

Package ‘LncFinder’

February 6, 2017

Type Package

Title Long Non-Coding RNA Identification Based on Features of
Sequence, EIIP and Secondary Structure

Version 1.0.0

Author Han Siyu [aut, cre],
Li Ying [aut],
Liang Yanchun [aut]

Maintainer Han Siyu <hansy15@mails.jlu.edu.cn>

Acknowledgments Cheng Ming, Fan Linrui, Guo Yuan, Li Yaolong

Description Functions for predicting sequences are mRNAs or long non-coding RNAs.
Default models are trained on human, mouse and wheat datasets by employing
SVM. Features are based on intrinsic composition of sequence, EIIP value
(electron-ion interaction pseudopotential) and secondary structure. The model
can also be built on users' own data.

License GPL-3

Depends R (>= 3.2.3), seqinr (>= 3.1-3), e1071 (>= 1.6-7), caret (>= 6.0-71), parallel (>= 3.1.0)

LazyData true

RoxygenNote 6.0.0

NeedsCompilation no

Repository CRAN

Date/Publication 2017-02-06 14:30:20

R topics documented:

build_model	2
demo_dataset	4
demo_DNA.seq	5
demo_SS.seq	5
extract_features	6
lnc_finder	8

make_frequencies	10
max_orf	13
run_RNAfold	14
svm_tune	15

Index	18
--------------	-----------

build_model	<i>Build Users' Own Model</i>
-------------	-------------------------------

Description

This function is used to build new models with users' own data.

Usage

```
build_model(lncRNA.seq, mRNA.seq, frequencies.file, SS.features = FALSE,
  lncRNA.format = "DNA", mRNA.format = "DNA", parallel.cores = 2,
  folds.num = 10, seed = 1, gamma.range = (2^seq(-5, 0, 1)),
  cost.range = c(1, 4, 8, 16, 24, 32))
```

Arguments

lncRNA.seq	Long non-coding sequences. Can be a FASTA file loaded by package "seqinr" (seqinr-package) or secondary structure sequences file (Dot-Bracket Notation) obtained from function run_RNAfold . If lncRNA.seq is secondary structure sequences file, parameter lncRNA.format should be defined as "SS".
mRNA.seq	mRNA sequences. FASTA file loaded by package "seqinr" or secondary structure sequences (Dot-Bracket Notation) obtained from function run_RNAfold . If mRNA.seq is secondary structure sequences file, parameter mRNA.format should be defined as "SS".
frequencies.file	String or a list obtained from function make_frequencies . Input species name "human", "mouse" or "wheat" to use prebuild frequencies files. Or assign a users' own frequencies file (Please refer to function make_frequencies for more information).
SS.features	Logical. If SS.features = TRUE, secondary structure features will be used to build the model. In this case, lncRNA.seq and mRNA.seq should be secondary structure sequences (Dot-Bracket Notation) obtained from function run_RNAfold and parameter lncRNA.format and mRNA.format should be set as "SS".
lncRNA.format	String. Define the format of lncRNA.seq. "DNA" for DNA sequences and "SS" for secondary structure sequences. Only when both mRNA.format and lncRNA.format are set as "SS", can the model with secondary structure features be built (SS.features = TRUE).
mRNA.format	String. Define the format of mRNA.seq. Can be "DNA" or "SS". "DNA" for DNA sequences and "SS" for secondary structure sequences. When this parameter is defined as "DNA", only the model without secondary structure features can be built. In this case, parameter SS.features should be set as FALSE.


```

cost.range = c(2, 6, 12, 20))

### Users can use default values of gamma.range and cost.range to find the
best parameters.
### Use your own frequencies file by assigning frequencies list to parameter
### "frequencies.file".

## End(Not run)

```

demo_dataset

A demo of dataset

Description

This dataset contains the features of 20 lncRNA sequences and 20 protein-coding sequences.

Usage

```
demo_dataset
```

Format

A data frame with 40 rows and 20 variables:

Label the class of the sequences

ORF.Max.Len the length of the longest ORF

ORF.Max.Cov the coverage of the longest ORF

Seq.lnc.dist Log-Distance.lncRNA

Seq.pct.dist Log-Distance.protein-coding transcripts

Seq.dist.ratio Distance-Ratio.sequence

Signal.Peak Signal as 1/3 position

SNR Signal to noise ratio

Signal.Min the minimum value of the top 10% power spectrum

Signal.Q1 the quantile Q1 of the top 10% power spectrum

Signal.Q2 the quantile Q2 of the top 10% power spectrum

Signal.Max the maximum value of the top 10% power spectrum

Dot.lnc.dist Log-Distance.acguD.lncRNA

Dot.pct.dist Log-Distance.acguD.protein-coding transcripts

Dot.dist.ratio Distance-Ratio.acguD

SS.lnc.dist Log-Distance.acgu-ACGU.lncRNA

SS.pct.dist Log-Distance.acgu-ACGU.protein-coding transcripts

SS.dist.ratio Distance-Ratio.acgu-ACGU

MFE Minimum free energy

UP.PCT Percentage of Unpair-Pair

Source

Sequences are selected from GENCODE.

`demo_DNA.seq`*A demo of DNA sequences*

Description

This file contains 10 DNA sequences.

Usage`demo_DNA.seq`**Format**

A list contains 10 DNA sequences.

The sequences are loaded by function `read.fasta`.

Source

DNA sequences are selected from GENCODE.

`demo_SS.seq`*A demo of secondary structure sequences*

Description

This file contains 10 SS (Secondary Structure) sequences.

Usage`demo_SS.seq`**Format**

A data frame with 3 rows and 10 variables:

The first row is RNA sequence; the second row is Dot-Bracket Notation of secondary structure sequences; the last row is minimum free energy (MFE).

Source

DNA sequences are selected from GENCODE. Secondary struture of each sequence is obtained from program "RNAfold".

extract_features	<i>Extract the Features.</i>
------------------	------------------------------

Description

This function can construct the dataset. This function is only used to extract the features, please use function [build_model](#) to build new models.

Usage

```
extract_features(Sequences, Label = NULL, SS.features = FALSE,
  format = "DNA", frequencies.file = "human", parallel.cores = 2)
```

Arguments

Sequences	mRNA sequences or long non-coding sequences. Can be a FASTA file loaded by package "seqinr" (seqinr-package) or secondary structure sequences (Dot-Bracket Notation) obtained from function run_RNAfold . If Sequences are secondary structure sequences file, parameter format should be defined as "SS".
Label	Optional. String. Indicate the label of the sequences such as "NonCoding", "Coding".
SS.features	Logical. If SS.features = TRUE, secondary structure features will be extracted. In this case, Sequences should be secondary structure sequences (Dot-Bracket Notation) obtained from function run_RNAfold and parameter format should be set as "SS".
format	String. Can be "DNA" or "SS". Define the format of Sequences. "DNA" for DNA sequences and "SS" for secondary structure sequences. This parameter must be set as "SS" when SS.features = TRUE.
frequencies.file	String or a list obtained from function make_frequencies . Input species name "human", "mouse" or "wheat" to use prebuild frequencies files. Or assign a users' own frequencies file (See function make_frequencies).
parallel.cores	Integer. The number of cores for parallel computation. By default the number of cores is 2. Users can set as -1 to run this function with all cores.

Details

This function extracts the features and constructs the dataset.

Considering that it is time consuming to obtain secondary structure sequences, users can build the model only with features of sequence and EIIP (SS.features = FALSE). When SS.features = TRUE, Sequences should be secondary structure sequences (Dot-Bracket Notation) obtained from function [run_RNAfold](#) and parameter format should be set as "SS".

Please note that:

Secondary structure features (SS.features) can improve the performance when the species of unevaluated sequences is identical to the species of the sequences that used to build the model.

However, if users are trying to predict sequences with the model trained on other species, `SS.features = TRUE` may lead to low accuracy.

Value

Returns a data.frame. 11 features when `SS.features = FALSE` and 19 features when `SS.features = TRUE`.

Features

1. Features based on sequence:

The length and coverage of the longest ORF (`ORF.Max.Len` and `ORF.Max.Cov`);

`Log-Distance.lncRNA (Seq.lnc.Dist)`;

`Log-Distance.protein-coding transcripts (Seq.pct.Dist)`;

`Distance-Ratio.sequence (Seq.Dist.Ratio)`.

2. Features based on EIIP (electron-ion interaction pseudopotential) value:

Signal as 1/3 position (`Signal.Peak`);

Signal to noise ratio (`SNR`);

the minimum value of the top 10% power spectrum (`Signal.Min`);

the quantile Q1 and Q2 of the top 10% power spectrum (`Signal.Q1` and `Signal.Q2`)

the maximum value of the top 10% power spectrum (`Signal.Max`).

3. Features based on secondary structure sequence:

`Log-Distance.acguD.lncRNA (Dot.lnc.dist)`;

`Log-Distance.acguD.protein-coding transcripts (Dot.pct.dist)`;

`Distance-Ratio.acguD (Dot.Dist.Ratio)`;

`Log-Distance.acgu-ACGU.lncRNA (SS.lnc.dist)`;

`Log-Distance.acgu-ACGU.protein-coding transcripts (SS.pct.dist)`;

`Distance-Ratio.acgu-ACGU (SS.Dist.Ratio)`;

Minimum free energy (`MFE`);

Percentage of Unpair-Pair (`UP.PCT`)

References

HAN Siyu, LIANG Yanchun and LI Ying*. LncFinder: Long Non-coding RNA Identification Tool Based on Features of Sequence Intrinsic Composition, Secondary Structure and EIIP Values. (2017) (*Submitted*)

Author(s)

Han Siyu

See Also

[svm_tune](#), [build_model](#), [make_frequencies](#), [run_RNAfold](#).

Examples

```
data(demo_DNA.seq)
Seqs <- demo_DNA.seq

### Extract features with prebuild frequencies.file:
my_features <- extract_features(Seqs, Label = "Class.of.the.Sequences",
                               SS.features = FALSE, format = "DNA",
                               frequencies.file = "mouse",
                               parallel.cores = 2)

### Use your own frequencies file by assign frequencies list to parameter
### "frequencies.file".
```

Inc_finder

Long Non-coding RNA Identification

Description

This function is used to predict sequences are non-coding transcripts or protein-coding transcripts.

Usage

```
Inc_finder(Sequences, SS.features = FALSE, format = "DNA",
           frequencies.file = "human", svm.model = "human", parallel.cores = 2)
```

Arguments

Sequences	Unevaluated sequences. Can be a FASTA file loaded by package "seqinr" (seqinr-package) or secondary structure sequences (Dot-Bracket Notation) obtained from function run_RNAfold . If Sequences is secondary structure sequences file, parameter format should be defined as "SS".
SS.features	Logical. If SS.features = TRUE, secondary structure features will be used. In this case, Sequences should be secondary structure sequences (Dot-Bracket Notation) obtained from function run_RNAfold and parameter format should be set as "SS".
format	String. Define the format of the Sequences. Can be "DNA" or "SS". "DNA" for DNA sequences and "SS" for secondary structure sequences. This parameter must be set as "SS" when SS.features = TRUE.
frequencies.file	String or a list obtained from function make_frequencies . Input species name "human", "mouse" or "wheat" to use prebuild frequencies files. Or assign a users' own frequencies file (See function make_frequencies).
svm.model	String or a svm model obtained from function build_model or svm_tune . Input species name "human", "mouse" or "wheat" to use prebuild models. Or assign a users' own model (See function build_model).
parallel.cores	Integer. The number of cores for parallel computation. By default the number of cores is 2. Users can set as -1 to run this function with all cores.

Details

Considering that it is time consuming to obtain secondary structure sequences, users can input nucleotide sequences and predict these sequences without secondary structure features (Set `SS.features` as `FALSE`).

Please note that:

`SS.features` can improve the performance when the species of unevaluated sequences is identical to the species of the sequences that used to build the model.

However, if users are trying to predict sequences with the model trained on other species, `SS.features` may lead to low accuracy.

For the details of `frequencies.file`, please refer to function [make_frequencies](#).

For the details of the features, please refer to function [extract_features](#).

Value

Returns a `data.frame`. Including the results of prediction (`Pred`); coding potential (`Coding.Potential`) and the features. For the details of the features, please refer to function [extract_features](#).

References

HAN Siyu, LIANG Yanchun and LI Ying*. lncFinder: Long Non-coding RNA Identification Tool Based on Features of Sequence Intrinsic Composition, Secondary Structure and EIIP Values. (2017) (*Submitted*)

Author(s)

Han Siyu

See Also

[build_model](#), [make_frequencies](#), [extract_features](#), [run_RNAfold](#).

Examples

```
data(demo_DNA.seq)
Seqs <- demo_DNA.seq

### Input one sequence:
OneSeq <- Seqs[1]
result_1 <- lnc_finder(OneSeq, SS.features = FALSE, format = "DNA",
                      frequencies.file = "human", svm.model = "human",
                      parallel.cores = 2)

## Not run:
### Or several sequences:
data(demo_SS.seq)
Seqs <- demo_SS.seq
result_2 <- lnc_finder(Seqs, SS.features = TRUE, format = "SS",
                      frequencies.file = "mouse", svm.model = "mouse",
                      parallel.cores = 2)
```

```

### A complete work flow:
### Calculate second structure on Windows OS,
RNAfold.path <- '"E:/Program Files/ViennaRNA/RNAfold.exe"'
SS.seq <- run_RNAfold(Seqs, RNAfold.path = RNAfold.path, parallel.cores = 2)

### Predict the sequences with secondary structure features,
result_2 <- lnc_finder(SS.seq, SS.features = TRUE, format = "SS",
                      frequencies.file = "mouse", svm.model = "mouse",
                      parallel.cores = 2)

### Predict sequences with your own model by assigning a new svm.model and
### frequencies.file to parameters "svm.model" and "frequencies.file"

## End(Not run)

```

make_frequencies	<i>Make the frequencies file with your own dataset</i>
------------------	--

Description

This function is used to calculate the frequencies of CDs and secondary structure sequences. This function is useful when users are trying to build their own model.

Usage

```

make_frequencies(cds.seq, mRNA.seq, lncRNA.seq, SS.features = FALSE,
                 cds.format = "DNA", lnc.format = "DNA", check.cds = TRUE,
                 ignore.illegal = TRUE)

```

Arguments

cds.seq	Coding sequences (mRNA without UTRs). Can be a FASTA file loaded by package "seqinr" (seqinr-package) or secondary structure sequences (Dot-Bracket Notation) obtained from function run_RNAfold . CDs are used to calculate hexamer frequencies of nucleotide sequences, thus secondary structure is not needed. Parameter cds.format should be "SS" when input is secondary structure sequences. (See details for more information.)
mRNA.seq	mRNA sequences with Dot-Bracket Notation. The secondary structure sequences can be obtained from function run_RNAfold . mRNA sequences are used to calculate the frequencies of acgu-ACGU and a acguD (see details), thus, mRNA sequences are required only when SS.features = TRUE.
lncRNA.seq	Long non-coding RNA sequences. Can be a FASTA file loaded by package "seqinr" (seqinr-package) or secondary structure sequences (Dot-Bracket Notation) obtained from function run_RNAfold . If SS.features = TRUE, lncRNA.seq must be RNA sequences with secondary structure sequences and parameter lnc.format should be defined as "SS".

SS.features	Logical. If SS.features = TRUE, frequencies of secondary structure will also be calculated and the model can be built with secondary structure features. In this case, mRNA.seq and lncRNA.seq should be secondary structure sequences.
cds.format	String. Define the format of the sequences of cds.seq. Can be "DNA" or "SS". "DNA" for DNA sequences and "SS" for secondary structure sequences.
lnc.format	String. Define the format of lncRNAs (lncRNA.seq). Can be "DNA" or "SS". "DNA" for DNA sequences and "SS" for secondary structure sequences. This parameter must be defined as "SS" when SS.features = TRUE.
check.cds	Logical. Incomplete CDs can lead to a false shift and an inaccurate hexamer frequencies. When check.cds = TRUE, hexamer frequencies will be calculated on the longest ORF. This parameter is strongly recommended to set as TRUE when mRNA is used as CDs.
ignore.illegal	Logical. If TRUE, the sequences with non-nucleotide characters (nucleotide characters: "a", "c", "g", "t") will be ignored when calculating hexamer frequencies.

Details

This function is used to make frequencies file. This file is needed when users are trying to build their own model.

In order to achieve high accuracy, mRNA should not be regarded as CDs and assigned to parameter cds.seq. However, CDs of some species may be insufficient for calculating frequencies, and mRNAs can be regarded as CDs with parameter check.cds = TRUE. In this case, hexamer frequencies will be calculated on ORF region.

Considering that it is time consuming to obtain secondary structure sequences, users can only provide nucleotide sequences and build a model without secondary structure features (SS.features = FALSE). If users want to build a model with secondary structure features, parameter SS.features should be set as TRUE. At the same time, the format of the sequences of mRNA.seq and lnc.seq should be secondary structure sequences (Dot-Bracket Notation). Secondary structure sequences can be obtained by function [run_RNAfold](#).

Please note that:

SS.features can improve the performance when the species of unevaluated sequences is identical to the species of the sequences that used to build the model.

However, if users are trying to predict sequences with the model trained on other species, SS.features may lead to low accuracy.

The frequencies file consists three groups: Hexamer Frequencies; acgu-ACGU Frequencies and acguD Frequencies.

Hexamer Frequencies are calculated on the original nucleotide sequences by employing k -mer scheme ($k = 6$), and the sliding window will slide 3nt each step.

For any secondary structure sequences (Dot-Bracket Notation), if one position is a dot, the corresponding nucleotide of the RNA sequence will be replaced with character "D". acguD Frequencies are the k -mer frequencies ($k = 4$) calculated on this new sequences.

Similarly, for any secondary structure sequences (Dot-Bracket Notation), if one position is "(" or ")", the corresponding nucleotide of the RNA sequence will be replaced with upper case ("A", "C", "G", "U").

A brief example,

```
DNA Sequence:      5'-   t   a   c   a   g   t   t   a   t   g   -3'
RNA Sequence:      5'-   u   a   c   a   g   u   u   a   u   g   -3'
Dot-Bracket Sequence:  5'-   .   .   .   .   (   (   (   (   (   (   -3'
acguD Sequence:    {      D, D, D, D, g, u, u, a, u, g    }
acgu-ACGU Sequence: {      u, a, c, a, G, U, U, A, U, G    }
```

Value

Returns a list which consists the frequencies of protein-coding sequences and non-coding sequences.

References

HAN Siyu, LIANG Yanchun and LI Ying*. LncFinder: Long Non-coding RNA Identification Tool Based on Features of Sequence Intrinsic Composition, Secondary Structure and EIIP Values. (2017) (*Submitted*)

Author(s)

Han Siyu

See Also

[run_RNAfold](#), [build_model](#), [extract_features](#).

Examples

```
### Only some brief examples:
data(demo_DNA.seq)
Seqs <- demo_DNA.seq

## Not run:
### Obtain the secondary structure sequences (Windows OS):
RNAfold.path <- '"E:/Program Files/ViennaRNA/RNAfold.exe"'
SS.seq <- run_RNAfold(Seqs, RNAfold.path = RNAfold.path, parallel.cores = 2)

### Make frequencies file with secondary strucutre features,
my_file_1 <- make_frequencies(cds.seq = SS.seq, mRNA.seq = SS.seq,
                             lncRNA.seq = SS.seq, SS.features = TRUE,
                             cds.format = "SS", lnc.format = "SS",
                             check.cds = TRUE, ignore.illegal = FALSE)

## End(Not run)

### Make frequencies file without secondary strucutre features,
my_file_2 <- make_frequencies(cds.seq = Seqs, lncRNA.seq = Seqs,
                             SS.features = FALSE, cds.format = "DNA",
                             lnc.format = "DNA", check.cds = TRUE,
                             ignore.illegal = FALSE)
```

```

### The input of cds.seq and lncRNA.seq can also be secondary structure
### sequences when SS.features = FALSE, such as,
data(demp_SS.seq)
SS.seq <- demo_SS.seq
my_file_3 <- make_frequencies(cds.seq = SS.seq, lncRNA.seq = Seqs,
                             SS.features = FALSE, cds.format = "SS",
                             lnc.format = "DNA", check.cds = TRUE,
                             ignore.illegal = FALSE)

```

max_orf

Find the longest ORF

Description

This function can find the longest ORF in one sequence.

Usage

```
max_orf(OneSeq, reverse.strand = FALSE)
```

Arguments

OneSeq Is one sequence. Can be a FASTA file read by package "seqinr" ([seqinr-package](#)) or just a string.

reverse.strand Logical. Whether find ORF on the reverse strand.

Details

This function can extract the longest ORF of one sequence. It returns the region, length and coverage of the longest ORF. Coverage is the ratio of ORF to transcript length. If `reverse.strand = TRUE`, ORF will be found on both the forward and reverse strand.

Value

Returns a list which consists ORF region (`ORF.Max.Seq`), length (`ORF.Max.Len`) and coverage (`ORF.Max.Cov`) of the longest ORF.

Author(s)

Han Siyu

Examples

```

### For one sequence:
OneSeq <- c("cccatgccagctagtaagcttagcc")
max_orf_1 <- max_orf(OneSeq, reverse.strand = TRUE)

### For a FASTA file contains several sequences:
## Not run:

```

```

### Use "read.fasta" function of package "seqinr" to read a FASTA file:
Seqs <- read.fasta(file =
"http://www.ncbi.nlm.nih.gov/WebSub/html/help/sample_files/nucleotide-sample.txt")

## End(Not run)

### Or just try to use our data "demo_DNA.seq"
data(demo_DNA.seq)
Seqs <- demo_DNA.seq

### Use apply function to find the longest ORF:
max_orf_2 <- sapply(Seqs, max_orf, reverse.strand = FALSE)

```

run_RNAfold

Obtain the Secondary Structure Sequences Using RNAfold

Description

This function can compute secondary structure sequences. The tool "RNAfold" of software "ViennaRNA" is required for this function.

Usage

```
run_RNAfold(Sequences, RNAfold.path = "RNAfold", parallel.cores = 2)
```

Arguments

Sequences	A FASTA file loaded by function <code>read.fasta</code> of package "seqinr" (seqinr-package).
RNAfold.path	String. Indicate the path of the program "RNAfold". By default is "RNAfold" for UNIX/Linux system. (See details.)
parallel.cores	Integer. The number of cores for parallel computation. By default the number of cores is 2, users can set as -1 to run this function with all cores.

Details

This function is used to compute secondary structure. The output of this function can be used in function `make_frequencies`, `extract_features`, `build_model` and `lnc_finder` when parameter `SS.features` is set as TRUE.

This function depends on the program "RNAfold" of software "ViennaRNA". (<http://www.tbi.univie.ac.at/RNA/index.html>)

Parameter `RNAfold.path` can be simply defined as "RNAfold" as default when the OS is UNIX/Linux. However, for some OS, such as Windows, users need to specify the `RNAfold.path` if the path of "RNAfold" haven't been added in environment variables.

This function can print the related information when the OS is UNIX/Linux, such as:

```
"25 of 100, length: 695 nt",
```

which means around 100 sequences are assigned to this node and the program is computing the 25th sequence. The length of this sequence is 695nt.

Value

Returns data.frame. The first row is RNA sequence; the second row is Dot-Bracket Notation of secondary structure sequences; the last row is minimum free energy (MFE).

Author(s)

Han Siyu

Examples

```
## Not run:
### For a FASTA file contains several sequences,
### Use "read.fasta" function of package "seqinr" to read a FASTA file:
Seqs <- read.fasta(file =
"http://www.ncbi.nlm.nih.gov/WebSub/html/help/sample_files/nucleotide-sample.txt")

### Or just try to use our data "demo_DNA.seq"
data(demo_DNA.seq)
Seqs <- demo_DNA.seq

### Windows:
RNAfold.path <- '"E:/Program Files/ViennaRNA/RNAfold.exe"'
SS.seq_1 <- run_RNAfold(Seqs, RNAfold.path = RNAfold.path, parallel.cores = 2)

### For UNIX/Linux, "RNAfold.path" can be just defined as "RNAfold" as default:
SS.seq_2 <- run_RNAfold(Seqs, RNAfold.path = "RNAfold", parallel.cores = 2)

## End(Not run)
```

svm_tune

Parameter Tuning of SVM

Description

This function conduct the parameter tuning of SVM. Parameters gamma and cost can be tuned using grid search.

Usage

```
svm_tune(dataset, folds.num = 10, seed = 1, gamma.range = (2^seq(-5, 0,
1)), cost.range = c(1, 4, 8, 16, 24, 32), return.model = TRUE,
parallel.cores = 2)
```

Arguments

dataset	The dataset obtained from function extract_features .
folds.num	Integer. Specify the number of folds for cross-validation. (Default: 10)
seed	Integer. Used to set the seed for cross-validation. (Default: 1)

<code>gamma.range</code>	The range of gamma. (Default: 2^{-5} to 1)
<code>cost.range</code>	The range of cost. (Default: $c(1, 4, 8, 16, 24, 32)$)
<code>return.model</code>	Logical. If TRUE, the function will return the best model trained on the full dataset. If FALSE, this function will return the optimal parameters.
<code>parallel.cores</code>	Integer. The number of cores for parallel computation. By default the number of cores is 2, users can set as -1 to run this function with all cores. If the number of <code>parallel.cores</code> is more than the <code>folds.num</code> (number of the folds for cross-validation), the number of <code>parallel.cores</code> will be set as <code>folds.num</code> automatically.

Details

During the model tuning, the performance of each combination of parameters will output. Sensitivity, Specificity, Accuracy, F-Measure and Kappa Value are used to evaluate the performances. The best gamma and cost (or best model) are selected based on Accuracy.

For the details of parameter gamma and cost, please refer to function `svm` of package "e1071".

For the details of metrics, please refer to function `confusionMatrix` of package "caret".

Value

Returns the optimal parameters when `return.model = FALSE`.

Or returns the best model when `return.model = TRUE`.

Author(s)

Han Siyu

See Also

`extract_features`

Examples

```
## Not run:
data(demo_DNA.seq)
Seqs <- demo_DNA.seq

positive_data <- extract_features(Seqs[1:5], Label = "NonCoding",
                                SS.features = FALSE, format = "DNA",
                                frequencies.file = "human",
                                parallel.cores = 2)

negative_data <- extract_features(Seqs[6:10], Label = "Coding",
                                SS.features = FALSE, format = "DNA",
                                frequencies.file = "human",
                                parallel.cores = 2)

my_dataset <- rbind(positive_data, negative_data)
```

```
### Or use our data "demo_dataset"
data(demo_dataset)
my_dataset <- demo_dataset

optimal_parameter <- svm_tune(my_dataset, folds.num = 2, seed = 1,
                             gamma.range = (2 ^ seq(-5, 0, 2)),
                             cost.range = c(1, 8, 16),
                             return.model = FALSE, parallel.core = 2)

### Users can set return.model = TRUE to return the best model.

## End(Not run)
```

Index

*Topic **datasets**

demo_dataset, [4](#)

demo_DNA.seq, [5](#)

demo_SS.seq, [5](#)

build_model, [2](#), [6–9](#), [12](#), [14](#)

confusionMatrix, [16](#)

demo_dataset, [4](#)

demo_DNA.seq, [5](#)

demo_SS.seq, [5](#)

extract_features, [3](#), [6](#), [9](#), [12](#), [14](#), [15](#)

lnc_finder, [3](#), [8](#), [14](#)

make_frequencies, [2](#), [3](#), [6–9](#), [10](#), [14](#)

max_orf, [13](#)

read.fasta, [5](#), [14](#)

run_RNAfold, [2](#), [6–12](#), [14](#)

svm, [3](#), [16](#)

svm_tune, [3](#), [7](#), [8](#), [15](#)