows the cis-regulatory elements.

**Author(s)**

Guidantonio Malagoli Tagliacruzchi guidantonio.malagolitagliacruzchi@unimore.it

**See Also**

cisREfindbed

**Examples**

```
library("SVM2CRMdata")
library("GenomicRanges")

setwd(system.file("data", package="SVM2CRMdata"))
load("CD4_matrixInputSVMbin100window1000.rda")
completeTABLE<-CD4_matrixInputSVMbin100window1000

new.strings<-gsub(x=colnames(completeTABLE[,c(6:ncol(completeTABLE))]),pattern="CD4.",replacement="")
new.strings<-gsub(new.strings,pattern=".norm.w100.bed",replacement="")
colnames(completeTABLE)[c(6:ncol(completeTABLE))]<-new.strings

#list_file<-grep(dir(),pattern=".sort.txt",value=TRUE)
#train_positive<-getSignal(list_file,chr="chr1",reference="p300.distal.fromTSS.txt",win.size=500,bin.si
#train_negative<-getSignal(list_file,chr="chr1",reference="random.region.hg18.nop300.txt",win.size=500,
setwd(system.file("data", package="SVM2CRMdata"))
load("train_positive.rda")
load("train_negative.rda")
training_set<-rbind(train_positive,train_negative)
colnames(training_set)[c(5:ncol(training_set))]<-gsub(x=gsub(x=colnames(training_set[,c(5:ncol(training
```

---

performanceSVM

*Estimate the performance of prediction.*

---

**Description**

This function use a contingency table and then estimate: sensitivity, false positive rate, accuracy, specificity, precision, positive predicted values, negative predicted values, false discovery rates, f-score. Useful to estimate the performance of a model. Internal function of plotROC.

**Usage**

```
performanceSVM(res)
```

**Arguments**

res                    a confusion matrix

**Details**

Some detailed description

**Value**

A data.frame with tpr.sensitivity, fpr, acc, spc.specificity, precision, ppv, npv, fdr, fscore.

**Author(s)**

Guidantonio Malagoli Tagliacruzchi [guidantonio.malagolitagliacruzchi@unimore.it](mailto:guidantonio.malagolitagliacruzchi@unimore.it)

**See Also**

[cisREfind](#), [tuningParametersCOmbROC](#), [predictionGW](#)

**Examples**

```
library("GenomicRanges")
library("SVM2CRMdata")

setwd(system.file("data", package="SVM2CRMdata"))
load("CD4_matrixInputSVMbin100window1000.rda")
completeTABLE<-CD4_matrixInputSVMbin100window1000

new.strings<-gsub(x=colnames(completeTABLE[,c(6:ncol(completeTABLE))]), pattern="CD4.", replacement="")
new.strings<-gsub(new.strings, pattern=".norm.w100.bed", replacement="")
colnames(completeTABLE)[c(6:ncol(completeTABLE))]<-new.strings

#list_file<-grep(dir(), pattern=".sort.txt", value=TRUE)

#train_positive<-getSignal(list_file, chr="chr1", reference="p300.distal.fromTSS.txt", win.size=500, bin.size=500)
#train_negative<-getSignal(list_file, chr="chr1", reference="random.region.hg18.nop300.txt", win.size=500, bin.size=500)
setwd(system.file("data", package="SVM2CRMdata"))
load("train_positive.rda")
load("train_negative.rda")

training_set<-rbind(train_positive, train_negative)
#the colnames of the training set should be the same of data_enhancer_svm
colnames(training_set)[c(5:ncol(training_set))]<-gsub(x=gsub(x=colnames(training_set[,c(5:ncol(training_set))]), pattern="chr1", replacement=""), pattern="chr1", replacement="")

setwd(system.file("extdata", package="SVM2CRMdata"))
data_level2 <- read.table(file = "GSM393946.distal.p300fromTSS.txt", sep = "\t", stringsAsFactors = FALSE)
data_level2<-data_level2[data_level2[,1]=="chr1",]

DB <- data_level2[, c(1:3)]
colnames(DB)<-c("chromosome", "start", "end")

label <- "p300"

table.final.overlap<-findFeatureOverlap(query=completeTABLE, subject=DB, select="all")

data_enhancer_svm<-createSVMinput(inputpos=table.final.overlap, inputfull=completeTABLE, label1="enhancer")
colnames(data_enhancer_svm)[c(5:ncol(data_enhancer_svm))]<-gsub(gsub(x=colnames(data_enhancer_svm[,c(5:ncol(data_enhancer_svm))]), pattern="chr1", replacement=""), pattern="chr1", replacement="")
```

```

listcolnames<-c("H2AK5ac", "H2AK9ac", "H3K23ac", "H3K27ac", "H3K27me3", "H3K4me1", "H3K4me3")
dftotann<-smoothInputFS(train_positive[,c(6:ncol(train_positive))],listcolnames,k=20)

results<-featSelectionWithKmeans(dftotann,5)

resultsFS<-results[[7]]

resultsFSfilter<-resultsFS[which(resultsFS[,2]>median(resultsFS[,2])),]

resultsFSfilterICRR<-resultsFSfilter[which(resultsFSfilter[,3]<0.50),]

listHM<-resultsFSfilterICRR[,1]
listHM<-gsub(gsub(listHM,pattern="_.",replacement=""),pattern="CD4.",replacement="")

selectFeature<-grep(x=colnames(training_set[,c(6:ncol(training_set))]),pattern=paste(listHM,collapse="|"))

colSelect<-c("chromosome", "start", "end", "label", selectFeature)
training_set<-training_set[,colSelect]

vecS <- c(2:length(listHM))
typeSVM <- c(0, 6, 7)[1]
costV <- c(0.001, 0.01, 0.1, 1, 10, 100, 1000)[6]
wlabel <- c("not_enhancer", "enhancer")
infofile<-data.frame(a=c(paste(listHM,"signal",sep=".")))
infofile[,1]<-gsub(gsub(x=infofile[,1],pattern="CD4.",replacement=""),pattern=".sort.bed",replacement="")

tuningTAB <- tuningParametersCombROC(training_set = training_set, typeSVM = typeSVM, costV = costV,differ

```

---

plotFscore

*Plot the F-score in relation with the sensitivity and specificity*

---

### Description

Plot the ROC curve of the best model using cross-fold validation

### Usage

```
plotFscore(tuningTAB)
```

### Arguments

tuningTAB      the output of tuningParametersComb.R

### Details

See documentation tuningParametersCombROC, performanceSVM

### Value

A pdf with the k-cross-correlation plots for: 1) k-cross-validation, 2) ROC horizontal 3) ROC vertical

**Author(s)**

Guidantonio Malagoli Tagliacruzchi [guidantonio.malagolitagliacruzchi@unimore.it](mailto:guidantonio.malagolitagliacruzchi@unimore.it)

**See Also**

`cisREfindbed`, `tuningParametersCombROC`, `performanceSVM`

**Examples**

```

library("GenomicRanges")
library("SVM2CRMdata")

setwd(system.file("data", package="SVM2CRMdata"))
load("CD4_matrixInputSVMbin100window1000.rda")
completeTABLE<-CD4_matrixInputSVMbin100window1000

new.strings<-gsub(x=colnames(completeTABLE[,c(6:ncol(completeTABLE))]),pattern="CD4.",replacement="")
new.strings<-gsub(new.strings,pattern=".norm.w100.bed",replacement="")
colnames(completeTABLE)[c(6:ncol(completeTABLE))]<-new.strings

list_file<-grep(dir(),pattern=".sort.txt",value=TRUE)

#train_positive<-getSignal(list_file,chr="chr1",reference="p300.distal.fromTSS.txt",win.size=500,bin.size=100)
#train_negative<-getSignal(list_file,chr="chr1",reference="random.region.hg18.nop300.txt",win.size=500,bin.size=100)
setwd(system.file("data", package="SVM2CRMdata"))
load("train_positive.rda")
load("train_negative.rda")
training_set<-rbind(train_positive,train_negative)
#the colnames of the training set should be the same of data_enhancer_svm
colnames(training_set)[c(5:ncol(training_set))]<-gsub(x=gsub(x=colnames(training_set[,c(5:ncol(training_set))]),pattern="p300",replacement=""),pattern="p300",replacement="")

setwd(system.file("extdata", package = "SVM2CRMdata"))
data_level2 <- read.table(file = "GSM393946.distal.p300fromTSS.txt",sep = "\t", stringsAsFactors = FALSE)
data_level2<-data_level2[data_level2[,1]=="chr1",]

DB <- data_level2[, c(1:3)]
colnames(DB)<-c("chromosome","start","end")

label <- "p300"

table.final.overlap<-findFeatureOverlap(query=completeTABLE,subject=DB,select="all")

data_enhancer_svm<-createSVMinput(inputpos=table.final.overlap,inputfull=completeTABLE,label1="enhancer")
colnames(data_enhancer_svm)[c(5:ncol(data_enhancer_svm))]<-gsub(gsub(x=colnames(data_enhancer_svm[,c(5:ncol(data_enhancer_svm))]),pattern="p300",replacement=""),pattern="p300",replacement="")

listcolnames<-c("H2AK5ac","H2AK9ac","H3K23ac","H3K27ac","H3K27me3","H3K4me1","H3K4me3")

dftotann<-smoothInputFS(train_positive[,c(6:ncol(train_positive))],listcolnames,k=20)

results<-featSelectionWithKmeans(dftotann,5)

resultsFS<-results[[7]]

```



```

resultsFSfilter<-resultsFS[which(resultsFS[,2]>median(resultsFS[,2])),]
resultsFSfilterICRR<-resultsFSfilter[which(resultsFSfilter[,3]<0.50),]

listHM<-resultsFSfilterICRR[,1]
listHM<-gsub(gsub(listHM,pattern="_.",replacement=""),pattern="CD4.",replacement="")

selectFeature<-grep(x=colnames(training_set[,c(6:ncol(training_set))]),pattern=paste(listHM,collapse="|"))

colSelect<-c("chromosome","start","end","label",selectFeature)
training_set<-training_set[,colSelect]

vecS <- c(2:length(listHM))
typeSVM <- c(0, 6, 7)[1]
costV <- c(0.001, 0.01, 0.1, 1, 10, 100, 1000)[6]
wlabel <- c("not_enhancer", "enhancer")
infofile<-data.frame(a=c(paste(listHM,"signal",sep=".")))
infofile[,1]<-gsub(gsub(x=infofile[,1],pattern="CD4.",replacement=""),pattern=".sort.bed",replacement="")

tuningTAB <- tuningParametersCombROC(training_set = training_set, typeSVM = typeSVM, costV = costV,different.weight=TRUE)

pdf("FSCORE_distribution.pdf")
plotFscore(tuningTAB)
dev.off()

```

---

plotROC

*Plot the ROC curve of the best model*


---

## Description

Plot the ROC curve of the best model using cross-fold validation. Internal function of tuningParametersCombROC and predictionGW.

## Usage

```
plotROC(datatrain,k,y,different.weight,type,cost,output)
```

## Arguments

datatrain	training set (data.frame)
y	a vector with the list of labels
k	number of iteration for k-fold cross validation default values is 5.
output	name of output file.
different.weight	specify if the classes are unbalanced.TRUE
type	type of kernel to use
cost	parameter cost of SVM

## Details

Some detailed description

**Value**

A pdf with the k-cross-correlation plots for: 1) k-cross-validation, 2) ROC horizontal 3) ROC vertical

**Author(s)**

Guidantonio Malagoli Tagliacruzchi guidantonio.malagolitagliacruzchi@unimore.it

**See Also**

cisREfindbed, tuningParametersCombROC, perftuningParametersCombROC, performanceSVM

**Examples**

```
library("GenomicRanges")
library("SVM2CRMdata")

setwd(system.file("data", package="SVM2CRMdata"))
load("CD4_matrixInputSVMbin100window1000.rda")
completeTABLE<-CD4_matrixInputSVMbin100window1000

new.strings<-gsub(x=colnames(completeTABLE[,c(6:ncol(completeTABLE))]),pattern="CD4.",replacement="")
new.strings<-gsub(new.strings,pattern=".norm.w100.bed",replacement="")
colnames(completeTABLE)[c(6:ncol(completeTABLE))]<-new.strings

#list_file<-grep(dir(),pattern=".sort.txt",value=TRUE)

#train_positive<-getSignal(list_file,chr="chr1",reference="p300.distal.fromTSS.txt",win.size=500,bin)
#train_negative<-getSignal(list_file,chr="chr1",reference="random.region.hg18.nop300.txt",win.size=500,bin)
setwd(system.file("data", package="SVM2CRMdata"))
load("train_positive.rda")
load("train_negative.rda")

training_set<-rbind(train_positive,train_negative)
#the colnames of the training set should be the same of data_enhancer_svm
colnames(training_set)[c(5:ncol(training_set))]<-gsub(x=gsub(x=colnames(training_set),c(5:ncol(training_set))),"p300",label))

setwd(system.file("extdata", package = "SVM2CRMdata"))
data_level2 <- read.table(file = "GSM393946.distal.p300fromTSS.txt", sep = "\t", stringsAsFactors = FALSE)
data_level2<-data_level2[data_level2[,1]=="chr1",]

DB <- data_level2[, c(1:3)]
colnames(DB)<-c("chromosome","start","end")

label <- "p300"

table.final.overlap<-findFeatureOverlap(query=completeTABLE,subject=DB,select="all")

data_enhancer_svm<-createSVMinput(inputpos=table.final.overlap,inputfull=completeTABLE,label1="enhancer")
colnames(data_enhancer_svm)[c(5:ncol(data_enhancer_svm))]<-gsub(gsub(x=colnames(data_enhancer_svm),c(5:ncol(data_enhancer_svm))),"p300",label))

listcolnames<-c("H2AK5ac","H2AK9ac","H3K23ac","H3K27ac","H3K27me3","H3K4me1","H3K4me3")

dftotann<-smoothInputFS(train_positive[,c(6:ncol(train_positive))],listcolnames,k=20)
```

```

results<-featSelectionWithKmeans(dftotann,5)

resultsFS<-results[[7]]

resultsFSfilter<-resultsFS[which(resultsFS[,2]>median(resultsFS[,2])),]

resultsFSfilterICRR<-resultsFSfilter[which(resultsFSfilter[,3]<0.50),]

listHM<-resultsFSfilterICRR[,1]
listHM<-gsub(gsub(listHM,pattern="_.",replacement=""),pattern="CD4.",replacement="")

selectFeature<-grep(x=colnames(training_set[,c(6:ncol(training_set))]),pattern=paste(listHM,collapse=" "))

colSelect<-c("chromosome", "start", "end", "label", selectFeature)
training_set<-training_set[,colSelect]

vecS <- c(2:length(listHM))
typeSVM <- c(0, 6, 7)[1]
costV <- c(0.001, 0.01, 0.1, 1, 10, 100, 1000)[6]
wlabel <- c("not_enhancer", "enhancer")
infofile<-data.frame(a=c(paste(listHM,"signal",sep=".")))
infofile[,1]<-gsub(gsub(x=infofile[,1],pattern="CD4.",replacement=""),pattern=".sort.bed",replacement="")

tuningTAB <- tuningParametersCombROC(training_set = training_set, typeSVM = typeSVM, costV = costV, dif

tuningTABfilter<-tuningTAB[tuningTAB$fscore<0.95,]
#row_max_fscore<-which.max(tuningTABfilter[tuningTABfilter$nHM >2, "fscore"])
row_max_fscore<-which.max(tuningTABfilter[, "fscore"])
listHM_prediction<-gsub(tuningTABfilter[row_max_fscore,4],pattern="//",replacement="|")

columnPR<-grep(colnames(training_set),pattern=paste(listHM_prediction,collapse="|"),value=TRUE)

predictionGW(training_set=training_set,data_enhancer_svm=data_enhancer_svm, listHM=columnPR,pcClass.string="enhancer",nClass.string="no")

```

---

predictionGW

*Perform prediction of cis-regulatory elements genome-wide*

---

### Description

This function perform the prediction genome-wide of the cis-regulatory elements. The function return the output of performanceSVM and a bed file with the position of enhancers and not enhancers regions.

### Usage

```
predictionGW(training_set,data_enhancer_svm,listHM,pcClass.string="enhancer",nClass.string="no")
```

### Arguments

training\_set    training set (data.frame)

<code>data_enhancer_svm</code>	the signals of all histone marks along the genome
<code>listHM</code>	a vector with the histone marks that you want to use perform the prediction
<code>pcClass.string</code>	label of the first class (e.g. "enhancer")
<code>nClass.string</code>	label of the second class (e.g. "not_enhancers")
<code>pcClass</code>	number of positive class in the test set
<code>ncClass</code>	number of negative class in the test set
<code>cost</code>	parameter of svm (default=100)
<code>type</code>	type of kernel (default=0)
<code>output.file</code>	name of the bed file of output

### Details

The ratio between the positive and negative regions usually is 1:10. However this ratio depends on you experimental design and your data. See documentation `cisREfindbed`, `tuningParametersCom-bROC`, `featSelectionWithKmeans`.

### Value

The performance of prediction and a bed file with the coordinates of genomic regions that contain the enhancers. The bed file is saved in the directory selected by the user.

### Author(s)

Guidantonio Malagoli Tagliacruzchi [guidantonio.malagolitagliacruzchi@unimore.it](mailto:guidantonio.malagolitagliacruzchi@unimore.it)

### See Also

`cisREfindbed`, `mclapply`

### Examples

```
library("GenomicRanges")
library("SVM2CRMdata")

setwd(system.file("data", package="SVM2CRMdata"))
load("CD4_matrixInputSVMbin100window1000.rda")
completeTABLE<-CD4_matrixInputSVMbin100window1000

new.strings<-gsub(x=colnames(completeTABLE[,c(6:ncol(completeTABLE))]), pattern="CD4.", replacement="")
new.strings<-gsub(new.strings, pattern=".norm.w100.bed", replacement="")
colnames(completeTABLE)[c(6:ncol(completeTABLE))]<-new.strings

#list_file<-grep(dir(), pattern=".sort.txt", value=TRUE)

#train_positive<-getSignal(list_file, chr="chr1", reference="p300.distal.fromTSS.txt", win.size=500, bin.si
#train_negative<-getSignal(list_file, chr="chr1", reference="random.region.hg18.nop300.txt", win.size=500,
setwd(system.file("data", package="SVM2CRMdata"))
load("train_positive.rda")
load("train_negative.rda")

training_set<-rbind(train_positive, train_negative)
#the colnames of the training set should be the same of data_enhancer_svm
```

```

colnames(training_set)[c(5:ncol(training_set))]<-gsub(x=gsub(x=colnames(training_set[,c(5:ncol(training

setwd(system.file("extdata", package = "SVM2CRMdata"))
data_level2 <- read.table(file = "GSM393946.distal.p300fromTSS.txt", sep = "\t", stringsAsFactors = FALSE)
data_level2<-data_level2[data_level2[,1]=="chr1",]

DB <- data_level2[, c(1:3)]
colnames(DB)<-c("chromosome", "start", "end")

label <- "p300"

table.final.overlap<-findFeatureOverlap(query=completeTABLE, subject=DB, select="all")

data_enhancer_svm<-createSVMinput(inputpos=table.final.overlap, inputfull=completeTABLE, label1="enhancer",
colnames(data_enhancer_svm)[c(5:ncol(data_enhancer_svm))]<-gsub(gsub(x=colnames(data_enhancer_svm[,c(5:

listcolnames<-c("H2AK5ac", "H2AK9ac", "H3K23ac", "H3K27ac", "H3K27me3", "H3K4me1", "H3K4me3")

dftotann<-smoothInputFS(train_positive[,c(6:ncol(train_positive))], listcolnames, k=20)

results<-featSelectionWithKmeans(dftotann, 5)

resultsFS<-results[[7]]

resultsFSfilter<-resultsFS[which(resultsFS[,2]>median(resultsFS[,2])),]

resultsFSfilterICRR<-resultsFSfilter[which(resultsFSfilter[,3]<0.50),]

listHM<-resultsFSfilterICRR[,1]
listHM<-gsub(gsub(listHM, pattern="_.", replacement=""), pattern="CD4.", replacement="")

selectFeature<-grep(x=colnames(training_set[,c(6:ncol(training_set))]), pattern=paste(listHM, collapse="|")

colSelect<-c("chromosome", "start", "end", "label", selectFeature)
training_set<-training_set[,colSelect]

vecS <- c(2:length(listHM))
typeSVM <- c(0, 6, 7)[1]
costV <- c(0.001, 0.01, 0.1, 1, 10, 100, 1000)[6]
wlabel <- c("not_enhancer", "enhancer")
infofile<-data.frame(a=c(paste(listHM, "signal", sep=".")))
infofile[,1]<-gsub(gsub(x=infofile[,1], pattern="CD4.", replacement=""), pattern=".sort.bed", replacement="")

tuningTAB <- tuningParametersCombROC(training_set = training_set, typeSVM = typeSVM, costV = costV, differ

tuningTABfilter<-tuningTAB[tuningTAB$fscore<0.95,]
#row_max_fscore<-which.max(tuningTABfilter[tuningTABfilter$nHM >2, "fscore"])
row_max_fscore<-which.max(tuningTABfilter[, "fscore"])
listHM_prediction<-gsub(tuningTABfilter[row_max_fscore,4], pattern="//", replacement="|")

columnPR<-grep(colnames(training_set), pattern=paste(listHM_prediction, collapse="|"), value=TRUE)

predictionGW(training_set=training_set, data_enhancer_svm=data_enhancer_svm, listHM=columnPR, pcClass.str:

```

---

smoothInputFS	<i>Smooth the signals of the histone marks to prepare the input for feature selection analysis</i>
---------------	--

---

### Description

Give the matrix obtained using `getSignal` this functions smooth the signals of each histone marks using a particular window (if `bin=100`). To size of smooth is `bin*k` (e.g. a parameter `k` equal to 2 means that the signal is smooth every 200bp).

### Usage

```
smoothInputFS(input_ann,k,listcolnames)
```

### Arguments

<code>input_ann</code>	the data.frame with the training set
<code>k</code>	the size of smooth in bp
<code>listcolnames</code>	the names of column in which perform the smoothing. A vector with the list of histone marks.

### Details

The smoothing is performed using the median

### Value

A data.frame with the smoothed signals of histone marks

### Author(s)

Guidantonio Malagoli Tagliacruzchi [guidantonio.malagolitagliacruzchi@unimore.it](mailto:guidantonio.malagolitagliacruzchi@unimore.it)

### See Also

`cisREfindbed`, `featSelectionWithKmeans`, `tuningParametersCombROC`

### Examples

```
library("GenomicRanges")
library("SVM2CRMdata")

setwd(system.file("data", package="SVM2CRMdata"))
load("CD4_matrixInputSVMbin100window1000.rda")
completeTABLE<-CD4_matrixInputSVMbin100window1000

new.strings<-gsub(x=colnames(completeTABLE[,c(6:ncol(completeTABLE))]),pattern="CD4.",replacement="")
new.strings<-gsub(new.strings,pattern=".norm.w100.bed",replacement="")
colnames(completeTABLE)[c(6:ncol(completeTABLE))]<-new.strings
```

```

#list_file<-grep(dir(),pattern=".sort.txt",value=TRUE)

#train_positive<-getSignal(list_file,chr="chr1",reference="p300.distal.fromTSS.txt",win.size=500,bin.size=500)
#train_negative<-getSignal(list_file,chr="chr1",reference="random.region.hg18.nop300.txt",win.size=500,bin.size=500)
setwd(system.file("data",package="SVM2CRMdata"))
load("train_positive.rda")
load("train_negative.rda")

training_set<-rbind(train_positive,train_negative)
colnames(training_set)[c(5:ncol(training_set))]<-gsub(x=gsub(x=colnames(training_set)[c(5:ncol(training_set))],value="chr1"),value="")

setwd(system.file("extdata", package = "SVM2CRMdata"))
data_level2 <- read.table(file = "GSM393946.distal.p300fromTSS.txt",sep = "\t", stringsAsFactors = FALSE)
data_level2<-data_level2[data_level2[,1]=="chr1",]

DB <- data_level2[, c(1:3)]
colnames(DB)<-c("chromosome","start","end")

label <- "p300"

table.final.overlap<-findFeatureOverlap(query=completeTABLE,subject=DB,select="all")

data_enhancer_svm<-createSVMinput(inputpos=table.final.overlap,inputfull=completeTABLE,label1="enhancers",label2="p300")
colnames(data_enhancer_svm)[c(5:ncol(data_enhancer_svm))]<-gsub(gsub(x=colnames(data_enhancer_svm)[c(5:ncol(data_enhancer_svm))],value="chr1"),value="")

listcolnames<-c("H2AK5ac","H2AK9ac","H3K23ac","H3K27ac","H3K27me3","H3K4me1","H3K4me3")

dftotann<-smoothInputFS(train_positive[,c(6:ncol(train_positive))],listcolnames,k=20)

```

---

tuningParametersCombROC

*Test different models using different kernel of SVM, values of cost functions, the number of histone marks.*

---

## Description

The function tuningParametersCombROC allow high flexibility: the user can set the type of kernel, the cost parameter of SVM, the number of histone marks. tuningParametersCombROC use performanceSVM and the output is a data.frame where for each model there are the parameters compute with performanceSVM. To help the user to discriminate how to discriminate the best model SVM2CRM implement several function to plot the results of performanceSVM.

## Usage

```
tuningParametersCombROC(training_set, typeSVM, costV,different.weight=TRUE, vecS,infofile,pcClass)
```

## Arguments

training_set	training set (data.frame)
typeSVM	a vector with the kinds of Kernel

costV	a vector with a list of cost parameters
different.weight	the data are unbalanced (default TRUE)
vecS	a vector with the number of histone marks (e.g. from 2 to x)
infofile	a data.frame where in the column "a" there are the histone marks, while in the column "b" a vectors of letters.
pcClass.string	label of the first class (e.g. "enhancers")
nClass.string	label of the second class (e.g. "not_enhancers")
pcClass	number of positive class in the training set
ncClass	number of negative class in the training set

**Details**

Some detailed description

**Value**

A data.frame with the values from performanceSVM for each trained model.

**Author(s)**

Guidantonio Malagoli Tagliazucchi [guidantonio.malagolitagliazucchi@unimore.it](mailto:guidantonio.malagolitagliazucchi@unimore.it)

**See Also**

`cisREfindbed`, `performanceSVM`, `plotFscore`, `plotROC`

**Examples**

```
library("GenomicRanges")
library("SVM2CRMdata")

setwd(system.file("data", package="SVM2CRMdata"))
load("CD4_matrixInputSVMbin100window1000.rda")
completeTABLE<-CD4_matrixInputSVMbin100window1000

new.strings<-gsub(x=colnames(completeTABLE[,c(6:ncol(completeTABLE))]),pattern="CD4.",replacement="")
new.strings<-gsub(new.strings,pattern=".norm.w100.bed",replacement="")
colnames(completeTABLE)[c(6:ncol(completeTABLE))]<-new.strings

#Create a vector that contain the list of the bed files that you want use during the analysis
#list_file<-grep(dir(),pattern=".sort.txt",value=TRUE)
#print(list_file)
#Here we used a data.frame that contain the genomic coordinates of p300 binding sites
#train_positive<-getSignal(list_file,chr="chr1",reference="p300.distal.fromTSS.txt",win.size=500,bin.size=500)
#train_negative<-getSignal(list_file,chr="chr1",reference="random.region.hg18.nop300.txt",win.size=500,bin.size=500)
setwd(system.file("data", package="SVM2CRMdata"))
load("train_positive.rda")
load("train_negative.rda")

training_set<-rbind(train_positive,train_negative)
colnames(training_set)[c(5:ncol(training_set))]<-gsub(x=gsub(x=colnames(training_set[,c(5:ncol(training_set))])
```



```

setwd(system.file("extdata", package = "SVM2CRMdata"))
data_level2 <- read.table(file = "GSM393946.distal.p300fromTSS.txt", sep = "\t", stringsAsFactors = FALSE)
data_level2<-data_level2[data_level2[,1]=="chr1",]

DB <- data_level2[, c(1:3)]
colnames(DB)<-c("chromosome", "start", "end")

label <- "p300"

table.final.overlap<-findFeatureOverlap(query=completeTABLE, subject=DB, select="all")

data_enhancer_svm<-createSVMinput(inputpos=table.final.overlap, inputfull=completeTABLE, label1="enhancers")
colnames(data_enhancer_svm)[c(5:ncol(data_enhancer_svm))]<-gsub(gsub(x=colnames(data_enhancer_svm[,c(5:ncol(data_enhancer_svm))]), pattern="H2AK5ac", replacement="H2AK9ac"), pattern="H2AK9ac", replacement="H2AK5ac")

listcolnames<-c("H2AK5ac", "H2AK9ac", "H3K23ac", "H3K27ac", "H3K27me3", "H3K4me1", "H3K4me3")

dftotann<-smoothInputFS(train_positive[,c(6:ncol(train_positive))], listcolnames, k=20)

results<-featSelectionWithKmeans(dftotann, 5)

resultsFS<-results[[7]]

resultsFSfilter<-resultsFS[which(resultsFS[,2]>median(resultsFS[,2])),]

resultsFSfilterICRR<-resultsFSfilter[which(resultsFSfilter[,3]<0.50),]

listHM<-resultsFSfilterICRR[,1]
listHM<-gsub(gsub(listHM, pattern="_.", replacement=""), pattern="CD4.", replacement="")

selectFeature<-grep(x=colnames(training_set[,c(6:ncol(training_set))]), pattern=paste(listHM, collapse="|"))

colSelect<-c("chromosome", "start", "end", "label", selectFeature)
training_set<-training_set[,colSelect]

vecS <- c(2:length(listHM))
typeSVM <- c(0, 6, 7)[1]
costV <- c(0.001, 0.01, 0.1, 1, 10, 100, 1000)[6]
wlabel <- c("not_enhancer", "enhancer")
infofile<-data.frame(a=c(paste(listHM, "signal", sep=".")))
infofile[,1]<-gsub(gsub(x=infofile[,1], pattern="CD4.", replacement=""), pattern=".sort.bed", replacement="")

tuningTAB <- tuningParametersCombROC(training_set = training_set, typeSVM = typeSVM, costV = costV, differe

```

# Index

- \*Topic **F-score, sensitivity, specificity, performance**
    - plotFscore, [15](#)
  - \*Topic **create bed, save output, results**
    - createBed, [3](#)
  - \*Topic **features selection, smooth signal, histone marks, reduction of complexity**
    - smoothInputFS, [22](#)
  - \*Topic **featureselection, k-means, relevant variables, ICRR**
    - featSelectionWithKmeans, [6](#)
  - \*Topic **frequencyHM, countHM, histonemarks**
    - frequencyHM, [10](#)
  - \*Topic **f**
    - plotROC, [17](#)
  - \*Topic **inputSVM, positive examples, negative examples, overlap**
    - createSVMinput, [5](#)
  - \*Topic **overlap, positive examples, negative examples**
    - findFeatureOverlap, [9](#)
  - \*Topic **performanceSVM**
    - performanceSVM, [13](#)
  - \*Topic **prediction, saveoutput, createbed**
    - predictionGW, [19](#)
  - \*Topic **preprocessing**
    - cisREfindbed, [2](#)
  - \*Topic **training-set, positive examples, negative examples, input**
    - getSignal, [12](#)
  - \*Topic **tuningparameters, brute-force, cost, type, SVM, models, combination, histone-marks**
    - tuningParametersCombROC, [23](#)
- 
- cisREfindbed, [2](#)
  - createBed, [3](#)
  - createSVMinput, [5](#)
  
  - featSelectionWithKmeans, [6](#)
  - findFeatureOverlap, [9](#)
  
  - frequencyHM, [10](#)
  - getSignal, [12](#)
  
  - performanceSVM, [13](#)
  - performanceSVM (performanceSVM), [13](#)
  - plotFscore, [15](#)
  - plotROC, [17](#)
  - predictionGW, [19](#)
  
  - smoothInputFS, [22](#)
  
  - tuningParametersCombROC, [23](#)
- 
- cisREfindbed, [2](#)
  - createBed, [3](#)
  - createSVMinput, [5](#)
  
  - featSelectionWithKmeans, [6](#)
  - findFeatureOverlap, [9](#)