Package 'LncFinder'

July 3, 2020

Type Package

Title LncRNA Identification and Analysis Using Heterologous Features

Version 1.1.4

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

Acknowledgments CHENG Ming, FAN Linrui, GUO Yuan, LI Yaolong, SUN Ying, WANG Ruoyu

Description Long non-coding RNAs identification and analysis. Default models are trained with 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. This package can also extract other classic features and build new classifiers. Reference: Han SY., Liang YC., Li Y., et al. (2018) <doi:10.1093/bib/bby065>.

URL http://bmbl.sdstate.edu/lncfinder/

License GPL-3

Depends R (>= 2.10)

Imports seqinr (>= 2.1-3),
e1071 (>= 1.0),
parallel (>= 2.1.0),
caret (>= 6.0-71),

LazyData true

RoxygenNote 7.1.0

Encoding UTF-8

R topics documented:

`build_model` *Build Users' Own Model*

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 `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 `read.fasta` 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 pre-build 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.
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. During the process of svm tuning, if the number of parallel.cores is more than the folds.num (number of the folds for cross-validation), the number of parallel.cores will be set as folds.num automatically.
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)
gamma.range	The range of gamma. (Default: 2 ^ seq(-5,0,1))
cost.range	The range of cost. (Default: c(1,4,8,16,24,32))

Details

This function is used to build a new model with users' own sequences. Users can use function lnc_finder to predict the sequences with new models.

For the details of frequencies.file, please refer to function make_frequencies.

For the details of the features, please refer to function extract_features.

For the details of svm tuning, please refer to function svm_tune.

Value

Returns a svm model.

References

Siyu Han, Yanchun Liang, Qin Ma, Yangyi Xu, Yu Zhang, Wei Du, Cankun Wang & Ying Li. LncFinder: an integrated platform for long non-coding RNA identification utilizing sequence intrinsic composition, structural information, and physicochemical property. *Briefings in Bioinformatics*, 2019, 20(6):2009-2027.

Author(s)

HAN Siyu

See Also

make_frequencies, lnc_finder, extract_features, svm_tune, svm.

Examples

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

### Build the model with pre-build frequencies.file:
my_model <- build_model(lncRNA.seq = Seqs[1:5], mRNA.seq = Seqs[6:10],
                        frequencies.file = "human", SS.features = FALSE,
                        lncRNA.format = "DNA", mRNA.format = "DNA",
                        parallel.cores = 2, folds.num = 2, seed = 1,
                        gamma.range = (2 ^ seq(-5, -1, 2)),
                        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)
```

`compute_EIIP` *Extract the EIIP-derived features*

Description

This function can extract EIIP-derived features proposed by Han et al (2018).

Usage

```
compute_EIIP(
  Sequences,
  label = NULL,
  spectrum.percent = 0.1,
  quantile.probs = seq(0, 1, 0.25)
)
```

Arguments

`Sequences`	A FASTA file loaded by function `read.fasta` of `seqinr-package`.
`label`	Optional. String. Indicate the label of the sequences such as "NonCoding", "Coding".
`spectrum.percent`	Numeric specifying the percentage of the sorted power spectrum that be used to calculate the quantile-based features. For example, if `spectrum.percent = 0.1`, the top 10% percent of the sorted power spectrum will be used to compute the quantiles.
`quantile.probs`	Numeric. The probabilities with values in [0,1].

Details

The function `compute_EIIP` can extract EIIP (electron-ion interaction pseudo-potential) features including: signal at 1/3 position (`Signal.Peak`), average power (`Average.Power`), signal to noise ratio (`SNR`), and quantile-based features of one specified percentage of the sorted power spectrum (e.g. `0%`, `20%`, `40%`, `60%`, `70%`, `100%` when `quantile.probs = seq(0,1,0.2)` and `spectrum.percent = 0.1`).

In method LncFinder, EIIP features includes `Signal.Peak`, `SNR`, 0% (`Signal.Min`), 25% (`Singal.Q1`, 50% `Signal.Q2`), and 75% (`Signal.Max`) of the top 10% sorted power spectrum, i.e. `quantile.prob = seq(0,1,0.25)` and `spectrum.percent = 0.1`.

Value

A dataframe including the EIIP-derived features.

References

Siyu Han, Yanchun Liang, Qin Ma, Yangyi Xu, Yu Zhang, Wei Du, Cankun Wang & Ying Li. LncFinder: an integrated platform for long non-coding RNA identification utilizing sequence intrinsic composition, structural information, and physicochemical property. *Briefings in Bioinformatics*, 2019, 20(6):2009-2027.

Lalović, Dragutin, and Veljko Veljković. The global average DNA base composition of coding regions may be determined by the electron-ion interaction potential. *Biosystems*, 1990, 23(4):311-316.

Achuthsankar S Nair & Sivarama Pillai Sreenadhan. A coding measure scheme employing electron-ion interaction pseudopotential (EIIP). *Bioinformation*, 2006, 1(6):197-202.

Author(s)

HAN Siyu

See Also

`extract_features`

Examples

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

EIIP_res <- compute_EIIP(Seqs, label = "NonCoding", spectrum.percent = 0.25,
                         quantile.probs = seq(0, 1, 0.25))
```

---

compute_EucDistance *Compute Euclidean Distance*

Description

This function can compute Euclidean Distance proposed by method LncFinder (Han et al. 2018). Euclidean Distance can be calculated on full sequence or the longest ORF region. The step and *k* of the sliding window can also be customized.

Usage

```
compute_EucDistance(
  Sequences,
  label = NULL,
  referFreq,
  k = 6,
  step = 1,
  alphabet = c("a", "c", "g", "t"),
  on.ORF = FALSE,
  auto.full = FALSE,
  parallel.cores = 2
)
```

Arguments

Sequences	A FASTA file loaded by function read.fasta of seqinr-package.
label	Optional. String. Indicate the label of the sequences such as "NonCoding", "Coding".
referFreq	a list obtained from function make_referFreq.
k	An integer that indicates the sliding window size. (Default: 6)
step	Integer defaulting to 1 for the window step.
alphabet	A vector of single characters that specify the different character of the sequence. (Default: alphabet = c("a","c","g","t"))
on.ORF	Logical. If TRUE, Euclidean Distance will be calculated on the longest ORF region. NOTE: If TRUE, the input has to be DNA sequences. (Default: FALSE)
auto.full	Logical. When on.ORF = TRUE but no ORF can be found, if auto.full = TRUE, Euclidean Distance will be calculated on full sequences automatically; if auto.full is FALSE, the sequences that have no ORF will be discarded. Ignored when on.ORF = FALSE. (Default: FALSE)
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 can compute Euclidean Distance proposed by LncFinder (HAN et al. 2018). In LncFinder, two schemes are provided to calculate Euclidean Distance: 1) step = 3 and k = 6 on the longest ORF region; 2) step = 1 and k = 6 on full sequence. Using this function compute_EucDistance, both step, k, and calculated region (full sequence or ORF) can be customized to maximize its availability.

Value

A dataframe.

References

Siyu Han, Yanchun Liang, Qin Ma, Yangyi Xu, Yu Zhang, Wei Du, Cankun Wang & Ying Li. LncFinder: an integrated platform for long non-coding RNA identification utilizing sequence intrinsic composition, structural information, and physicochemical property. *Briefings in Bioinformatics*, 2019, 20(6):2009-2027.

Author(s)

HAN Siyu

See Also

make_referFreq, compute_LogDistance, compute_hexamerScore.

Examples

```
## Not run:
Seqs <- seqinr::read.fasta(file =
"http://www.ncbi.nlm.nih.gov/WebSub/html/help/sample_files/nucleotide-sample.txt")

referFreq <- make_referFreq(cds.seq = Seqs, lncRNA.seq = Seqs, k = 6, step = 3,
                            alphabet = c("a", "c", "g", "t"), on.orf = TRUE,
                            ignore.illegal = TRUE)

data(demo_DNA.seq)
Sequences <- demo_DNA.seq

EucDistance <- compute_EucDistance(Sequences, label = "NonCoding", referFreq = referFreq,
                                   k = 6, step = 3, alphabet = c("a", "c", "g", "t"),
                                   on.ORF = TRUE, auto.full = TRUE, parallel.cores = 2)

## End(Not run)
```

---

compute_FickettScore *Compute Fickett TESTCODE Score*

Description

This function can compute Fickett TESTCODE score of DNA sequences proposed by James W.Fickett (Fickett JW. 1982). Fickett TESTCODE score can be calculated on full sequence or the longest ORF region.

Usage

```
compute_FickettScore(
  Sequences,
  label = NULL,
  on.ORF = FALSE,
  auto.full = FALSE,
  parallel.cores = 2
)
```

Arguments

Sequences	A FASTA file loaded by function read.fasta of seqinr-package.
label	Optional. String. Indicate the label of the sequences such as "NonCoding", "Coding".

on.ORF	Logical. If TRUE, Fickett TESTCODE score will be calculated on the longest ORF region.
auto.full	Logical. When on.ORF = TRUE but no ORF can be found, if auto.full = TRUE, Fickett TESTCODE score will be calculated on full sequences automatically; if auto.full is FALSE, the sequences that have no ORF will be discarded. Ignored when on.ORF = FALSE. (Default: FALSE)
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 can compute Fickett TESTCODE score proposed by James W.Fickett (Fickett JW. 1982). Fickett TESTCODE score is selected as feature by method CPAT (Wang et al. 2013) and CPC2 (Kang et al. 2017). In CPAT, Fickett TESTCODE score is calculated on the longest ORF region, but CPC2 calculates the score on full sequence. This function compute_FickettScore improves the CPAT's code and is capable of computing the score on the longest ORF region as well as full sequence.

Value

A dataframe.

References

James W.Fickett. Recognition of protein coding regions in DNA sequences. *Nucleic Acids Research*, 1982, 10(17):5303-5318.

Siyu Han, Yanchun Liang, Qin Ma, Yangyi Xu, Yu Zhang, Wei Du, Cankun Wang & Ying Li. LncFinder: an integrated platform for long non-coding RNA identification utilizing sequence intrinsic composition, structural information, and physicochemical property. *Briefings in Bioinformatics*, 2019, 20(6):2009-2027.

Liguo Wang, Hyun Jung Park, Surendra Dasari, Shengqin Wang, JeanPierre Kocher & Wei Li. CPAT: coding-potential assessment tool using an alignment-free logistic regression model. *Nucleic Acids Research*, 2013, 41(6):e74-e74.

Yu-Jian Kang, De-Chang Yang, Lei Kong, Mei Hou, Yu-Qi Meng, Liping Wei & Ge Gao. CPC2: a fast and accurate coding potential calculator based on sequence intrinsic features. *Nucleic Acids Research*, 2017, 45(W1):W12-W16.

Author(s)

HAN Siyu

Examples

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

FickettScore <- compute_FickettScore(Seqs, label = NULL, on.ORF = TRUE,
                                     auto.full = TRUE, parallel.cores = 2)

## End(Not run)
```

compute_GC *Calculate GC content*

Description

This function can GC content of the input sequences.

Usage

```
compute_GC(
  Sequences,
  label = NULL,
  on.ORF = FALSE,
  auto.full = FALSE,
  parallel.cores = 2
)
```

Arguments

Sequences	A FASTA file loaded by function read.fasta of seqinr-package.
label	Optional. String. Indicate the label of the sequences such as "NonCoding", "Coding".
on.ORF	Logical. If TRUE, GC content will be calculated on the longest ORF region.
auto.full	Logical. When on.ORF = TRUE but no ORF can be found, if auto.full = TRUE, GC content will be calculated on full sequences automatically; if auto.full is FALSE, the sequences that have no ORF will be discarded. Ignored when on.ORF = FALSE. (Default: FALSE)
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 can basically compute GC content of DNA sequences: GC content = (nc + ng) / (na + nc + ng + nt). The function will ignored the ambiguous bases.

Value

A dataframe.

Author(s)

HAN Siyu

See Also

GC (package "seqinr-package")

Examples

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

gcContent <- compute_GC(Seqs, label = "NonCoding",on.ORF = TRUE,
                        auto.full = TRUE, parallel.cores = 2)

## End(Not run)
```

`compute_hexamerScore` *Compute Hexamer Score*

Description

This function can compute hexamer score proposed by method CPAT (Wang et al. 2013). Hexamer score can be calculated on full sequence or the longest ORF region. The step and *k* of the sliding window can also be customized.

Usage

```
compute_hexamerScore(
  Sequences,
  label = NULL,
  referFreq,
  k = 6,
  step = 1,
  alphabet = c("a", "c", "g", "t"),
  on.ORF = FALSE,
  auto.full = FALSE,
  parallel.cores = 2
)
```

Arguments

`Sequences`	A FASTA file loaded by function `read.fasta` of `seqinr-package`.
`label`	Optional. String. Indicate the label of the sequences such as "NonCoding", "Coding".
`referFreq`	A list obtained from function `make_referFreq`.
`k`	An integer that indicates the sliding window size. (Default: `6`)
`step`	Integer defaulting to `1` for the window step.
`alphabet`	A vector of single characters that specify the different character of the sequence. (Default: `alphabet = c("a","c","g","t")`)
`on.ORF`	Logical. If `TRUE`, hexamer score will be calculated on the longest ORF region. NOTE: If `TRUE`, the input has to be DNA sequences. (Default: `FALSE`)
`auto.full`	Logical. When `on.ORF = TRUE` but no ORF can be found, if `auto.full = TRUE`, hexamer score will be calculated on full sequences automatically; if `auto.full` is `FALSE`, the sequences that have no ORF will be discarded. Ignored when `on.ORF = FALSE`. (Default: `FALSE`)
`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 can compute hexamer score proposed by CPAT (Wang et al. 2013). In CPAT, hexamer score is calculated on the longest ORF region, and the step of the sliding window is 3 (i.e. `step = 3`). Hexamer means six adjoining bases, thus `k = 6`. But in function `compute_hexamerScore`, both `step`, `k`, and calculated region (full sequence or ORF) can be customized to maximize its availability.

Value

A dataframe.

References

Liguo Wang, Hyun Jung Park, Surendra Dasari, Shengqin Wang, JeanPierre Kocher, & Wei Li. CPAT: coding-potential assessment tool using an alignment-free logistic regression model. *Nucleic Acids Research*, 2013, 41(6):e74-e74.

Siyu Han, Yanchun Liang, Qin Ma, Yangyi Xu, Yu Zhang, Wei Du, Cankun Wang & Ying Li. LncFinder: an integrated platform for long non-coding RNA identification utilizing sequence intrinsic composition, structural information, and physicochemical property. *Briefings in Bioinformatics*, 2019, 20(6):2009-2027.

Author(s)

HAN Siyu

See Also

`make_referFreq`, `compute_LogDistance`, `compute_EucDistance`.

Examples

```
## Not run:
Seqs <- seqinr::read.fasta(file =
"http://www.ncbi.nlm.nih.gov/WebSub/html/help/sample_files/nucleotide-sample.txt")

referFreq <- make_referFreq(cds.seq = Seqs, lncRNA.seq = Seqs, k = 6, step = 1,
                            alphabet = c("a", "c", "g", "t"), on.orf = TRUE,
                            ignore.illegal = TRUE)

data(demo_DNA.seq)
Sequences <- demo_DNA.seq

hexamerScore <- compute_hexamerScore(Sequences, label = "NonCoding", referFreq = referFreq,
                                     k = 6, step = 1, alphabet = c("a", "c", "g", "t"),
                                    on.ORF = TRUE, auto.full = TRUE, parallel.cores = 2)

## End(Not run)
```

compute_kmer *Compute* k*-mer Features*

Description

This function can calculate the *k*-mer frequencies of the sequences.

Usage

```
compute_kmer(
  Sequences,
  label = NULL,
  k = 1:5,
  step = 1,
  freq = TRUE,
  improved.mode = FALSE,
  alphabet = c("a", "c", "g", "t"),
  on.ORF = FALSE,
  auto.full = FALSE,
  parallel.cores = 2
)
```

Arguments

Sequences	A FASTA file loaded by function read.fasta of seqinr-package.
label	Optional. String. Indicate the label of the sequences such as "NonCoding", "Coding".
k	An integer that indicates the sliding window size. (Default: 1:5)
step	Integer defaulting to 1 for the window step.
freq	Logical. If TRUE, the frequencies of different patterns are returned instead of counts. (Default: TRUE)
improved.mode	Logical. If TRUE, the frequencies will be normalized using the method proposed by PLEK (Li et al. 2014). Ignored if freq = FALSE. (Default: FALSE)
alphabet	A vector of single characters that specify the different character of the sequence. (Default: alphabet = c("a","c","g","t"))
on.ORF	Logical. If TRUE, the *k*-mer frequencies will be calculated on the longest ORF region. NOTE: If TRUE, the sequences have to be DNA. (Default: FALSE)
auto.full	Logical. When on.ORF = TRUE but no ORF can be found, if auto.full = TRUE, the *k*-mer frequencies will be calculated on the full sequence automatically; if auto.full is FALSE, the sequences that have no ORF will be discarded. Ignored when on.ORF = FALSE. (Default: FALSE)
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 can extract *k*-mer features. k and step can be customized. The count (freq = FALSE) or frequencies (freq = TRUE) of different patterns can be returned. If freq = TRUE, improved.mode is available. The improved mode is proposed by method PLEK. (Ref: Li et al. 2014)

Value

A dataframe.

Author(s)

HAN Siyu

Examples

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

kmer_res1 <- compute_kmer(Seqs, k = 1:5, step = 1, freq = TRUE, improved.mode = FALSE)

kmer_res2 <- compute_kmer(Seqs, k = 1:5, step = 3, freq = TRUE,
                          improved.mode = TRUE, on.ORF = TRUE, auto.full = TRUE)

## End(Not run)
```

---

compute_LogDistance *Compute Logarithm Distance*

Description

This function can compute Logarithm Distance proposed by method LncFinder (Han et al. 2018). Logarithm Distance can be calculated on full sequence or the longest ORF region. The step and *k* of the sliding window can also be customized.

Usage

```
compute_LogDistance(
  Sequences,
  label = NULL,
  referFreq,
  k = 6,
  step = 1,
  alphabet = c("a", "c", "g", "t"),
  on.ORF = FALSE,
  auto.full = FALSE,
  parallel.cores = 2
)
```

Arguments

Sequences	A FASTA file loaded by function read.fasta of seqinr-package.
label	Optional. String. Indicate the label of the sequences such as "NonCoding", "Coding".
referFreq	a list obtained from function make_referFreq.
k	An integer that indicates the sliding window size. (Default: 6)

step	Integer defaulting to 1 for the window step.
alphabet	A vector of single characters that specify the different character of the sequence. (Default: alphabet = c("a","c","g","t"))
on.ORF	Logical. If TRUE, Logarithm Distance will be calculated on the longest ORF region. NOTE: If TRUE, the input has to be DNA sequences. (Default: FALSE)
auto.full	Logical. When on.ORF = TRUE but no ORF can be found, if auto.full = TRUE, Logarithm Distance will be calculated on full sequences automatically; if auto.full is FALSE, the sequences that have no ORF will be discarded. Ignored when on.ORF = FALSE. (Default: FALSE)
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 can compute Logarithm Distance proposed by LncFinder (HAN et al. 2018). In LncFinder, two schemes are provided to calculate Logarithm Distance: 1) step = 3 and k = 6 on the longest ORF region; 2) step = 1 and k = 6 on full sequence. Method LncFinder uses scheme 1 to extract Logarithm Distance features. Using this function compute_EucDistance, both step, k, and calculated region (full sequence or ORF) can be customized to maximize its availability.

Value

A dataframe.

References

Siyu Han, Yanchun Liang, Qin Ma, Yangyi Xu, Yu Zhang, Wei Du, Cankun Wang & Ying Li. LncFinder: an integrated platform for long non-coding RNA identification utilizing sequence intrinsic composition, structural information, and physicochemical property. *Briefings in Bioinformatics*, 2019, 20(6):2009-2027.

Author(s)

HAN Siyu

See Also

make_referFreq, compute_EucDistance, compute_hexamerScore.

Examples

```
## Not run:
Seqs <- seqinr::read.fasta(file =
"http://www.ncbi.nlm.nih.gov/WebSub/html/help/sample_files/nucleotide-sample.txt")

referFreq <- make_referFreq(cds.seq = Seqs, lncRNA.seq = Seqs, k = 6, step = 3,
                            alphabet = c("a", "c", "g", "t"), on.orf = TRUE,
                            ignore.illegal = TRUE)

data(demo_DNA.seq)
Sequences <- demo_DNA.seq

LogDistance <- compute_LogDistance(Sequences, label = "NonCoding", referFreq = referFreq,
                                   k = 6, step = 3, alphabet = c("a", "c", "g", "t"),
```

```
                                    on.ORF = TRUE, auto.full = TRUE, parallel.cores = 2)

## End(Not run)
```

`compute_pI` *Compute Theoretical Isoelectric Point*

Description

This function is basically a wrapper for function `computePI`. This function translate DNA sequence into protein, and compute the theoretical isoelectric point (pI) of this protein.

Usage

```
compute_pI(
  Sequences,
  label = NULL,
  on.ORF = FALSE,
  auto.full = FALSE,
  ambiguous.base = FALSE,
  parallel.cores = 2
)
```

Arguments

`Sequences`	A FASTA file loaded by function `read.fasta` of `seqinr-package`.
`label`	Optional. String. Indicate the label of the sequences such as "NonCoding", "Coding".
`on.ORF`	Logical. If `TRUE`, pI will be calculated on the longest ORF region. NOTE: If `TRUE`, the input has to be DNA sequences. (Default: `FALSE`)
`auto.full`	Logical. When `on.ORF = TRUE` but no ORF can be found, if `auto.full = TRUE`, pI will be calculated on full sequences automatically; if `auto.full` is `FALSE`, the sequences that have no ORF will be discarded. Ignored when `on.ORF = FALSE`. (Default: `FALSE`)
`ambiguous.base`	If `TRUE`, ambiguous bases are taken into account when translating DNA sequences into proteins.
`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 can compute the pI of DNA sequences. Method CPC2 (Kang et al. 2017) uses this feature to identify lncRNAs, and this feature is evaluated in the article LncFinder (Han et al. 2018).

Using this function, the theoretical pI can be computed on full sequence or the longest ORF region. In CPC2, pI is calculated on ORF region.

Value

A dataframe.

Author(s)

HAN Siyu

Examples

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

pI_res <- compute_pI(Sequences, on.ORF = TRUE, auto.full = FALSE, ambiguous.base = FALSE)

## 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

```
data(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

```
data(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

```
data(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 structure of each sequence is obtained from program "RNAfold".

extract_features *Extract the Features*

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 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 = TURE.
frequencies.file	String or a list obtained from function make_frequencies. Input species name "human", "mouse" or "wheat" to use pre-build 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 as TRUE may lead to low accuracy.

Value

Returns a data.frame. 11 features when SS.features is FALSE, and 19 features when SS.features is 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 at 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 (Singal.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

Siyu Han, Yanchun Liang, Qin Ma, Yangyi Xu, Yu Zhang, Wei Du, Cankun Wang & Ying Li. LncFinder: an integrated platform for long non-coding RNA identification utilizing sequence intrinsic composition, structural information, and physicochemical property. *Briefings in Bioinformatics*, 2019, 20(6):2009-2027.

Author(s)

HAN Siyu

See Also

svm_tune, build_model, make_frequencies, run_RNAfold, read_SS.

Examples

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

### Extract features with pre-build 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".

## End(Not run)
```

`find_orfs` *Find ORFs*

Description

This function can find all the ORFs in one sequence.

Usage

```
find_orfs(OneSeq, reverse.strand = FALSE, max.only = TRUE)
```

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. Default: `FALSE`
`max.only`	Logical. If `TRUE`, only the longest ORF will be returned. Default: `TRUE`

Details

This function can extract ORFs of one sequence. It returns ORF region, length and coverage of the longest ORF when `max.only` = `TRUE` or ORF region, start position, end position, length and coverage of all the ORFs when `max.only` = `FALSE`. Coverage is the the ratio of the ORF to transcript length. If `reverse.strand` = `TRUE`, ORF will also be found on reverse strand.

Value

If `max.only` = `TRUE`, the function returns a list which consists the ORF region (`ORF.Max.Seq`), length (`ORF.Max.Len`) and coverage (`ORF.Max.Cov`) of the longest ORF. If `max.only` = `FALSE`, the function returns a dataframe which consists all the ORF sequences.

Author(s)

HAN Siyu

Examples

```
### For one sequence:
OneSeq <- c("cccatgcccagctagtaagcttagcc")
orf.info_1 <- find_orfs(OneSeq, reverse.strand = TRUE, max.only = FALSE)

### For a FASTA file contains several sequences:
## Not run:
### Use "read.fasta" function of package "seqinr" to read a FASTA file:
Seqs <- seqinr::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:
orf.info_2 <- sapply(Seqs, find_orfs, reverse.strand = FALSE, max.only = FALSE)
```

`lnc_finder` *Long Non-coding RNA Identification*

Description

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

Usage

```
lnc_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 `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.
`format`	String. Define the format of the `Sequences`. Can be `"DNA"` or `"SS"`. `"DNA"` for DNA sequences and `"SS"` for secondary structure sequences.
`frequencies.file`	String or a list obtained from function `make_frequencies`. Input species name `"human"`, `"mouse"` or `"wheat"` to use pre-build 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 pre-build 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

Siyu Han, Yanchun Liang, Qin Ma, Yangyi Xu, Yu Zhang, Wei Du, Cankun Wang & Ying Li. LncFinder: an integrated platform for long non-coding RNA identification utilizing sequence intrinsic composition, structural information, and physicochemical property. *Briefings in Bioinformatics*, 2019, 20(6):2009-2027.

Author(s)

HAN Siyu

See Also

build_model, make_frequencies, extract_features, run_RNAfold, read_SS.

Examples

```
## Not run:
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)

### 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` *Make the frequencies file for new classifier construction*

Description

This function is used to calculate the frequencies of lncRNAs, CDs, and secondary structure sequences. The frequencies file can be used to build the classifier using function `extract_features`. Functions `make_frequencies` and `extract_features` are useful when users are trying to build their own model.

NOTE: Function `make_frequencies` makes the frequencies file for building the classifiers of LncFinder method. If users need to calculate Logarithm-Distance, Euclidean-Distance, and hexamer score, the frequencies file need to be computed using function `make_referFreq`.

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 `seqinr-package` or secondary structure sequences (Dot-Bracket Notation) obtained form 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 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 = TURE.
check.cds	Logical. Incomplete CDs can lead to a false shift and a 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 for LncFinder method. 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 3 nt 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

Siyu Han, Yanchun Liang, Qin Ma, Yangyi Xu, Yu Zhang, Wei Du, Cankun Wang & Ying Li. LncFinder: an integrated platform for long non-coding RNA identification utilizing sequence intrinsic composition, structural information, and physicochemical property. *Briefings in Bioinformatics*, 2019, 20(6):2009-2027.

Author(s)

HAN Siyu

See Also

`run_RNAfold`, `read_SS`, `build_model`, `extract_features`, `make_referFreq`.

Examples

```
### Only for 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)
```

`make_referFreq` *Make Frequencies File for Log.Dist, Euc.Dist, and hexamer score*

Description

This function is used to calculate the frequencies of lncRNAs and CDs. The Frequencies file can be used to calculate Logarithm-Distance (`compute_LogDistance`), Euclidean-Distance (`compute_EucDistance`), and hexamer score (`compute_hexamerScore`).

NOTE: If users need to make frequencies file to build new LncFinder classifier using function `extract_features`, please refer to function `make_frequencies`.

Usage

```
make_referFreq(
  cds.seq,
  lncRNA.seq,
  k = 6,
  step = 1,
  alphabet = c("a", "c", "g", "t"),
  on.orf = TRUE,
  ignore.illegal = TRUE
)
```

Arguments

`cds.seq`	Coding sequences (mRNA without UTRs). Can be a FASTA file loaded by `seqinr-package`.
`lncRNA.seq`	Long non-coding RNA sequences. Can be a FASTA file loaded by `seqinr-package`.
`k`	An integer that indicates the sliding window size. (Default: `6`)
`step`	Integer defaulting to `1` for the window step.
`alphabet`	A vector of single characters that specify the different character of the sequence. (Default: `alphabet = c("a","c","g","t")`)
`on.orf`	Logical. Incomplete CDs can lead to a false shift and a inaccurate hexamer frequencies. When `on.orf = TRUE`, the frequencies will be calculated on the longest ORF. This parameter is strongly recommended to set as `TRUE` when mRNA is used as CDs. Only available when `alphabet = c("a","c","g","t")`. (Default: `TRUE`)

`ignore.illegal` Logical. If `TRUE`, the sequences with non-nucleotide characters (nucleotide characters: "a", "c", "g", "t") will be ignored when calculating the frequencies. Only available when `alphabet = c("a","c","g","t")`. (Default: `TRUE`)

Details

This function is used to make frequencies file for the computation of Logarithm-Distance (`compute_LogDistance`), Euclidean-Distance (`compute_EucDistance`), and hexamer score (`compute_hexamerScore`).

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. In that case, mRNAs can be regarded as CDs with parameter `on.orf = TRUE`, and the frequencies will be calculated on ORF region. If `on.orf = TRUE`, users can set `step = 3` to simulate the translation process.

Value

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

References

Siyu Han, Yanchun Liang, Qin Ma, Yangyi Xu, Yu Zhang, Wei Du, Cankun Wang & Ying Li. LncFinder: an integrated platform for long non-coding RNA identification utilizing sequence intrinsic composition, structural information, and physicochemical property. *Briefings in Bioinformatics*, 2019, 20(6):2009-2027.

Author(s)

HAN Siyu

See Also

`make_frequencies`, `compute_LogDistance`, `compute_EucDistance`, `compute_hexamerScore`.

Examples

```
## Not run:
Seqs <- seqinr::read.fasta(file =
"http://www.ncbi.nlm.nih.gov/WebSub/html/help/sample_files/nucleotide-sample.txt")

referFreq <- make_referFreq(cds.seq = Seqs, lncRNA.seq = Seqs, k = 6, step = 1,
                            alphabet = c("a", "c", "g", "t"), on.orf = TRUE,
                            ignore.illegal = TRUE)

## End(Not run)
```

read_SS *Read Secondary Structure Information*

Description

This function can read secondary structure information from your own file instead of obtaining from function run_RNAfold. This function will be useful if users have had secondary structure sequences (Dot-Bracket Notation).

Usage

```
read_SS(
  oneFile.loc,
  seqRNA.loc,
  seqSS.loc,
  separateFile = TRUE,
  withMFE = TRUE
)
```

Arguments

oneFile.loc	String. The location of your sequence file. This file should contains one (and only one) RNA sequence and its secondary structure sequence in Dot-Bracket Notation. This parameter needs to be defined only when separateFile = FALSE. See Details for more information.
seqRNA.loc	String. The location of your RNA sequences file (FASTA format). If your RNA sequences and secondary structure sequences are in two files, you need to define the locations of two files respectively. And the files with multiple sequences are supported for this option. This parameter needs to be defined only when separateFile is TRUE. Location of secondary structure sequences file is also needed (parameter seqSS.loc). See Details for more information.
seqSS.loc	String. The location of your secondary structure sequences file (FASTA format).
separateFile	Logical. Your RNA sequence(s) and secondary structure sequence(s) are in separate files? If separateFile = FALSE, your file should have one (and only one) RNA sequence and its secondary structure sequence. No limit when separateFile = TRUE.
withMFE	Logical. Whether MFE is provided at the end of secondary structure sequence. If withMFE = TRUE, MFE will be extracted. The format should be in accordance with the output format of RNAfold.

Details

When users want to predict sequences with secondary structure features, users may have had their own secondary structure sequences. With this function, users can read SS information from their files. Two kind of files are supported: RNA sequence and SS sequence in one file separateFile is FALSE or in separate files separateFile = TRUE.

separateFile = FALSE is used for secondary structure that obtained from some popular programs, such as RNAfold. In this case, the output file only contains one RNA sequence and its SS. Besides, this file only have two rows: RNA sequence and its SS sequences. Thus, this option is more

favorable when the file only have one sequence and the sequence are in accordance with the output format of RNAfold.

If users obtained the SS sequence from experiments, RNA sequence and SS sequence may be in two files. In this case, users can select `separateFile = TRUE`. Two files should be in FASTA format and one file can have multiple sequences. The sequences in two files should have the same order. If your data are obtained from experiments or other sources, it is highly recommended that users should build new model with this data, since the SS sequences of pre-built model are obtained for RNAfold and may have many differences with experimental data.

Value

A dataframe. The first row is RNA sequence, the second row is Dot-Bracket Notation of secondary structure sequence, the third row is MFE (if MFE is provided).

Author(s)

HAN Siyu

See Also

run_RNAfold

Examples

```
## Not run:
### Load sequence data
data("demo_DNA.seq")
Seqs <- demo_DNA.seq[1:4]
### Convert sequences from vector to string.
Seqs <- sapply(Seqs, seqinr::getSequence, as.string = TRUE)
### Write a fasta file.
seqinr::write.fasta(Seqs, names = names(Seqs), file.out = "tmp.RNA.fa", as.string = TRUE)

### For Windows system: (Your path of RNAfold.)
RNAfold.path <- '"E:/Program Files/ViennaRNA/RNAfold.exe"'
### Define the parameters of RNAfold. See documents of RNAfold for more information.
RNAfold.command <- paste(RNAfold.path, " --noPS -i tmp.RNA.fa -o output", sep = "")
### Run RNAfold and output four result files.
system(RNAfold.command)

### Read secondary structure information for one file.
result_1 <- read_SS(oneFile.loc = "output_ENST00000510062.1.fold",
                    separateFile = FALSE, withMFE = TRUE)
### Read secondary sturcture sequences for multiple files.
filePath <- dir(pattern = ".fold")
result_2 <- sapply(filePath, read_SS, separateFile = FALSE, withMFE = TRUE)
result_2 <- as.data.frame(result_2)

## End(Not run)
```

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 read.fasta of 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.

If users have their own SS data, users can use function read_SS to load them, instead of obtaining from RNAfold.

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

See Also

read_SS

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[1:2], 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_cv k-*fold Cross Validation for SVM*

Description

This function conduct *k*-fold Cross Validation for SVM.

Usage

```
svm_cv(
  dataset,
  label.col = 1,
  positive.class = NULL,
  folds.num = 10,
  seed = 1,
  parallel.cores = 2,
  ...
)
```

Arguments

dataset	The dataset obtained from function extract_features. Or datasets used to build the classifier.
label.col	integer specifying the column number of the label. (Default: 1)
positive.class	Character. Indicate the positive class of the dataset. (Default: NonCoding) The value of this parameter should be identical to one of the classes of the response vectors.
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)

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. If the number of parallel.cores is more than the folds.num (number of the folds for cross-validation), the number of parallel.cores will be set as folds.num automatically.

... additional parameters for function svm.

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, svm_tune.

Examples

```
## Not run:
data(demo_dataset)
my_dataset <- demo_dataset

cv_res <- svm_cv(my_dataset, folds.num = 4, seed = 1,
                 parallel.core = 2, cost = 3, kernel = "radial", gamma = 0.5)

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

## 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,
  label.col = 1,
  positive.class = "NonCoding",
  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. Or datasets used to build the classifier.
label.col	integer specifying the column number of the label. (Default: 1)
positive.class	Character. Indicate the positive class of the dataset. (Default: NonCoding) The value of this parameter should be identical to one of the classes of the response vectors.
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)
gamma.range	The range of gamma. (Default: 2 ^ seq(-5,0,1))
cost.range	The range of cost. (Default: c(1,4,8,16,24,32))
return.model	Logical. If TRUE, the function will return the best model trained on the full dataset. If FALSE, this function will return the optimal parameters.
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. If the number of parallel.cores is more than the folds.num (number of the folds for cross-validation), the number of parallel.cores will be set as folds.num 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, svm_cv.

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, positive.class = "NonCoding",
                              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