Package 'LION'

July 23, 2022

Type Package

Title Predicting Long non-coding RNA-Protein Interaction

Version 0.2.8

Maintainer HAN Siyu <siyu.han@tum.de>

Acknowledgments
TAI Yun Hsiu, WANG Ruoyu, CHEN Hang, CHENG Ming, GUO Yuan, LI Han, FAN Linrui, YUE Yanzhu

Description Prediction of lncRNA-
protein interaction using sequence, motif, secondary structure and physicochemical properies.

License GPL (>= 3)

Depends R (>= 3.5.0)

Imports seqinr (>= 2.1-3),
randomForest (>= 4.6-14),
parallel (>= 2.1.0),
caret (>= 6.0-71),
RCurl (>= 1.95-4.12)

URL https://github.com/HAN-Siyu/LION

RelevantData https://github.com/HAN-Siyu/LION_Supplementary

BugReports https://github.com/HAN-Siyu/LION/issues

LazyData true

RoxygenNote 7.1.1

Encoding UTF-8

R topics documented:

computeFreq 2
computeMLC 5
computeMotifs 6
computePhysChem 9
computePhysChem_AAindex 11
computeStructure 13
demoIDX 16
demoNegativeSeq 16
demoPositiveSeq 16

evaluatePrediction . . . 17
featureFreq . . . 18
featureMotifs . . . 20
featurePhysChem . . . 22
featureStructure . . . 24
formatSeq . . . 26
mod_LION . . . 27
mod_LncADeep . . . 27
mod_lncPro . . . 27
mod_rpiCOOL . . . 28
mod_RPISeq . . . 28
normalizeData . . . 28
randomForest_CV . . . 29
randomForest_RFE . . . 31
randomForest_tune . . . 32
runPredator . . . 34
runRNAsubopt . . . 36
run_confidentPrediction . . . 37
run_LION . . . 39
run_LncADeep . . . 41
run_lncPro . . . 44
run_rpiCOOL . . . 47
run_RPISeq . . . 50

Index **53**

`computeFreq` *Computation of the K-Mer Frequencies of RNA or Protein Sequences*

Description

This function can calculate the *k*-mer frequencies of RNA or protein sequences. Three kinds of protein representations are available.

Usage

```
computeFreq(
  seqs,
  seqType = c("RNA", "Pro"),
  computePro = c("RPISeq", "DeNovo", "rpiCOOL"),
  k = 3,
  EDP = FALSE,
  normalize = c("none", "row", "column"),
  normData = NULL,
  parallel.cores = 2,
  cl = NULL
)
```

Arguments

seqs	sequences loaded by function read.fasta from seqinr-package. Or a list of RNA/protein sequences. RNA sequences will be converted into lower case letters, but protein sequences will be converted into upper case letters. Each sequence should be a vector of single characters.
seqType	a string that specifies the nature of the sequence: "RNA" or "Pro" (protein). If the input is DNA sequence and seqType = "RNA", the DNA sequence will be converted to RNA sequence automatically. Default: "RNA".
computePro	a string that specifies the computation mode of protein sequences: "RPISeq", "DeNovo", or "rpiCOOL". Ignored when seqType = "RNA". Three modes indicate three different amino acid residues classifications that corresponds to methods "RPISeq", "De Novo prediction", and "rpiCOOL". See details below. Default: "RPISeq".
k	an integer that indicates the sliding window step. Default: 3.
EDP	logical. If TRUE, entropy density profile (EDP) will be computed. Default: FALSE.
normalize	can be "none", "row" or "column". Indicate if the frequencies should be normalized. If normalize, should the features be normalized by row (each sequence) or by column (each feature)? See details below. Default: "none".
normData	is the normalization data generated by this function. If the input dataset is training set, or normalize strategy is "none" or "row", just leave normData = NULL. If users want to build test set and the normalize strategy is "column", the normalization data of the corresponding training set generated by this function should be passed to this argument. See examples.
parallel.cores	an integer specifying the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
cl	parallel cores to be passed to this function.

Details

Function computeFreq calculate the *k*-mer frequencies of RNA/protein sequences. Three computation modes of protein frequencies are:

RPISeq: {A, G, V}, {I, L, F, P}, {Y, M, T, S}, {H, N, Q, W}, {R, K}, {D, E}, {C} (Ref: [3]);

DeNovo: {D, E}, {H, R, K}, {C, G, N, Q, S, T, Y}, {A, F, I, L, M, P, V, W} (Ref: [4]).

rpiCOOL: {A, E}, {I, L, F, M, V}, {N, D, T, S}, {G}, {P}, {R, K, Q, H}, {Y, W}, {C} (Ref: [5]).

If EDP = TRUE, entropy density profile (EDP) will be computed with equation: s_*i* = -1/H * c_*i* * log2(c_*i*), H = -sum(c_*j* * log2(c_*j*)). c is the frequencies, *i* and *j* represents the indices of *k*-mer frequencies. (Ref: [6])

The function also provides two normalization strategies: by row (each sequence) or by column (each feature). If by row, the dataset will be processed with equation (Ref: [2]): d_*i* = (f_*i* - min{f_1, f_2, ...}) / max{f_1, f_2, ...}. f_1, f_2, ..., f_*i* are the original values of each row.

If by column, the dataset will be processed with: d_*i* = (f_*i* - min{f_1, f_2, ...}) / (max{f_1, f_2, ...} - min{_f1, f_2, ...}).

In [2], normalization is computed by row (each sequence).

Value

If `normalize = "none"` or `normalize = "row"`, this function will return a data frame. Row names are sequences names, and column names are polymer names.

If `normalize = "column"`, the function will return a list containing features (a data frame named "feature") and normalization values (a list named "normData") for extracting features for test sets.

References

[1] Han S, *et al*. LION: an integrated R package for effective ncRNA-protein interaction prediction. (*Submitted*)

[2] Shen J, Zhang J, Luo X, *et al*. Predicting protein-protein interactions based only on sequences information. Proc. Natl. Acad. Sci. U. S. A. 2007; 104:4337-41

[3] Muppirala UK, Honavar VG, Dobbs D. Predicting RNA-protein interactions using only sequence information. BMC Bioinformatics 2011; 12:489

[4] Wang Y, Chen X, Liu Z-P, *et al*. De novo prediction of RNA-protein interactions from sequence information. Mol. BioSyst. 2013; 9:133-142

[5] Akbaripour-Elahabad M, Zahiri J, Rafeh R, *et al*. rpiCOOL: A tool for In Silico RNA-protein interaction detection using random forest. J. Theor. Biol. 2016; 402:1-8

[6] Yang C, Yang L, Zhou M, *et al*. LncADeep: an ab initio lncRNA identification and functional annotation tool based on deep learning. Bioinformatics. 2018; 34(22):3825-3834.

See Also

`featureFreq`

Examples

```
# Use "read.fasta" function of package "seqinr" to read a FASTA file:

seqs1 <- seqinr::read.fasta(file =
"http://www.ncbi.nlm.nih.gov/WebSub/html/help/sample_files/nucleotide-sample.txt")
seqFreq1 <- computeFreq(seqs1, seqType = "RNA", k = 4, normalize = "row",
                        parallel.cores = 2)

data(demoPositiveSeq)
seqs2 <- demoPositiveSeq$Pro.positive

# Training set with normalization on column:

seqFreq2 <- computeFreq(seqs2, seqType = "Pro", computePro = "RPISeq", k = 3,
                        normalize = "column", parallel.cores = 2)

# If build a test set with normalization on column,
# "normData" of the corresponding training set (generated by this function) is required:

seqFreq3 <- computeFreq(seqs2, seqType = "Pro", computePro = "RPISeq", k = 3,
                        normalize = "column", normData = seqFreq2$normData,
                        parallel.cores = 2)

# If no normalization used:

seqFreq4 <- computeFreq(seqs2, seqType = "Pro", computePro = "DeNovo", k = 3,
                        normalize = "none", parallel.cores = 2)
```

computeMLC *Computation of the most-like CDS region of an RNA sequence*

Description

This function compute the most-like CDS (MLC) region of one RNA. Methods based on the longest open reading frame (ORF) and maximum subarray sum (MSS) are supported.

Usage

```
computeMLC(oneRNA, mode = c("ORF", "MSS"))
```

Arguments

oneRNA	one RNA loaded by function read.fasta from seqinr-package. Or a list of one RNA sequence. The sequence should be a vector of single characters.
mode	can be "ORF" (compute the longest open reading frame) and/or "MSS" (compute subsequence having the maximum subarray sum of hexamer score).

Value

A data frame. The MLC region, length and coverage of the MLC will be returned. MSS-based method is based on [1] and [2]. ORF extraction function is borrowed from our previous work [3].

References

[1] Yang C, Yang L, Zhou M, *et al*. LncADeep: an ab initio lncRNA identification and functional annotation tool based on deep learning. Bioinformatics. 2018; 34(22):3825-3834.

[2] Sun L, Luo H, Bu D, *et al*. Utilizing sequence intrinsic composition to classify protein-coding and long non-coding transcripts. Nucleic acids research. 2013; 41(17):e166-e166.

[3] Han S, Liang Y, Ma Q, *et al*. 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.

Examples

```
# Use "read.fasta" function of package "seqinr" to read a FASTA file:

seqRNA <- seqinr::read.fasta(file =
"http://www.ncbi.nlm.nih.gov/WebSub/html/help/sample_files/nucleotide-sample.txt")

# Compute the MLC region using both "ORF" and "MSS" methods:

MLC_list <- lapply(seqRNA, computeMLC, mode = c("ORF", "MSS"))
```

computeMotifs *Counting the Number of Motifs in RNA or Protein Sequences*

Description

Counts the number of motifs occurring in RNA/protein sequences. Motifs employed by tool "rpiCOOL" can be selected. New motifs can also be defined.

Usage

```
computeMotifs(
  seqs,
  seqType = c("RNA", "Pro"),
 motifRNA = c("rpiCOOL", "Fox1", "Nova", "Slm2", "Fusip1", "PTB", "ARE", "hnRNPA1",
   "PUM", "U1A", "HuD", "QKI", "U2B", "SF1", "HuR", "YB1", "AU", "UG", "selected5"),
 motifPro = c("rpiCOOL", "E", "H", "K", "R", "H_R", "EE", "KK", "HR_RH", "RS_SR",
    "RGG", "YGG"),
  newMotif = NULL,
  newMotifOnly = FALSE,
  parallel.cores = 2,
  cl = NULL
)
```

Arguments

seqs	sequences loaded by function read.fasta from seqinr-package. Or a list of RNA/protein sequences. RNA sequences will be converted into lower case letters, but protein sequences will be converted into upper case letters. Each sequence should be a vector of single characters.
seqType	a string that specifies the nature of the sequence: "RNA" or "Pro" (protein). If the input is DNA sequence and seqType = "RNA", the DNA sequence will be converted to RNA sequence automatically. Default: "RNA".
motifRNA	strings specifying the motifs that are counted in RNA sequences. Ignored if seqType = "Pro". Options: "rpiCOOL", "selected5", "Fox1", "Nova", "Slm2", "Fusip1", "PTB", "ARE", "hnRNPA1", "PUM", "U1A", "HuD", "QKI", "U2B", "SF1", "HuR", "YB1", "AU", and "UG". Multiple elements can be selected at the same time. If "rpiCOOL", all default motifs will be counted. "selected5" indicates the total number of the occurrences of: PUM, Fox-1, U1A, Nova, and ARE which are regarded as the five most over-presented binding motifs. See details below.
motifPro	strings specifying the motifs that are counted in protein sequences. Ignored if seqType = "RNA". Options: "rpiCOOL", "E", "H", "K", "R", "H_R", "EE", "KK", "HR_RH", "RS_SR", "RGG", and "YGG". Multiple elements can be selected at the same time. "H_R" indicates the total number of the occurrences of: H and R. "HR_RH" indicates the total number of the occurrences of: HR and RH. "RS_SR" indicates the total number of the occurrences of: RS and SR. If "rpiCOOL", the default motifs of rpiCOOL ("E", "K", "H_R", "EE", "KK", "RS_SR", "RGG", and "YGG") will be counted. See details below.

newMotif	list defining new motifs not listed above. New motifs are counted in RNA or protein sequences. For example, newMotif = list(hnRNPA1 = c("UAGGGU","UAGGGA"),SF1 = "UACUAAC"). This parameter can be used together with parameter motifRNA or motifPro. Default: NULL.
newMotifOnly	logical. If TRUE, only the new motifs defined in newMotif will be counted. Default: FALSE.
parallel.cores	an integer specifying the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
cl	parallel cores to be passed to this function.

Details

This function can count the motifs in RNA or protein sequences.

The default motifs are selected or derived from tool "rpiCOOL" (Ref: [2]).

- Motifs of RNA
  1. Fox1: UGCAUGU;
  2. Nova: UCAUUUCAC, UCAUUUCAU, CCAUUUCAC, CCAUUUCAU;
  3. Slm2: UAAAC, UAAAA, UAAUC, UAAUA;
  4. Fusip1: AAAGA, AAAGG, AGAGA, AGAGG, CAAGA, CAAGG, CGAGA, CGAGG;
  5. PTB: UUUUU, UUUCU, UCUUU, UCUCU;
  6. ARE: UAUUUAUU;
  7. hnRNPA1: UAGGGU, UAGGGA;
  8. PUM: UGUAAAUA, UGUAGAUA, UGUAUAUA, UGUACAUA;
  9. U1A: AUUGCAC;
  10. HuD: UUAUUU;
  11. QKI: AUUAAU, AUUAAC, ACUAAU, ACUAAC;
  12. U2B: AUUGCAG;
  13. SF1: UACUAAC;
  14. HuR: UUUAUUU, UUUGUUU, UUUCUUU, UUUUUUU;
  15. YB1: CCUGCG, UCUGCG;
  16. AU: AU;
  17. UG: UG.

  If "rpiCOOL", all default motifs will be counted, and there is no need to input other default motifs. "selected5" indicates the total number of the occurrences of: PUM, Fox-1, U1A, Nova, and ARE which are regarded as the five most over-represented binding motifs.
- Motifs of protein
  1. E: E;
  2. H: H;
  3. K: K;
  4. R: R;
  5. EE: EE;
  6. KK: KK;
  7. HR ("H_R"): H, R;
  8. HR ("HR_RH"): HR, RH;

9. RS ("RS_SR"): RS, SR;
10. RGG: RGG;
11. YGG: YGG.
    If "rpiCOOL", default motifs of rpiCOOL ("E", "K", "H_R", "EE", "KK", "RS_SR", "RGG", and "YGG") will be counted.

There are some minor differences between this function and the extraction scheme of rpiCOOL. In this function, motifs will be scanned directly. As to the extraction scheme of rpiCOOL, some motifs ("UG", "AU", and "H_R") are scanned in a 10 nt/aa sliding-window.

New motif patterns are also supported. Users can pass new patterns to argument "newMotif" as a list. Format:

newMotif = list(*motif_name* = c("*motif_pattern_1*","*motif_pattern_2*")).

For example: newMotif = list(HR_RH = c("HR","RH"),RGG = "RGG"). "HR_RH" is the name of this motif which contains two patterns: "HR" and "RH".

Value

This function returns a data frame. Row names are sequences names, and column names are motif names.

References

[1] Han S, *et al*. LION: an integrated R package for effective ncRNA-protein interaction prediction. (*Submitted*)

[2] Akbaripour-Elahabad M, Zahiri J, Rafeh R, *et al*. rpiCOOL: A tool for In Silico RNA-protein interaction detection using random forest. J. Theor. Biol. 2016; 402:1-8

[3] Pancaldi V, Bahler J. In silico characterization and prediction of global protein-mRNA interactions in yeast. Nucleic Acids Res. 2011; 39:5826-36

[4] Castello A, Fischer B, Eichelbaum K, *et al*. Insights into RNA Biology from an Atlas of Mammalian mRNA-Binding Proteins. Cell 2012; 149:1393-1406

[5] Ray D, Kazan H, Cook KB, *et al*. A compendium of RNA-binding motifs for decoding gene regulation. Nature 2013; 499:172-177

[6] Jiang P, Singh M, Coller HA. Computational assessment of the cooperativity between RNA binding proteins and MicroRNAs in Transcript Decay. PLoS Comput. Biol. 2013; 9:e1003075

See Also

featureMotifs

Examples

```
data(demoPositiveSeq)
seqsRNA <- demoPositiveSeq$RNA.positive
seqsPro <- demoPositiveSeq$Pro.positive

motifRNA1 <- computeMotifs(seqsRNA, seqType = "RNA", motifRNA = "rpiCOOL",
                           parallel.cores = 2)

motifRNA2 <- computeMotifs(seqsRNA, seqType = "RNA",
                           motifRNA = c("Fox1", "HuR", "ARE"), parallel.cores = 2)

motifPro1 <- computeMotifs(seqsPro, seqType = "Pro",
```

```
                          motifPro = c("rpiCOOL", "HR_RH"), parallel.cores = 2)

# Customized motifs are also supported and can be extracted with default motifs.
# Pass new motif patterns to "newMotif" argument as a list:

motifPro2 <- computeMotifs(seqsPro, seqType = "Pro", motifPro = c("E", "K", "KK"),
                          newMotif = list(HR_RH = c("HR", "RH"), RGG = "RGG"),
                          parallel.cores = 2)

motifPro3 <- computeMotifs(seqsPro, seqType = "Pro", motifPro = c("rpiCOOL"),
                          newMotif = list(HR_RH = c("HR", "RH"), RGG = "RGG"),
                          parallel.cores = 2)

# set "newMotifOnly = TRUE", if compute customized motifs only:

motifPro4 <- computeMotifs(seqsPro, seqType = "Pro",
                          newMotif = list(HR_RH = c("HR", "RH"), RGG = "RGG"),
                          newMotifOnly = TRUE, parallel.cores = 2)
```

---

computePhysChem *Computation of the Physicochemical Features of RNA or Protein Sequences*

Description

The function computePhysChem computes the physicochemical features of RNA or protein sequences.

Usage

```
computePhysChem(
  seqs,
  seqType = c("RNA", "Pro"),
  Fourier.len = 10,
  physchemRNA = c("hydrogenBonding", "vanderWaal"),
 physchemPro = c("polarity.Grantham", "polarity.Zimmerman", "bulkiness.Zimmerman",
    "isoelectricPoint.Zimmerman", "hphob.BullBreese", "hphob.KyteDoolittle",
    "hphob.Eisenberg", "hphob.HoppWoods"),
  as.list = TRUE,
  parallel.cores = 2,
  cl = NULL
)
```

Arguments

seqs	sequences loaded by function read.fasta from seqinr-package. Or a list of RNA/protein sequences. RNA sequences will be converted into lower case letters, but protein sequences will be converted into upper case letters. Each sequence should be a vector of single characters.
seqType	a string that specifies the nature of the sequence: "RNA" or "Pro" (protein). If the input is DNA sequence and seqType = "RNA", the DNA sequence will be converted to RNA sequence automatically. Default: "RNA".

Fourier.len	positive integer specifying the Fourier series length that will be used as features. The Fourier.len should be >= the length of the input sequence. Default: 10.
physchemRNA	strings specifying the physicochemical properties that are computed in RNA sequences. Ignored if seqType = "Pro". Options: "hydrogenBonding" for Hydrogen-bonding and "vanderWaal" for Van der Waal's interaction Multiple elements can be selected at the same time. (Ref: [2])
physchemPro	strings specifying the physicochemical properties that are computed in protein sequences. Ignored if seqType = "RNA". Options: "polarity.Grantham", "polarity.Zimmerman", "bulkiness.Zimmerman", "isoelectricPoint.Zimmerman", "hphob.BullBreese", "hphob.KyteDoolittle", "hphob.Eisenberg", and "hphob.HoppWoods". Multiple elements can be selected at the same time. See details below. (Ref: [3-9])
as.list	logical. The result will be returned as a list or a data frame.
parallel.cores	an integer that indicates the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
cl	parallel cores to be passed to this function.

Details

The default physicochemical properties are selected or derived from tool "catRAPID" (Ref: [10]) and "lncPro" (Ref: [11]). In "catRAPID", Fourier.len = 50; in "lncPro", Fourier.len is set as 10.

- The physicochemical properties of RNA
  1. Hydrogen-bonding ("hydrogenBonding") (Ref: [2])
  2. Van der Waal's interaction ("vanderWaal") (Ref: [2])
- The physicochemical properties of protein sequence
  1. polarity "polarity.Grantham" (Ref: [3])
  2. polarity "polarity.Zimmerman" (Ref: [4])
  3. bulkiness "bulkiness.Zimmerman" Ref: [4]
  4. isoelectric point "isoelectricPoint.Zimmerman" (Ref: [4])
  5. hydropathicity "hphob.BullBreese" (Ref: [5])
  6. hydropathicity "hphob.KyteDoolittle" (Ref: [6])
  7. hydropathicity "hphob.Eisenberg" (Ref: [7])
  8. hydropathicity "hphob.HoppWoods" (Ref: [8])

Value

This function returns a data frame if as.list = FALSE or returns a list if as.list = TRUE.

References

[1] Han S, *et al*. LION: an integrated R package for effective ncRNA-protein interaction prediction. (*Submitted*)

[2] Morozova N, Allers J, Myers J, *et al*. Protein-RNA interactions: exploring binding patterns with a three-dimensional superposition analysis of high resolution structures. Bioinformatics 2006; 22:2746-52

[3] Grantham R. Amino acid difference formula to help explain protein evolution. Science 1974; 185:862-4

[4] Zimmerman JM, Eliezer N, Simha R. The characterization of amino acid sequences in proteins by statistical methods. J. Theor. Biol. 1968; 21:170-201

[5] Bull HB, Breese K. Surface tension of amino acid solutions: a hydrophobicity scale of the amino acid residues. Arch. Biochem. Biophys. 1974; 161:665-670

[6] Kyte J, Doolittle RF. A simple method for displaying the hydropathic character of a protein. J. Mol. Biol. 1982; 157:105-132

[7] Eisenberg D, Schwarz E, Komaromy M, *et al*. Analysis of membrane and surface protein sequences with the hydrophobic moment plot. J. Mol. Biol. 1984; 179:125-42

[8] Hopp TP, Woods KR. Prediction of protein antigenic determinants from amino acid sequences. Proc. Natl. Acad. Sci. U. S. A. 1981; 78:3824-8

[9] Kawashima S, Kanehisa M. AAindex: amino acid index database. Nucleic Acids Res. 2000; 28:374

[10] Bellucci M, Agostini F, Masin M, *et al*. Predicting protein associations with long noncoding RNAs. Nat. Methods 2011; 8:444-445

[11] Lu Q, Ren S, Lu M, *et al*. Computational prediction of associations between long non-coding RNAs and proteins. BMC Genomics 2013; 14:651

See Also

featurePhysChem

Examples

```
data(demoPositiveSeq)
seqsRNA <- demoPositiveSeq$RNA.positive
seqsPro <- demoPositiveSeq$Pro.positive

# Return a data frame:
physChemRNA <- computePhysChem(seqs = seqsRNA, seqType = "RNA",
                               Fourier.len = 10, as.list = FALSE)

# Return a list:
physChemPro <- computePhysChem(seqs = seqsPro, seqType = "Pro", Fourier.len = 8,
                               physchemPro = c("polarity.Grantham",
                                               "polarity.Zimmerman",
                                               "hphob.BullBreese",
                                               "hphob.KyteDoolittle",
                                               "hphob.Eisenberg",
                                               "hphob.HoppWoods"),
                               as.list = TRUE)
```

---

computePhysChem_AAindex

Computation of the AAindex-Based Physicochemical Features of Protein Sequences

---

Description

The function computePhysChem_AAindex computes the physicochemical features of protein sequences with the help of AAindex provided by aaindex from seqinr-package.

Usage

```
computePhysChem_AAindex(
  seqPro,
  entry.index = NULL,
  Fourier.len = 10,
  parallel.cores = 2,
  cl = NULL
)
```

Arguments

seqPro	protein sequences loaded by function read.fasta from seqinr-package. Each sequence should be a vector of single characters.
entry.index	The entry number of AAindex. For example the entry number of "Kyte & Doolittle Hydrophaty" index is 151, and the entry number of "Hopp-Woods Hydrophilicity" is 115. Thus, the corresponding entry.index is c(151,115) for these two physicochemical indices. See examples and aaindex of "seqinr" package.
Fourier.len	positive integer specifying the Fourier series length that will be used as features. The Fourier.len should be >= the length of the input sequence. Default: 10.
parallel.cores	an integer that indicates the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
cl	parallel cores to be passed to this function.

Value

This function returns a data frame.

References

[1] Kawashima S, Kanehisa M. AAindex: amino acid index database. Nucleic Acids Res. 2000; 28:374

[2] Charif D, Lobry JR. SeqinR 1.0-2: a contributed package to the R project for statistical computing devoted to biological sequences retrieval and analysis. In: Structural approaches to sequence evolution: Molecules, networks, populations. 2007; 207-232

See Also

computePhysChem, aaindex

Examples

```
data(demoPositiveSeq)
seqs <- demoPositiveSeq$Pro.positive
data(aaindex, package = "seqinr")
```

```
# Check the aaindex list provided by package "seqinr":

?seqinr::aaindex

# Supose that we need "Kyte & Doolittle Hydrophaty" index,
# and we can find the entries with Kyte as author:

which(sapply(aaindex, function(x) length(grep("Kyte", x$A)) != 0))

# This returns entry number 151.
# And 151 is the "entry.index" used by this function.
# Users can also obtain the entry number by retrieving information in other
# fields such as "x$D" (Data description) or "x$T" (Title of the article).
# See documentation of "seqinr::aaindex" for detailed information.

feature_aaindex_1 <- computePhysChem_AAindex(seqPro = seqs, entry.index = 151,
                                             Fourier.len = 10, parallel.cores = 2)

# If you need to compute features with more than aaindex data:

feature_aaindex_2 <- computePhysChem_AAindex(seqPro = seqs, entry.index = c(151, 68),
                                             Fourier.len = 10, parallel.cores = 2)
```

`computeStructure` *Computation of the Secondary Structural Features of RNA or Protein Sequences*

Description

The function `computeStructure` computes the secondary structural features of RNA or protein sequences. ViennaRNA package and Predator is required.

Usage

```
computeStructure(
  seqs,
  seqType = c("RNA", "Pro"),
  structureRNA.num = 6,
  structurePro = c("ChouFasman", "DeleageRoux", "Levitt"),
  Fourier.len = 10,
  workDir.Pro = getwd(),
  as.list = TRUE,
  path.RNAsubopt = "RNAsubopt",
  path.Predator = "Predator/predator",
  path.stride = "Predator/stride.dat",
  verbose = FALSE,
  parallel.cores = 2,
  cl = NULL
)
```

Arguments

seqs	sequences loaded by function read.fasta from seqinr-package. Or a list of RNA/protein sequences. RNA sequences will be converted into lower case letters, but protein sequences will be converted into upper case letters, and non-AA letters will be ignored. Each sequence should be a vector of single characters.
seqType	a string that specifies the nature of the sequence: "RNA" or "Pro" (protein). If the input is DNA sequence and seqType = "RNA", the DNA sequence will be converted to RNA sequence automatically. Default: "RNA".
structureRNA.num	integer. The number of random samples of suboptimal structures. Default: 6.
structurePro	strings specifying the secondary structural information that are extracted from protein sequences. Ignored if seqType = "RNA". Options: "ChouFasman", "DeleageRoux", and "Levitt". See details below.(Ref: [2-4]) Multiple elements can be selected at the same time.
Fourier.len	positive integer specifying the Fourier series length that will be used as features. The Fourier.len should be >= the length of the input sequence. Default: 10.
workDir.Pro	string specifying the directory for temporary files. The temp files will be deleted automatically when the calculation is completed.
as.list	logical. The result will be returned as a list or data frame.
path.RNAsubopt	string specifying the location of RNAsubopt program. (Ref: [5])
path.Predator	string specifying the location of Predator program. (Ref: [6])
path.stride	string specifying the location of file "stride.dat" required by program Predator.
verbose	logical. Should the relevant information be printed during the calculation? (Only available on Linux.)
parallel.cores	an integer that indicates the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
cl	parallel cores to be passed to this function.

Details

The secondary structures of RNA and protein are computed by RNAsubopt and Predator, respectively. And the protein secondary features are encoded using three amino acid scales:

1. Chou & Fasman conformational parameter (Ref: [2])
2. Deleage & Roux conformational parameter (Ref: [3])
3. Levitt normalised frequency (Ref: [4])

The feature encoding strategy is based on lncPro (Ref: [7]).

This function depends on the program "RNAsubopt" of software "ViennaRNA" (http://www.tbi.univie.ac.at/RNA/index.html) and "Predator" (https://bioweb.pasteur.fr/packages/pack@predator@2.1.2).

Parameter path.RNAsubopt can be simply defined as "RNAsubopt" as default when the OS is UNIX/Linux. However, for some OS, such as Windows, users may need to specify the path.RNAsubopt if the path of "RNAsubopt" haven't been added in environment variables (e.g. path.RNAsubopt = '"C:/Program Files/ViennaRNA/RNAsubopt.exe"').

Program "Predator" is only available on UNIX/Linux and 32-bit Windows OS.

Value

This function returns a data frame if `as.list = FALSE` or returns a list if `as.list = TRUE`.

References

[1] Han S, *et al*. LION: an integrated R package for effective ncRNA-protein interaction prediction. (*Submitted*)

[2] Chou PY, Fasman GD. Prediction of the secondary structure of proteins from their amino acid sequence. Adv. Enzymol. Relat. Areas Mol. Biol. 1978; 47:45-148

[3] Deleage G, Roux B. An algorithm for protein secondary structure prediction based on class prediction. Protein Eng. Des. Sel. 1987; 1:289-294

[4] Levitt M. Conformational preferences of amino acids in globular proteins. Biochemistry 1978; 17:4277-85

[5] Frishman D, Argos P. Incorporation of non-local interactions in protein secondary structure prediction from the amino acid sequence. Protein Eng. 1996; 9:133-42

[6] Lorenz R, Bernhart SH, Honer zu Siederdissen C, *et al*. ViennaRNA Package 2.0. Algorithms Mol. Biol. 2011; 6:26

[7] Lu Q, Ren S, Lu M, *et al*. Computational prediction of associations between long non-coding RNAs and proteins. BMC Genomics 2013; 14:651

See Also

`runRNAsubopt`, `runPredator`, `featureStructure`

Examples

```
## Not run:
data(demoPositiveSeq)
seqsRNA <- demoPositiveSeq$RNA.positive
seqsPro <- demoPositiveSeq$Pro.positive

# You need to use your own paths:

path.Predator <- "/mnt/external_drive_1/hansy/predator/predator"
path.stride <- "/mnt/external_drive_1/hansy/predator/stride.dat"

structureRNA <- computeStructure(seqsRNA, seqType = "RNA", structureRNA.num = 6,
                                 Fourier.len = 10, as.list = FALSE,
                                 path.RNAsubopt = "RNAsubopt", parallel.cores = 2)

structurePro <- computeStructure(seqsPro, seqType = "Pro",
                                 structurePro = c("ChouFasman", "DeleageRoux",
                                                  "Levitt"),
                                 Fourier.len = 10, workDir.Pro = getwd(),
                                 as.list = TRUE, path.Predator = path.Predator,
                                 path.stride = path.stride, parallel.cores = 2)

## End(Not run)
```

demoIDX *A demo of sequence indices*

Description

This data is an example of the input of formatSeq.

Usage

```
data(demoIDX)
```

Format

A data frame containing the names of RNA and protein sequences.

demoNegativeSeq *A demo of negative pairs*

Description

This file contains 20 negative RNA-protein interaction pairs. Required by some examples.

Usage

```
data(demoNegativeSeq)
```

demoPositiveSeq *A demo of positive pairs*

Description

This file contains 20 positive RNA-protein interaction pairs. Required by some examples.

Usage

```
data(demoPositiveSeq)
```

evaluatePrediction *Evaluate Predicted Result*

Description

This function can evaluate prediction based on reference labels and predicted results.

Usage

```
evaluatePrediction(reference, prediction, positive.class = NULL)
```

Arguments

reference	a factor/character string of classes to be used as the true results.
prediction	a factor/character string of predicted classes.
positive.class	NULL or string. Which class is the positive class? Should be one of the classes in label column. The first class in label column will be selected as the positive class if leave positive.class = NULL.

Details

reference and prediction should have exactly the same classes. More information please refer to confusionMatrix.

Value

A dataframe that reports TP, TN, FP, FN, Sensitivity, Specificity, Accuracy, F-Measure (F1-Score), MCC (Matthews Correlation Coefficient) and Cohen's Kappa.

References

Kuhn M. Building predictive models in R using the caret package. Journal of statistical software. 2008; 28(5):1-26.

See Also

confusionMatrix

Examples

```
# Following codes only show how to use this function
# and cannot reflect the genuine performance of tools or classifiers.

data(demoPositiveSeq)
seqRNA <- demoPositiveSeq$RNA.positive
seqPro <- demoPositiveSeq$Pro.positive

# Predicting ncRNA-protein pairs using RPISeq (web-based):

Res_RPISeq <- run_RPISeq(seqRNA = seqRNA, seqPro = seqPro,
                         parallel.cores = 2) # Network is required.

# Evaluating the result:
```

```
perf_RPISeq <- evaluatePrediction(reference = rep("Non.Interact", 20),
                                  prediction = Res_RPISeq$RPISeq_Web_RF_pred,
                                  positive.class = "Non.Interact")
```

`featureFreq` *Extraction of the* K*-Mer Features of RNA and Protein Sequences*

Description

Basically a wrapper for `computeFreq` function. This function can calculate the *k*-mer frequencies of RNA and protein sequences at the same time and format the results as the dataset that can be used to build classifier.

Usage

```
featureFreq(
  seqRNA,
  seqPro,
  label = NULL,
  featureMode = c("concatenate", "combine"),
  computePro = c("RPISeq", "DeNovo", "rpiCOOL"),
  k.Pro = 3,
  k.RNA = 4,
  EDP = FALSE,
  normalize = c("none", "row", "column"),
  normData = NULL,
  parallel.cores = 2,
  cl = NULL
)
```

Arguments

`seqRNA`	RNA sequences loaded by function `read.fasta` from `seqinr-package`. Or a list of RNA sequences. RNA sequences will be converted into lower case letters. Each sequence should be a vector of single characters.
`seqPro`	protein sequences loaded by function `read.fasta` from `seqinr-package`. Or a list of protein sequences. Protein sequences will be converted into upper case letters. Each sequence should be a vector of single characters.
`label`	optional. A string or a vector of strings or `NULL`. Indicates the class of the samples such as "Interact", "Non.Interact". Default: `NULL`.
`featureMode`	a string that can be `"concatenate"` or `"combine"`. If `"concatenate"`, the *k*-mer features of RNA and proteins will be simply concatenated. If `"combine"`, the returned dataset will be formed by combining the *k*-mer features of RNA and proteins. See details below. Default: `"concatenate"`.
`computePro`	a string that specifies the computation mode of protein sequence: `"RPISeq"`, `"DeNovo"`, or `"rpiCOOL"`. Ignored when `seqType = "RNA"`. Three modes indicate three different amino acid residues classifications that corresponds to the methods "RPISeq", "De Novo prediction", and "rpiCOOL". See details below. Default: `"RPISeq"`.

k.Pro	an integer that indicates the sliding window step of RNA sequences. Default: 4.
k.RNA	an integer that indicates the sliding window step of protein sequences. Default: 3.
EDP	logical. If TRUE, entropy density profile (EDP) will be computed. Default: FALSE.
normalize	can be "none", "row" or "column". Indicate if the frequencies should be normalized. If normalize, should the features be normalized by row (each sequence) or by column (each feature)? See details below. Default: "none".
normData	is the normalization data generated by this function. If the input dataset is training set, or normalize strategy is "none" or "row", just leave normData = NULL. If users want to build test set and the normalize strategy is "column", the normalization data of the corresponding training set generated by this function should be passed to this argument. See examples.
parallel.cores	an integer specifying the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
cl	parallel cores to be passed to this function.

Details

see computeFreq.

Value

If normalize = "none" or normalize = "row", this function will return a data frame. Row names are sequences names, and column names are polymer names. The names of RNA and protein sequences are separated with ".", i.e. row names format: "*RNASequenceName.proteinSequenceName*" (e.g. "YDL227C.YOR198C"). If featureMode = "combine", the polymers of RNA and protein sequences are also separated with ".", i.e. column format: "*RNAPolymerName.proteinPolymerName*" (e.g. "aa.CCA").

If normalize = "column", the function will return a list containing features (a data frame named "feature") and normalization values (a list named "normData") for extracting features for test sets.

References

[1] Han S, *et al*. LION: an integrated R package for effective ncRNA-protein interaction prediction. (*Submitted*)

[2] Shen J, Zhang J, Luo X, *et al*. Predicting protein-protein interactions based only on sequences information. Proc. Natl. Acad. Sci. U. S. A. 2007; 104:4337-41

[3] Muppirala UK, Honavar VG, Dobbs D. Predicting RNA-protein interactions using only sequence information. BMC Bioinformatics 2011; 12:489

[4] Wang Y, Chen X, Liu Z-P, *et al*. De novo prediction of RNA-protein interactions from sequence information. Mol. BioSyst. 2013; 9:133-142

[5] Akbaripour-Elahabad M, Zahiri J, Rafeh R, *et al*. rpiCOOL: A tool for In Silico RNA-protein interaction detection using random forest. J. Theor. Biol. 2016; 402:1-8

See Also

computeFreq

Examples

```
data(demoPositiveSeq)
seqsRNA <- demoPositiveSeq$RNA.positive
seqsPro <- demoPositiveSeq$Pro.positive

dataset1 <- featureFreq(seqRNA = seqsRNA, seqPro = seqsPro, label = "Interact",
                        featureMode = "comb", computePro = "DeNovo", k.Pro = 3,
                        k.RNA = 2, normalize = "row", parallel.cores = 2)

# Training set with normalization on column:

dataset2 <- featureFreq(seqRNA = seqsRNA, seqPro = seqsPro, featureMode = "conc",
                        computePro = "rpiCOOL", k.Pro = 3, k.RNA = 4,
                        normalize = "column", parallel.cores = 2)

# If build a test set with normalization on column,
# "normData" of the corresponding training set (generated by this function) is required:

dataset3 <- featureFreq(seqRNA = seqsRNA, seqPro = seqsPro, featureMode = "conc",
                        computePro = "rpiCOOL", k.Pro = 3, k.RNA = 4,
                        normalize = "column", normData = dataset2$normData,
                        parallel.cores = 2)
```

---

featureMotifs *Extraction of the Motif Features of RNA and Protein Sequences*

Description

Basically a wrapper for computeMotifs function. This function can count the motifs of RNA and protein sequences at the same time and format the results as the dataset that can be used to build classifier.

Usage

```
featureMotifs(
  seqRNA,
  seqPro,
  label = NULL,
  featureMode = c("concatenate", "combine"),
  newMotif.RNA = NULL,
  newMotif.Pro = NULL,
  newMotifOnly.RNA = FALSE,
  newMotifOnly.Pro = FALSE,
  parallel.cores = 2,
  cl = NULL,
  ...
)
```

Arguments

seqRNA	RNA sequences loaded by function read.fasta from seqinr-package. Or a list of RNA sequences. RNA sequences will be converted into lower case letters.

seqPro	protein sequences loaded by function read.fasta from seqinr-package. Or a list of protein sequences. Protein sequences will be converted into upper case letters.
label	optional. A string or a vector of strings or NULL. Indicates the class of the samples such as "Interact", "Non.Interact". Default: NULL.
featureMode	a string that can be "concatenate" or "combine". If "concatenate", the motif features of RNA and proteins will be simply concatenated. If "combine", the returned dataset will be formed by combining the motif features of RNA and proteins. See details below. Default: "concatenate".
newMotif.RNA	a list specifying the motifs that are counted in RNA sequences. Default: NULL. For example, newMotif = list(hnRNPA1 = c("UAGGGU","UAGGGA"),SF1 = "UACUAAC"). Can be used with parameter motifRNA (see parameter ...) to count motifs in RNA sequences.
newMotif.Pro	a list specifying the motifs that are counted in protein sequences. Default: NULL. For example, newMotif = list(YGG = "YGG",E = "E"). Can be used with parameter motifPro (see parameter ...) to count motifs in protein sequences.
newMotifOnly.RNA	logical. If TRUE, only the new motifs defined in newMotif.RNA will be counted. Default: FALSE.
newMotifOnly.Pro	logical. If TRUE, only the new motifs defined in newMotif.Pro will be counted. Default: FALSE.
parallel.cores	an integer that indicates the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
cl	parallel cores to be passed to this function.
...	argument motifRNA and motifPro to be passed to computeMotifs. Used to compute the default motifs. See examples below.

Details

If featureMode = "concatenate", *m* RNA motif features will be simply concatenated with *n* protein motif features, and the final result has *m* + *n* features. If featureMode = "combine", *m* RNA motif features will be combined with *n* protein motif features, resulting in *m* * *n* possible combinations.

... can be used to pass the default motif patterns of RNA and protein sequences. See arguments motifRNA and motifPro in computeMotifs.

Value

This function returns a data frame. Row names are the sequences names, and column names are the motif names. The names of RNA and protein sequences are separated with ".", i.e. row names format: "*RNASequenceName.proteinSequenceName*" (e.g. "YDL227C.YOR198C"). If featureMode = "combine", the motif names of RNA and protein sequences are also separated with ".", i.e. column names format: "motif_*RNAMotifName*.motif_*proteinMotifName*" (e.g. "motif_PUM.motif_EE").

References

[1] Han S, *et al*. LION: an integrated R package for effective ncRNA-protein interaction prediction. (*Submitted*)

[2] Akbaripour-Elahabad M, Zahiri J, Rafeh R, *et al*. rpiCOOL: A tool for In Silico RNA-protein interaction detection using random forest. J. Theor. Biol. 2016; 402:1-8

[3] Pancaldi V, Bahler J. In silico characterization and prediction of global protein-mRNA interactions in yeast. Nucleic Acids Res. 2011; 39:5826-36

[4] Castello A, Fischer B, Eichelbaum K, *et al*. Insights into RNA Biology from an Atlas of Mammalian mRNA-Binding Proteins. Cell 2012; 149:1393-1406

[5] Ray D, Kazan H, Cook KB, *et al*. A compendium of RNA-binding motifs for decoding gene regulation. Nature 2013; 499:172-177

[6] Jiang P, Singh M, Coller HA. Computational assessment of the cooperativity between RNA binding proteins and MicroRNAs in Transcript Decay. PLoS Comput. Biol. 2013; 9:e1003075

See Also

computeMotifs

Examples

```
data(demoPositiveSeq)
seqsRNA <- demoPositiveSeq$RNA.positive
seqsPro <- demoPositiveSeq$Pro.positive

dataset1 <- featureMotifs(seqRNA = seqsRNA, seqPro = seqsPro, featureMode = "conc",
                          newMotif.RNA = list(motif1 = c("cc", "cu")),
                          newMotif.Pro = list(motif2 = "KK"),
                          motifRNA = c("Fusip1", "AU", "UG"),
                          motifPro = c("E", "K", "HR_RH"))

dataset2 <- featureMotifs(seqRNA = seqsRNA, seqPro = seqsPro, featureMode = "comb",
                          newMotif.RNA = list(motif1 = c("cc", "cu")),
                          newMotif.Pro = list(motif2 = c("R", "H")),
                          newMotifOnly.RNA = TRUE, newMotifOnly.Pro = FALSE)
```

---

featurePhysChem *Extraction of the Physicochemical Features of RNA and Protein Sequences*

Description

Basically a wrapper for computePhysChem function. This function can extract physicochemical features of RNA and protein sequences at the same time and format the results as the dataset that can be used to build classifier.

Usage

```
featurePhysChem(
  seqRNA,
  seqPro,
  label = NULL,
  parallel.cores = 2,
  cl = NULL,
  ...
)
```

Arguments

seqRNA	RNA sequences loaded by function read.fasta from seqinr-package. Or a list of RNA sequences. RNA sequences will be converted into lower case letters. Each sequence should be a vector of single characters.
seqPro	protein sequences loaded by function read.fasta from seqinr-package. Or a list of protein sequences. Protein sequences will be converted into upper case letters. Each sequence should be a vector of single characters.
label	optional. A string or a vector of strings or NULL. Indicates the class of the samples such as "Interact", "Non.Interact". Default: NULL.
parallel.cores	an integer that indicates the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
cl	parallel cores to be passed to this function.
...	arguments (Fourier.len, physchemRNA and physchemPro) to be passed to computePhysChem. See computePhysChem and examples below.

Details

see computePhysChem.

Value

This function returns a data frame.

References

[1] Han S, *et al*. LION: an integrated R package for effective ncRNA-protein interaction prediction. (*Submitted*)

[2] Morozova N, Allers J, Myers J, *et al*. Protein-RNA interactions: exploring binding patterns with a three-dimensional superposition analysis of high resolution structures. Bioinformatics 2006; 22:2746-52

[3] Grantham R. Amino acid difference formula to help explain protein evolution. Science 1974; 185:862-4

[4] Zimmerman JM, Eliezer N, Simha R. The characterization of amino acid sequences in proteins by statistical methods. J. Theor. Biol. 1968; 21:170-201

[5] Bull HB, Breese K. Surface tension of amino acid solutions: a hydrophobicity scale of the amino acid residues. Arch. Biochem. Biophys. 1974; 161:665-670

[6] Kyte J, Doolittle RF. A simple method for displaying the hydropathic character of a protein. J. Mol. Biol. 1982; 157:105-132

[7] Eisenberg D, Schwarz E, Komaromy M, *et al*. Analysis of membrane and surface protein sequences with the hydrophobic moment plot. J. Mol. Biol. 1984; 179:125-42

[8] Hopp TP, Woods KR. Prediction of protein antigenic determinants from amino acid sequences. Proc. Natl. Acad. Sci. U. S. A. 1981; 78:3824-8

[9] Kawashima S, Kanehisa M. AAindex: amino acid index database. Nucleic Acids Res. 2000; 28:374

[10] Bellucci M, Agostini F, Masin M, *et al*. Predicting protein associations with long noncoding RNAs. Nat. Methods 2011; 8:444-445

[11] Lu Q, Ren S, Lu M, *et al*. Computational prediction of associations between long non-coding RNAs and proteins. BMC Genomics 2013; 14:651

See Also

computePhysChem

Examples

```
data(demoPositiveSeq)
seqsRNA <- demoPositiveSeq$RNA.positive
seqsPro <- demoPositiveSeq$Pro.positive

# Pass "Fourier.len", "physchemRNA" and "physchemPro" using "..." argument:

dataset1 <- featurePhysChem(seqRNA = seqsRNA, seqPro = seqsPro,
                            label = "Interact", Fourier.len = 10,
                            physchemRNA = c("hydrogenBonding", "vanderWaal"),
                            physchemPro = c("polarity.Grantham", "polarity.Zimmerman",
                                            "hphob.BullBreese", "hphob.KyteDoolittle",
                                            "hphob.Eisenberg", "hphob.HoppWoods"))

# Using the default setting:

dataset2 <- featurePhysChem(seqRNA = seqsRNA, seqPro = seqsPro)
```

---

featureStructure *Extraction of the Secondary Structural Features of RNA and Protein Sequences*

Description

Basically a wrapper for computeStructure function. This function can extract secondary structure features of RNA and protein sequences at the same time and format the results as the dataset that can be used to build classifier. ViennaRNA package and Predator is required.

Usage

```
featureStructure(
  seqRNA,
  seqPro,
  label = NULL,
  parallel.cores = 2,
  cl = NULL,
  ...
)
```

Arguments

seqRNA	RNA sequences loaded by function read.fasta from seqinr-package. Or a list of RNA sequences. RNA sequences will be converted into lower case letters. Each sequence should be a vector of single characters.
seqPro	protein sequences loaded by function read.fasta from seqinr-package. Or a list of protein sequences. Protein sequences will be converted into upper case letters. Each sequence should be a vector of single characters.

label	optional. A string or a vector of strings or NULL. Indicates the class of the samples such as "Interact", "Non.Interact". Default: NULL.
parallel.cores	an integer that indicates the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
cl	parallel cores to be passed to this function.
...	arguments (structureRNA.num, structurePro, Fourier.len, workDir.Pro, path.RNAsubopt, path.Predator and path.stride passed to function computeStructure. See example below.

Details

see computeStructure.

Value

This function returns a data frame.

References

[1] Han S, *et al*. LION: an integrated R package for effective ncRNA-protein interaction prediction. (*Submitted*)

[2] Chou PY, Fasman GD. Prediction of the secondary structure of proteins from their amino acid sequence. Adv. Enzymol. Relat. Areas Mol. Biol. 1978; 47:45-148

[3] Deleage G, Roux B. An algorithm for protein secondary structure prediction based on class prediction. Protein Eng. Des. Sel. 1987; 1:289-294

[4] Levitt M. Conformational preferences of amino acids in globular proteins. Biochemistry 1978; 17:4277-85

[5] Frishman D, Argos P. Incorporation of non-local interactions in protein secondary structure prediction from the amino acid sequence. Protein Eng. 1996; 9:133-42

[6] Lorenz R, Bernhart SH, Honer zu Siederdissen C, *et al*. ViennaRNA Package 2.0. Algorithms Mol. Biol. 2011; 6:26

[7] Lu Q, Ren S, Lu M, *et al*. Computational prediction of associations between long non-coding RNAs and proteins. BMC Genomics 2013; 14:651

See Also

runRNAsubopt, runPredator, computeStructure

Examples

```
## Not run:
data(demoNegativeSeq)
seqsRNA <- demoNegativeSeq$RNA.negative
seqsPro <- demoNegativeSeq$Pro.negative

# Use your own paths of the program Predator and file "stride.dat". For example:

path.Predator <- "/mnt/external_drive_1/hansy/predator/predator"
path.stride <- "/mnt/external_drive_1/hansy/predator/stride.dat"

# Pass "structureRNA.num", "structurePro", "Fourier.len", "workDir.Pro",
```

```
# "path.RNAsubopt", "path.Predator" and "path.stride" using "..." argument:

dataset <- featureStructure(seqRNA = seqsRNA, seqPro = seqsPro, label = "Non.Interact",
                            parallel.cores = 2, structureRNA.num = 6,
                            structurePro = c("ChouFasman", "DeleageRoux", "Levitt"),
                            Fourier.len = 10, workDir.Pro = "tmpDir",
                            path.RNAsubopt = "RNAsubopt", path.Predator = path.Predator,
                            path.stride = path.stride)

## End(Not run)
```

`formatSeq` *Format RNA/Protein Sequences According to the Index*

Description

This function generates a list of sequences according to the specified indices. The sequence list can be used as input for feature extraction or prediction.

Usage

```
formatSeq(idx, seqs)
```

Arguments

`idx`	specifying the sequence indices.
`seqs`	sequences loaded by function `read.fasta` from `seqinr-package`. Or a list of sequences.

Details

This function is useful for formatting the sequences using the specified indices (or names) and a sequence list. For example, the names of RNA-protein interaction pairs have been provided, but the sequences are randomly listed in one file. This function can generate a list containing the sequences whose names are listed in `idx` object. See examples below.

Value

This function returns a list.

Examples

```
data(demoIDX)
data(demoPositiveSeq)

new_RNA <- formatSeq(demoIDX$RNA_index, demoPositiveSeq$RNA.positive)
new_Pro <- formatSeq(demoIDX$Pro_index, demoPositiveSeq$Pro.positive)

names(demoPositiveSeq$Pro.positive)
names(demoPositiveSeq$RNA.positive)

names(new_RNA)
names(new_Pro)
```

```
# new_RNA and new_Pro can be further used to extract features.
```

`mod_LION` *Retrained random forest classifier of LION method*

Description

This file is required by function `run_LION` to predict ncRNA-protein interaction. Users don't need to load this file manually.

Usage

```
data(mod_LION)
```

`mod_LncADeep` *Retrained classifier of LncADeep method*

Description

This file is required by function `run_LncADeep` to predict ncRNA-protein interaction. Users don't need to load this file manually.

Usage

```
data(mod_LncADeep)
```

`mod_lncPro` *Retrained random forest classifier of lncPro method*

Description

This file is required by function `run_lncPro` to predict ncRNA-protein interaction. Users don't need to load this file manually.

Usage

```
data(mod_lncPro)
```

mod_rpiCOOL *Retrained random forest classifier of rpiCOOL method*

Description

This file is required by function `run_rpiCOOL` to predict ncRNA-protein interaction. Users don't need to load this file manually.

Usage

```
data(mod_rpiCOOL)
```

mod_RPISeq *Retrained random forest classifier of RPISeq method*

Description

This file is required by function `run_RPISeq` to predict ncRNA-protein interaction. Users don't need to load this file manually.

Usage

```
data(mod_RPISeq)
```

normalizeData *Normalize Data*

Description

This function is used to normalize dataset.

Usage

```
normalizeData(
  dataset,
  direction = c("row", "column"),
  returnNorm = TRUE,
  ignoreColumn = NULL
)
```

Arguments

`dataset`	input dataset. As a data frame.
`direction`	`"row"` or "column" to indicate the normalization direction (see details).
`returnNorm`	logical. Used for `direction = "column"`. The function will return normalized data that can be used to process new dataset (test sets, for example). Ignored when `direction = "row"`.
`ignoreColumn`	numeric or `NULL`. Input the column number of the dataset to indicate which column(s) will not be normalized. Default: `NULL`.

Details

The function provides two normalization strategies: by row or by column. If by row, the dataset will be processed with equation (see reference): d_*i* = (f_*i* - min{f_1, f_2, ...}) / max{f_1, f_2, ...}. f_1, f_2, ..., f_*i* are the original values of each row.

If by column, the dataset will be processed with: d_*i* = (f_*i* - min{f_1, f_2, ...}) / (max{f_1, f_2, ...} - min{f_1, f_2, ...}).

Value

If `direction = "column"` and `returnNorm = TRUE`, the function returns a list containing normalized dataset and normalization data. Otherwise, only return the processed dataset.

References

Shen J, Zhang J, Luo X, *et al*. Predicting protein-protein interactions based only on sequences information. Proc. Natl. Acad. Sci. U. S. A. 2007; 104:4337-41

Examples

```
data(demoPositiveSeq)
seqRNA <- demoPositiveSeq$RNA.positive
seqPro <- demoPositiveSeq$Pro.positive

dataset <- featureFreq(seqRNA = seqRNA, seqPro = seqPro, label = "Interact",
                       featureMode = "conc", computePro = "DeNovo", k.Pro = 3,
                       k.RNA = 2, normalize = "none", parallel.cores = 2)

processed_1 <- normalizeData(dataset, direction = "row", ignoreColumn = 1)
processed_2 <- normalizeData(dataset[,-1], direction = "column",
                             returnNorm = TRUE, ignoreColumn = NULL)
```

---

randomForest_CV *Evaluation of Random Forest Classifier with K-Fold Cross Validation*

Description

Evaluation of Random Forest Classifier with K-Fold Cross Validation

Usage

```
randomForest_CV(
  datasets = list(),
  label.col = 1,
  positive.class = NULL,
  folds.num = 10,
  ntree = 1500,
  seed = 1,
  parallel.cores = 2,
  ...
)
```

Arguments

datasets	a list containing one or several input datasets. See examples.
label.col	an integer. Column number of the label.
positive.class	NULL or string. Which class is the positive class? Should be one of the classes in label column. The first class in label column will be selected as the positive class if leave positive.class = NULL.
folds.num	an integer. Number of folds. Default 10 for 10-fold cross validation.
ntree	parameter for random forest. Default: 1500. See randomForest.
seed	random seed for data splitting. Integer.
parallel.cores	an integer specifying the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
...	other parameters passed to randomForest function.

Value

This function return the performance of *k*-fold CV.

See Also

randomForest_RFE, randomForest_tune, randomForest

Examples

```
# Following codes only show how to use this function
# and cannot reflect the genuine performance of tools or classifiers.

data(demoPositiveSeq)
data(demoNegativeSeq)

dataPositive <- featureFreq(seqRNA = demoPositiveSeq$RNA.positive,
                            seqPro = demoPositiveSeq$Pro.positive,
                            label = "Interact", featureMode = "conc",
                            computePro = "DeNovo", k.Pro = 3, k.RNA = 2,
                            normalize = "none", parallel.cores = 2)

dataNegative <- featureFreq(seqRNA = demoNegativeSeq$RNA.negative,
                            seqPro = demoNegativeSeq$Pro.negative,
                            label = "Non.Interact", featureMode = "conc",
                            computePro = "DeNovo", k.Pro = 3, k.RNA = 2,
                            normalize = "none", parallel.cores = 2)

dataset <- rbind(dataPositive, dataNegative)

Perf_CV <- randomForest_CV(datasets = list(dataset), label.col = 1, ntree = 100,
                           parallel.cores = 2, mtry = 20)

# if you have more than one input dataset,
# use "datasets = list(dataset1, dataset2, dataset3)".
```

randomForest_RFE *Feature Selection Using Random Forest Classifier and Recursive Feature Elimination*

Description

Feature Selection Using Random Forest Classifier and Recursive Feature Elimination

Usage

```
randomForest_RFE(
  datasets = list(),
  label.col = 1,
  positive.class = NULL,
  featureNum.range = NULL,
  folds.num = 10,
  ntree = 1500,
  seed = 1,
  parallel.cores = 2,
  ...
)
```

Arguments

datasets	should be a list containing one or several input datasets. See examples.
label.col	an integer. The number of label column.
positive.class	NULL or string. Which class is the positive class? Should be one of the classes in label column. The first class in label column will be selected as the positive class if leave positive.class = NULL.
featureNum.range	is the range of feature number in each RFE iteration. For example, if the original feature set has 100 features and featureNum.range = c(10,50,80), 20 low-ranked features will be eliminated in the first iteration, and 80 features are used to build model in the second iteration (All features are used in the first iteration). If leave NULL, RFE will iterate five times according to feature set, i.e. c(1,26,50,75,100) for 100-dimension feature set.
folds.num	an integer. Number of folds. Default 10 for 10-fold cross validation.
ntree	parameter for random forest. Default: 1500. See randomForest.
seed	random seed for data splitting. Integer.
parallel.cores	an integer specifying the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
...	other parameters passed to randomForest function.

Value

The function returns a list containing importance scores and relevant performance of the features.

See Also

randomForest_CV, randomForest_tune, randomForest

Examples

```
# Following codes only show how to use this function
# and cannot reflect the genuine performance of tools or classifiers.

data(demoPositiveSeq)
data(demoNegativeSeq)

RNA.positive <- demoPositiveSeq$RNA.positive
Pro.positive <- demoPositiveSeq$Pro.positive
RNA.negative <- demoNegativeSeq$RNA.negative
Pro.negative <- demoNegativeSeq$Pro.negative

dataPositive <- featureFreq(seqRNA = RNA.positive, seqPro = Pro.positive,
                            label = "Interact", featureMode = "conc",
                            computePro = "DeNovo", k.Pro = 3, k.RNA = 2,
                            normalize = "none", parallel.cores = 2)

dataNegative <- featureFreq(seqRNA = RNA.negative, seqPro = Pro.negative,
                            label = "Non.Interact", featureMode = "conc",
                            computePro = "DeNovo", k.Pro = 3, k.RNA = 2,
                            normalize = "none", parallel.cores = 2)

dataset <- rbind(dataPositive, dataNegative)

Perf_RFE <- randomForest_RFE(datasets = list(dataset), label.col = 1,
                             positive.class = "Interact",
                             featureNum.range = c(20, 50, 100),
                             folds.num = 5, ntree = 50, seed = 123,
                             parallel.cores = 2, mtry = 20)

# if you have more than one input dataset,
# use "datasets = list(dataset1, dataset2, dataset3)".
```

---

randomForest_tune *Determine mtry for Random Forest Classifier Using K-Fold Cross Validation*

Description

Determine mtry for Random Forest Classifier Using K-Fold Cross Validation

Usage

```
randomForest_tune(
  datasets = list(),
  label.col = 1,
  positive.class = NULL,
  folds.num = 10,
```

```
  ntree = 3000,
  mtry.ratios = c(0.1, 0.2, 0.4, 0.6, 0.8),
  seed = 1,
  return.model = TRUE,
  parallel.cores = 2,
  ...
)
```

Arguments

datasets	should be a list containing one or several input datasets. If input several datasets, stratified cross validation will be performed. See examples.
label.col	an integer. Column number of the label.
positive.class	NULL or string. Which class is the positive class? Should be one of the classes in label column. The first class in label column will be selected as the positive class if leave positive.class = NULL.
folds.num	an integer. Number of folds. Default 10 for 10-fold cross validation.
ntree	integer, number of trees to grow. See randomForest. Default: 3000.
mtry.ratios	(only when mode = "retrain") used to indicate the ratios of mtry when tuning the random forest classifier. mtry = ratio of mtry * number of features Default: c(0.1,0.2,0.4,0.6,0.8).
seed	random seed for data splitting. Integer.
return.model	logical. If TRUE, the function will return a random forest model built with the optimal ntree. The training set is the combination of all input datasets.
parallel.cores	an integer specifying the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
...	other parameters (except ntree and mtry) passed to randomForest function.

Value

If return.model = TRUR, the function returns a random forest model. If FALSE, the function returns the optimal ntree and the performance.

See Also

randomForest_RFE, randomForest_CV, randomForest

Examples

```
# Following codes only show how to use this function
# and cannot reflect the genuine performance of tools or classifiers.

data(demoPositiveSeq)
data(demoNegativeSeq)

RNA.positive <- demoPositiveSeq$RNA.positive
Pro.positive <- demoPositiveSeq$Pro.positive
RNA.negative <- demoNegativeSeq$RNA.negative
Pro.negative <- demoNegativeSeq$Pro.negative

dataPositive <- featureFreq(seqRNA = RNA.positive, seqPro = Pro.positive,
```

```
                              label = "Interact", featureMode = "conc",
                              computePro = "DeNovo", k.Pro = 3, k.RNA = 2,
                              normalize = "none", parallel.cores = 2)

dataNegative <- featureFreq(seqRNA = RNA.negative, seqPro = Pro.negative,
                              label = "Non.Interact", featureMode = "conc",
                              computePro = "DeNovo", k.Pro = 3, k.RNA = 2,
                              normalize = "none", parallel.cores = 2)

dataset <- rbind(dataPositive, dataNegative)

Perf_tune <- randomForest_tune(datasets = list(dataset), label.col = 1,
                                positive.class = "Interact", folds.num = 5,
                                ntree = 150, seed = 123,
                                return.model = TRUE, parallel.cores = 2,
                                importance = TRUE)

# if you have more than one input dataset,
# use "datasets = list(dataset1, dataset2, dataset3)".
```

`runPredator` *Run Predator in R*

Description

This function can run and capture the result of Predator in R by invoking the OS command. The results can be formatted and used as the features of tool "lncPro". Predator is required (https://bioweb.pasteur.fr/packages/pack@predator@2.1.2). NOTE: Predator does not support 64-bit MS Windows OS. Linux OS is recommended).

Usage

```
runPredator(
  seqs,
  path.Predator,
  path.stride,
  workDir = getwd(),
  verbose = FALSE,
  parallel.cores = 2,
  cl = NULL
)
```

Arguments

`seqs`	RNA sequences loaded by function `read.fasta` from `seqinr-package`. Or a list of RNA/protein sequences. RNA sequences will be converted into lower case letters. Each sequence should be a vector of single characters. (Non-AA letters will be ignored.)
`path.Predator`	string specifying the location of Predator program.
`path.stride`	string specifying the location of file "stride.dat" required by program Predator.

workDir	string specifying the directory for temporary files. The temp files will be deleted automatically when the calculation is completed. The directory does not exist will be created automatically.
verbose	logical. Should the relevant information be printed during the calculation? (Only available on Linux OS.)
parallel.cores	an integer that indicates the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
cl	parallel cores to be passed to this function.

Details

This function depends on the program Predator (https://bioweb.pasteur.fr/packages/pack@predator@2.1.2). Program Predator is only available on Linux and 32-bit Windows OS.

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

"25 of 100,length: 50 aa",

which means around 100 sequences are assigned to this node, and the program is computing the 25th sequence. The length of this sequence is 50 aa.

Value

This function returns a data frame of the raw outputs of Predator.

References

Frishman D, Argos P. Incorporation of non-local interactions in protein secondary structure prediction from the amino acid sequence. Protein Eng. 1996; 9:133-42

See Also

runRNAsubopt

Examples

```
## Not run:
data(demoPositiveSeq)
seqsPro <- demoPositiveSeq$Pro.positive

path.Predator <- "/mnt/external_drive_1/hansy/predator/predator"
path.stride <- "/mnt/external_drive_1/hansy/predator/stride.dat"

path.stride <- "/media/psf/AllFiles/Volumes/Work/Projects/ncPro/lnc-pro/predator/stride.dat"
path.Predator <- "/media/psf/AllFiles/Volumes/Work/Projects/ncPro/lnc-pro/predator/predator"

Predator.res <- runPredator(seqs = seqsPro, path.Predator = path.Predator,
                            path.stride = path.stride, workDir = "tmp",
                            verbose = TRUE, parallel.cores = 2)

## End(Not run)
```

runRNAsubopt *Run RNAsubopt in R*

Description

This function can run and capture the result of RNAsubopt (ViennaRNA Package) in R by invoking the OS command. The results can be formatted and used as the features of tool "catRAPID", "lncPro", etc. ViennaRNA Package is required (http://www.tbi.univie.ac.at/RNA/index.html).

Usage

```
runRNAsubopt(
  seqs,
  structureRNA.num = 6,
  path.RNAsubopt = "RNAsubopt",
  returnSum = FALSE,
  verbose = FALSE,
  parallel.cores = 2,
  cl = NULL
)
```

Arguments

seqs	RNA sequences loaded by function read.fasta from seqinr-package. Or a list of RNA/protein sequences. RNA sequences will be converted into lower case letters. Each sequence should be a vector of single characters.
structureRNA.num	integer. Number of random samples of suboptimal structures. Default: 6.
path.RNAsubopt	string specifying the location of RNAsubopt program.
returnSum	logical. If FALSE, the raw output of RNAsubopt (Dot-Bracket Notation of RNA structure) will be returned. If TRUE, the results of RNAsubopt will be converted into numeric sequence: In any RNA structure sequence (Dot-Bracket Notation), "." (Dot) will be replaced with 0; "(" and ")" (Bracket) will be replaced with 1. And *structureRNA.num* binary sequences will be added to obtain a new numeric sequence.
verbose	logical. Should the relevant information be printed during the calculation? (Only available on UNIX/Linux.)
parallel.cores	an integer that indicates the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
cl	parallel cores to be passed to this function.

Details

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

Parameter path.RNAsubopt can be simply defined as "RNAsubopt" as default when the OS is UNIX/Linux. However, for some OS, such as Windows, users need to specify the path.RNAsubopt

if the path of "RNAsubopt" haven't been added in environment variables (e.g. `path.RNAsubopt = '"C:/Program Files/ViennaRNA/RNAsubopt.exe"'`).

This function can print the related information when running on Linux OS, such as:

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

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

Value

This function returns a data frame that contains the raw outputs when `returnSum = FALSE`. Or a list of numeric results of RNAsubopt when `returnSum = TRUE`.

References

Lorenz R, Bernhart SH, Honer zu Siederdissen C, *et al*. ViennaRNA Package 2.0. Algorithms Mol. Biol. 2011; 6:26

See Also

`runPredator`

Examples

```
## Not run:
data(demoPositiveSeq)
seqsRNA <- demoPositiveSeq$RNA.positive

path.RNAsubopt <- "RNAsubopt" # you need to use your own paths, for example:
# path.RNAsubopt <- '"C:/Program Files (x86)/ViennaRNA Package/RNAsubopt.exe"' # in Windows OS

RNAsubopt_1 <- runRNAsubopt(seqs = seqsRNA, path.RNAsubopt = path.RNAsubopt,
                            returnSum = FALSE, verbose = TRUE,
                            parallel.cores = 2)

RNAsubopt_2 <- runRNAsubopt(seqs = seqsRNA, structureRNA.num = 6,
                            path.RNAsubopt = path.RNAsubopt,
                            returnSum = TRUE, parallel.cores = 2)

## End(Not run)
```

---

run_confidentPrediction

Confident Prediction of RNA-Protein Interaction Using Multiple Methods Simultaneously

---

Description

This function can predict lncRNA/RNA-protein interactions using all supported methods, which is useful to have a high-confident prediction.

Usage

```
run_confidentPrediction(
  seqRNA,
  seqPro,
  label = NULL,
 methods = c("RPISeq_web", "RPISeq_retrain", "lncPro_original", "lncPro_retrain",
    "rpiCOOL_retrain", "LncADeep_retrain", "LION"),
  RPISeq.mod = NULL,
  lncPro.mod = NULL,
  rpiCOOL.mod = NULL,
  LncADeep.mod = NULL,
  LION.mod = NULL,
  parallel.cores = 2,
  cl = NULL
)
```

Arguments

seqRNA	RNA sequences loaded by function read.fasta from seqinr-package. Or a list of RNA/protein sequences. RNA sequences will be converted into lower case letters.
seqPro	protein sequences loaded by function read.fasta from seqinr-package. Or a list of protein sequences. Protein sequences will be converted into upper case letters. Each sequence should be a vector of single characters.
label	optional. A string or a vector of strings or NULL. Used to give labels or notes to the output result. Default: NULL.
methods	strings. Indicate the method(s) to be used for prediction. Can be: "RPISeq_web", "RPISeq_retrain", "lncPro_original", "lncPro_retrain", "rpiCOOL_retrain", "LncADeep_retrain" and "LION".
RPISeq.mod, lncPro.mod, rpiCOOL.mod, LncADeep.mod, LION.mod	use default retrained model (if NULL) or assign a new retrained model? New retrained model can be generated with run_RPISeq, run_lncPro, run_rpiCOOL, run_LncADeep and LION.
parallel.cores	an integer that indicates the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
cl	parallel cores to be passed to this function.

Value

A list containing the predicted results.

References

Han S, *et al*. LION: an integrated R package for effective ncRNA-protein interaction prediction. (*Submitted*)

Examples

```
#' # Following codes only show how to use this function
# and cannot reflect the genuine performance of tools or classifiers.
```

```
data(demoPositiveSeq)
seqRNA <- demoPositiveSeq$RNA.positive
seqPro <- demoPositiveSeq$Pro.positive

# Using methods RPISeq (retrained model) and rpiCOOL (retrained model):

Res_confidence <- run_confidentPrediction(seqRNA = seqRNA, seqPro = seqPro,
                                  methods = c("RPISeq_retrain", "rpiCOOL_retrain", "LION"),
                                            label = "Interact", # label is optional
                                            parallel.cores = 2)
# Convert to data frame:

Res_confidence_df <- do.call("cbind", Res_confidence)
Res_confidence_df <- Res_confidence_df[!duplicated(names(Res_confidence_df))]
```

`run_LION` *Predict RNA-Protein Interaction Using LION Method*

Description

This function can predict lncRNA/RNA-protein interactions using LION method. Model retraining and feature extraction are also supported.

Usage

```
run_LION(
  seqRNA,
  seqPro,
  mode = c("prediction", "retrain", "feature"),
  retrained.model = NULL,
  label = NULL,
  positive.class = NULL,
  folds.num = 10,
  ntree = 3000,
  mtry.ratios = c(0.1, 0.2, 0.4, 0.6, 0.8),
  seed = 1,
  parallel.cores = 2,
  cl = NULL,
  ...
)
```

Arguments

`seqRNA`	RNA sequences loaded by function `read.fasta` from `seqinr-package`. Or a list of RNA/protein sequences. RNA sequences will be converted into lower case letters.
`seqPro`	protein sequences loaded by function `read.fasta` from `seqinr-package`. Or a list of protein sequences. Protein sequences will be converted into upper case letters. Each sequence should be a vector of single characters.

mode	a string. Set "prediction" to predict ncRNA-protein pairs and return prediction results; set "retrain" to build a new random forest model using the input data; set "feature" to return a data frame contains the extracted features. Users can use the extracted features generated by mode = "feature" to train classifiers with other machine learning algorithms. Default: "prediction".
retrained.model	(only when mode = "prediction") use the default model or a new retrained model to predict ncRNA-protein pairs? If NULL, default machine learning model will be used. Or pass the model generated by this function with parameter "mode = retrain". Default: NULL. See examples below.
label	a string or a vector of strings or NULL. Optional when mode = "prediction" or mode = "feature": used to give labels or notes to the output result. Required when mode = "retrain": must be a vector of strings that corresponds to input sequences. Each string indicates the class of each input pair. Default: NULL.
positive.class	(only when mode = "retrain") NULL or a string used to indicate which class is the positive class, Should be one of the classes in label or leave positive.class = NULL. In the latter case, the first class in label will be used as the positive class. Default: NULL.
folds.num	(only when mode = "retrain") an integer indicates the number of folds for cross validation. Default: 10 for 10-fold cross validation.
ntree	integer, number of trees to grow. See randomForest. Default: 3000.
mtry.ratios	(only when mode = "retrain") used to indicate the ratios of mtry when tuning the random forest classifier. mtry = ratio of mtry * number of features Default: c(0.1,0.2,0.4,0.6,0.8).
seed	(only when mode = "retrain") an integer indicates the random seed for data splitting.
parallel.cores	an integer that indicates the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
cl	parallel cores to be passed to this function.
...	(only when mode = "retrain") other parameters (except ntree and mtry) passed to randomForest function.

Value

If mode = "prediction", this function returns a data frame that contains the predicted results.

If mode = "retrain", this function returns a random forest classifier.

If mode = "feature", this function returns a data frame that contains the extracted features.

References

Han S, *et al*. LION: an integrated R package for effective ncRNA-protein interaction prediction. (*Submitted*)

Examples

```
# Following codes only show how to use this function
# and cannot reflect the genuine performance of tools or classifiers.

data(demoPositiveSeq)
```

```
seqRNA <- demoPositiveSeq$RNA.positive
seqPro <- demoPositiveSeq$Pro.positive

# Predicting ncRNA-protein pairs:

Res_LION_1 <- run_LION(seqRNA = seqRNA, seqPro = seqPro,
                       parallel.cores = 2) # using the default setting

# the above command is equivalent to:
Res_LION_2 <- run_LION(seqRNA = seqRNA, seqPro = seqPro, mode = "prediction",
                       retrained.model = NULL, label = NULL,
                       parallel.cores = 2)

# Train a new model:

# Argument "label" which indicates the class of each input pair is required here.
# "label" should correspond to the classes of "seqRNA" and "seqPro".
# "positive.class" should be one of the classes in argument "label" or can be set as "NULL".
# In the latter case, the first label in "label" will be used as the positive class.
# Parameters of random forest, such as "replace", can be passed using "..." argument.

LION_model <- run_LION(seqRNA = seqRNA, seqPro = seqPro, mode = "retrain",
                       label = rep(c("Interact", "Non.Interact"), each = 10),
                       positive.class = NULL, folds.num = 5, ntree = 100,
                       seed = 1, parallel.cores = 2, replace = FALSE)

# Predicting using new built model by setting "retrained.model = LION_model":

Res_LION_2 <- run_LION(seqRNA = seqRNA, seqPro = seqPro, mode = "prediction",
                       retrained.model = LION_model,
                       label = rep(c("Interact", "Non.Interact"), each = 10),
                       parallel.cores = 2)

# Only extracting features:

LION_feature_df <- run_LION(seqRNA = seqRNA, seqPro = seqPro, mode = "feature",
                            label = "LION_feature", parallel.cores = 2)

# Extracted features can be used to build classifiers using other machine learning
# algorithms, which provides users with more flexibility.
```

`run_LncADeep` *Predict RNA-Protein Interaction Using LncADeep's Features*

Description

This function can predict lncRNA/RNA-protein interactions using rebuilt model trained with LncADeep's feature set. Model retraining and feature extraction are also supported. LncADeep selects 110 features to build its classifier. Here, the 110 top features are determined by averaging feature scores of 33 evaluation results provided by LncADeep. LncADeep's original model is trained using deep neural network (DNN). Considering that DNN architecture is hard to perform parameter tuning, we rebuild the model using the same machine algorithm (random forest) as the other methods. Users can build DNN model with the features generated by this function.

Usage

```
run_LncADeep(
  seqRNA,
  seqPro,
  mode = c("prediction", "retrain", "feature"),
  retrained.model = NULL,
  label = NULL,
  positive.class = NULL,
  folds.num = 10,
  ntree = 3000,
  mtry.ratios = c(0.1, 0.2, 0.4, 0.6, 0.8),
  seed = 1,
  parallel.cores = 2,
  cl = NULL,
  ...
)
```

Arguments

seqRNA	RNA sequences loaded by function read.fasta from seqinr-package. Or a list of RNA/protein sequences. RNA sequences will be converted into lower case letters.
seqPro	protein sequences loaded by function read.fasta from seqinr-package. Or a list of protein sequences. Protein sequences will be converted into upper case letters. Each sequence should be a vector of single characters.
mode	a string. Set "prediction" to predict ncRNA-protein pairs and return prediction results; set "retrain" to build a new random forest model using the input data; set "feature" to return a data frame contains the extracted features. Users can use the extracted features generated by mode = "feature" to train classifiers with other machine learning algorithms. Default: "prediction".
retrained.model	(only when mode = "prediction") use the default model or a new retrained model to predict ncRNA-protein pairs? If NULL, default machine learning model will be used. Or pass the model generated by this function with parameter "mode = retrain". Default: NULL. See examples below.
label	a string or a vector of strings or NULL. Optional when mode = "prediction" or mode = "feature": used to give labels or notes to the output result. Required when mode = "retrain": must be a vector of strings that corresponds to input sequences. Each string indicates the class of each input pair. Default: NULL.
positive.class	(only when mode = "retrain") NULL or a string used to indicate which class is the positive class, Should be one of the classes in label or leave positive.class = NULL. In the latter case, the first class in label will be used as the positive class. Default: NULL.
folds.num	(only when mode = "retrain") an integer indicates the number of folds for cross validation. Default: 10 for 10-fold cross validation.
ntree	integer, number of trees to grow. See randomForest. Default: 3000.
mtry.ratios	(only when mode = "retrain") used to indicate the ratios of mtry when tuning the random forest classifier. mtry = ratio of mtry * number of features Default: c(0.1,0.2,0.4,0.6,0.8).

seed	(only when mode = "retrain") an integer indicates the random seed for data splitting.
parallel.cores	an integer that indicates the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
cl	parallel cores to be passed to this function.
...	(only when mode = "retrain") other parameters (except ntree and mtry) passed to randomForest function.

Value

If mode = "prediction", this function returns a data frame that contains the predicted results.

If mode = "retrain", this function returns a random forest classifier.

If mode = "feature", this function returns a data frame that contains the extracted features.

References

Yang C, Yang L, Zhou M, *et al*. LncADeep: an ab initio lncRNA identification and functional annotation tool based on deep learning. Bioinformatics. 2018; 34(22):3825-3834.

Examples

```
# Following codes only show how to use this function
# and cannot reflect the genuine performance of tools or classifiers.

data(demoPositiveSeq)
seqRNA <- demoPositiveSeq$RNA.positive
seqPro <- demoPositiveSeq$Pro.positive

# Predicting ncRNA-protein pairs:

Res_LncADeep_1 <- run_LncADeep(seqRNA = seqRNA, seqPro = seqPro, mode = "prediction",
                               retrained.model = NULL, label = "LncADeep_res",
                               parallel.cores = 2) # using default rebuilt model

# Train a new model:

# Argument "label" which indicates the class of each input pair is required here.
# "label" should correspond to the classes of "seqRNA" and "seqPro".
# "positive.class" should be one of the classes in argument "label" or can be set as "NULL".
# In the latter case, the first label in "label" will be used as the positive class.
# Parameters of random forest, such as "nodesize", can be passed using "..." argument.

LncADeep_model <- run_LncADeep(seqRNA = seqRNA, seqPro = seqPro, mode = "retrain",
                               label = rep(c("Interact", "Non.Interact"), each = 10),
                               positive.class = NULL, folds.num = 5, ntree = 100,
                               seed = 1, parallel.cores = 2, nodesize = 2)

# Predicting using new built model by setting "retrained.model = LncADeep_model":

Res_LncADeep_2 <- run_LncADeep(seqRNA = seqRNA, seqPro = seqPro, mode = "prediction",
                               retrained.model = LncADeep_model, label = NULL,
                               parallel.cores = 2)
```

```
# Only extracting features:

LncADeep_feature_df <- run_LncADeep(seqRNA = seqRNA, seqPro = seqPro, mode = "feature",
                                    label = "feature", parallel.cores = 2)

# Extracted features can be used to build classifiers using other machine learning
# algorithms, which provides users with more flexibility.
```

---

run_lncPro *Predict RNA-Protein Interaction Using lncPro Method*

Description

This function can predict lncRNA/RNA-protein interactions using lncPro method. Model retraining and feature extraction are also supported. Programs "RNAsubopt" from software "ViennaRNA Package" and "Predator" is required. Please also note that "Predator" is only available on UNIX/Linux and 32-bit Windows OS.

Usage

```
run_lncPro(
  seqRNA,
  seqPro,
  mode = c("prediction", "retrain", "feature"),
  path.RNAsubopt = "RNAsubopt",
  path.Predator = "Predator/predator",
  path.stride = "Predator/stride.dat",
  workDir.Pro = getwd(),
  prediction = c("original", "retrained"),
  retrained.model = NULL,
  label = NULL,
  positive.class = NULL,
  folds.num = 10,
  ntree = 3000,
  mtry.ratios = c(0.1, 0.2, 0.4, 0.6, 0.8),
  seed = 1,
  parallel.cores = 2,
  cl = NULL,
  ...
)
```

Arguments

seqRNA	RNA sequences loaded by function read.fasta from seqinr-package. Or a list of RNA/protein sequences. RNA sequences will be converted into lower case letters.
seqPro	protein sequences loaded by function read.fasta from seqinr-package. Or a list of protein sequences. Protein sequences will be converted into upper case letters. Each sequence should be a vector of single characters.

mode	a string. Set "prediction" to predict ncRNA-protein pairs and return prediction results; set "retrain" to build a new random forest model using the input data; set "feature" to return a data frame contains the extracted features. Users can use the extracted features generated by mode = "feature" to train classifiers with other machine learning algorithms. Default: "prediction".
path.RNAsubopt	a string specifying the location of "RNAsubopt" program.
path.Predator	a string specifying the location of "Predator" program.
path.stride	a string specifying the location of file "stride.dat" required by program Predator.
workDir.Pro	a string specifying the directory for temporary files used for process protein sequences. The temp files will be deleted automatically when the calculation is completed. If the directory does not exist, it will be created automatically.
prediction	(only when mode = "prediction") set "original" to use original lncPro algorithm, or set "retrained" to call retrained model. The retrained model is constructed with the same features as the original version, but random classifier is employed to build the classifier.
retrained.model	(only when mode = "prediction" and prediction = "retrained") use the default model or a new retrained model to predict ncRNA-protein pairs? If NULL, default machine learning model will be used. Or pass the model generated by this function with parameter "mode = retrain". Default: NULL. See examples below.
label	a string or a vector of strings or NULL. Optional when mode = "prediction" or mode = "feature": used to give labels or notes to the output result. Required when mode = "retrain": must be a vector of strings that corresponds to input sequences. Each string indicates the class of each input pair. Default: NULL.
positive.class	(only when mode = "retrain") NULL or a string used to indicate which class is the positive class, Should be one of the classes in label or leave positive.class = NULL. In the latter case, the first class in label will be used as the positive class. Default: NULL.
folds.num	(only when mode = "retrain") an integer indicates the number of folds for cross validation. Default: 10 for 10-fold cross validation.
ntree	integer, number of trees to grow. See randomForest. Default: 3000.
mtry.ratios	(only when mode = "retrain") used to indicate the ratios of mtry when tuning the random forest classifier. mtry = ratio of mtry * number of features Default: c(0.1,0.2,0.4,0.6,0.8).
seed	(only when mode = "retrain") an integer indicates the random seed for data splitting.
parallel.cores	an integer that indicates the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
cl	parallel cores to be passed to this function.
...	(only when mode = "retrain") other parameters (except ntree and mtry) passed to randomForest function.

Details

The method is proposed by lncPro. This function, runlncPro, has improved and fixed the original code.

runlncPro depends on the program "RNAsubopt" of software "ViennaRNA" (http://www.tbi.univie.ac.at/RNA/index.html) and "Predator" (https://bioweb.pasteur.fr/packages/pack@predator@2.1.2).

Parameter path.RNAsubopt can be simply defined as "RNAsubopt" as default when the OS is UNIX/Linux. However, for some OS, such as Windows, users may need to specify the path.RNAsubopt if the path of "RNAsubopt" haven't been added in environment variables (e.g. path.RNAsubopt = '"C:/Program Files/ViennaRNA/RNAsubopt.exe"').

Program "Predator" is only available on UNIX/Linux and 32-bit Windows OS.

Value

If mode = "prediction", this function returns a data frame that contains the predicted results.

If mode = "retrain", this function returns a random forest classifier.

If mode = "feature", this function returns a data frame that contains the extracted features.

References

Lu Q, Ren S, Lu M, *et al*. Computational prediction of associations between long non-coding RNAs and proteins. BMC Genomics 2013; 14:651

Examples

```
## Not run:

# Following codes only show how to use this function
# and cannot reflect the genuine performance of tools or classifiers.

data(demoPositiveSeq)
seqRNA <- demoPositiveSeq$RNA.positive
seqPro <- demoPositiveSeq$Pro.positive

# Predicting ncRNA-protein pairs (you need to use your own paths):

path.RNAsubopt <- "RNAsubopt"
path.Predator <- "/mnt/external_drive_1/hansy/predator/predator"
path.stride <- "/mnt/external_drive_1/hansy/predator/stride.dat"
workDir.Pro <- "tmp"

Res_lncPro_1 <- run_lncPro(seqRNA = seqRNA, seqPro = seqPro, mode = "prediction",
                           path.RNAsubopt = path.RNAsubopt, path.Predator = path.Predator,
                             path.stride = path.stride, workDir.Pro = workDir.Pro,
                             prediction = "original", label = "lncPro_original",
                             parallel.cores = 10) # using original algorithm

Res_lncPro_2 <- run_lncPro(seqRNA = seqRNA, seqPro = seqPro, mode = "prediction",
                           path.RNAsubopt = path.RNAsubopt, path.Predator = path.Predator,
                             path.stride = path.stride, workDir.Pro = workDir.Pro,
                             prediction = "retrained", retrained.model = NULL,
                             label = "lncPro_retrained",
                             parallel.cores = 10) # using default rebuilt model

# Train a new model:

# Argument "label" which indicates the class of each input pair is required here.
# "label" should correspond to the classes of "seqRNA" and "seqPro".
```

```
# "positive.class" should be one of the classes in argument "label" or can be set as "NULL".
# In the latter case, the first label in "label" will be used as the positive class.
# Parameters of random forest, such as "replace", "nodesize", can be passed using "..." argument.

lncPro_model = run_lncPro(seqRNA = seqRNA, seqPro = seqPro, mode = "retrain",
                          path.RNAsubopt = path.RNAsubopt, path.Predator = path.Predator,
                           path.stride = path.stride, workDir.Pro = workDir.Pro,
                           label = rep(c("Interact", "Non.Interact"), each = 10),
                           positive.class = NULL, folds.num = 10,
                           ntree = 100, seed = 1, parallel.cores = 2, replace = FALSE)

# Predicting using new built model by setting "retrained.model = lncPro_model":

Res_lncPro_3 <- run_lncPro(seqRNA = seqRNA, seqPro = seqPro, mode = "prediction",
                          path.RNAsubopt = path.RNAsubopt, path.Predator = path.Predator,
                            path.stride = path.stride, workDir.Pro = workDir.Pro,
                            prediction = "retrained", retrained.model = lncPro_model,
                            label = rep(c("Interact", "Non.Interact"), each = 10),
                            parallel.cores = 10)

# Only extracting features:

lncPro_feature_df <- run_lncPro(seqRNA = seqRNA, seqPro = seqPro,
                                mode = "feature", path.RNAsubopt = path.RNAsubopt,
                               path.Predator = path.Predator, path.stride = path.stride,
                                workDir.Pro = workDir.Pro, label = "Interact",
                                parallel.cores = 10)

# Extracted features can be used to build classifiers using other machine learning
# algorithms, which provides users with more flexibility.


## End(Not run)
```

`run_rpiCOOL` *Predict RNA-Protein Interaction Using rpiCOOL's Features*

Description

This function can predict lncRNA/RNA-protein interactions using rebuilt model trained with rpiCOOL's feature set. Model retraining and feature extraction are also supported. The codes of this function slightly differ from rpiCOOL's script.

Usage

```
run_rpiCOOL(
  seqRNA,
  seqPro,
  mode = c("prediction", "retrain", "feature"),
  retrained.model = NULL,
  label = NULL,
  positive.class = NULL,
  folds.num = 10,
```

```
  ntree = 3000,
  mtry.ratios = c(0.1, 0.2, 0.4, 0.6, 0.8),
  seed = 1,
  parallel.cores = 2,
  cl = NULL,
  ...
)
```

Arguments

seqRNA	RNA sequences loaded by function read.fasta from seqinr-package. Or a list of RNA/protein sequences. RNA sequences will be converted into lower case letters.
seqPro	protein sequences loaded by function read.fasta from seqinr-package. Or a list of protein sequences. Protein sequences will be converted into upper case letters. Each sequence should be a vector of single characters.
mode	a string. Set "prediction" to predict ncRNA-protein pairs and return prediction results; set "retrain" to build a new random forest model using the input data; set "feature" to return a data frame contains the extracted features. Users can use the extracted features generated by mode = "feature" to train classifiers with other machine learning algorithms. Default: "prediction".
retrained.model	(only when mode = "prediction") use the default model or a new retrained model to predict ncRNA-protein pairs? If NULL, default machine learning model will be used. Or pass the model generated by this function with parameter "mode = retrain". Default: NULL. See examples below.
label	a string or a vector of strings or NULL. Optional when mode = "prediction" or mode = "feature": used to give labels or notes to the output result. Required when mode = "retrain": must be a vector of strings that corresponds to input sequences. Each string indicates the class of each input pair. Default: NULL.
positive.class	(only when mode = "retrain") NULL or a string used to indicate which class is the positive class, Should be one of the classes in label or leave positive.class = NULL. In the latter case, the first class in label will be used as the positive class. Default: NULL.
folds.num	(only when mode = "retrain") an integer indicates the number of folds for cross validation. Default: 10 for 10-fold cross validation.
ntree	integer, number of trees to grow. See randomForest. Default: 3000.
mtry.ratios	(only when mode = "retrain") used to indicate the ratios of mtry when tuning the random forest classifier. mtry = ratio of mtry * number of features Default: c(0.1,0.2,0.4,0.6,0.8).
seed	(only when mode = "retrain") an integer indicates the random seed for data splitting.
parallel.cores	an integer that indicates the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
cl	parallel cores to be passed to this function.
...	(only when mode = "retrain") other parameters (except ntree and mtry) passed to randomForest function.

Value

If mode = "prediction", this function returns a data frame that contains the predicted results.

If mode = "retrain", this function returns a random forest classifier.

If mode = "feature", this function returns a data frame that contains the extracted features.

References

Akbaripour-Elahabad M, Zahiri J, Rafeh R, *et al*. rpiCOOL: A tool for In Silico RNA-protein interaction detection using random forest. J. Theor. Biol. 2016; 402:1-8

Examples

```
# Following codes only show how to use this function
# and cannot reflect the genuine performance of tools or classifiers.

data(demoPositiveSeq)
seqRNA <- demoPositiveSeq$RNA.positive
seqPro <- demoPositiveSeq$Pro.positive

# Predicting ncRNA-protein pairs:

Res_rpiCOOL_1 <- run_rpiCOOL(seqRNA = seqRNA, seqPro = seqPro, mode = "prediction",
                             retrained.model = NULL, label = "rpiCOOL_res",
                             parallel.cores = 2) # using default rebuilt model

# Train a new model:

# Argument "label" which indicates the class of each input pair is required here.
# "label" should correspond to the classes of "seqRNA" and "seqPro".
# "positive.class" should be one of the classes in argument "label" or can be set as "NULL".
# In the latter case, the first label in "label" will be used as the positive class.
# Parameters of random forest, such as "replace", can be passed using "..." argument.

rpiCOOL_model = run_rpiCOOL(seqRNA = seqRNA, seqPro = seqPro, mode = "retrain",
                            label = rep(c("Interact", "Non.Interact"), each = 10),
                            positive.class = NULL, folds.num = 5, ntree = 300,
                            seed = 1, parallel.cores = 2, replace = FALSE)

# Predicting using new built model by setting "retrained.model = rpiCOOL_model":

Res_rpiCOOL_2 <- run_rpiCOOL(seqRNA = seqRNA, seqPro = seqPro, mode = "prediction",
                             retrained.model = rpiCOOL_model, label = NULL,
                             parallel.cores = 2)

# Only extracting features:

rpiCOOL_feature_df <- run_rpiCOOL(seqRNA = seqRNA, seqPro = seqPro, mode = "feature",
                                  label = "feature", parallel.cores = 2)

# Extracted features can be used to build classifiers using other machine learning
# algorithms, which provides users with more flexibility.
```

run_RPISeq *Predict RNA-Protein Interaction Using RPISeq Method*

Description

This function can predict lncRNA/RNA-protein interactions using RPISeq method. Both the web-based original version and retrained model are available. Network is required to use the original version. Model retraining and feature extraction are also supported.

Usage

```
run_RPISeq(
  seqRNA,
  seqPro,
  mode = c("prediction", "retrain", "feature"),
  prediction = c("web", "retrained"),
  retrained.model = NULL,
  label = NULL,
  positive.class = NULL,
  folds.num = 10,
  ntree = 3000,
  mtry.ratios = c(0.1, 0.2, 0.4, 0.6, 0.8),
  seed = 1,
  parallel.cores = 2,
  cl = NULL,
  ...
)
```

Arguments

seqRNA	RNA sequences loaded by function read.fasta from seqinr-package. Or a list of RNA/protein sequences. RNA sequences will be converted into lower case letters.
seqPro	protein sequences loaded by function read.fasta from seqinr-package. Or a list of protein sequences. Protein sequences will be converted into upper case letters. Each sequence should be a vector of single characters.
mode	a string. Set "prediction" to predict ncRNA-protein pairs and return prediction results; set "retrain" to build a new random forest model using the input data; set "feature" to return a data frame contains the extracted features. Users can use the extracted features generated by mode = "feature" to train classifiers with other machine learning algorithms. Default: "prediction".
prediction	(only when mode = "prediction") set "web" to use original web-based RPISeq algorithm (network is required), or set "retrained" to call retrained model.
retrained.model	(only when mode = "prediction" and prediction = "retrained") use the default model or a new retrained model to predict ncRNA-protein pairs? If NULL, default machine learning model will be used. Or pass the model generated by this function with parameter "mode = retrain". Default: NULL. See examples below.

label	a string or a vector of strings or NULL. Optional when mode = "prediction" or mode = "feature": used to give labels or notes to the output result. Required when mode = "retrain": must be a vector of strings that corresponds to input sequences. Each string indicates the class of each input pair. Default: NULL.
positive.class	(only when mode = "retrain") NULL or a string used to indicate which class is the positive class, Should be one of the classes in label or leave positive.class = NULL. In the latter case, the first class in label will be used as the positive class. Default: NULL.
folds.num	(only when mode = "retrain") an integer indicates the number of folds for cross validation. Default: 10 for 10-fold cross validation.
ntree	integer, number of trees to grow. See randomForest. Default: 3000.
mtry.ratios	(only when mode = "retrain") used to indicate the ratios of mtry when tuning the random forest classifier. mtry = ratio of mtry * number of features Default: c(0.1,0.2,0.4,0.6,0.8).
seed	(only when mode = "retrain") an integer indicates the random seed for data splitting.
parallel.cores	an integer that indicates the number of cores for parallel computation. Default: 2. Set parallel.cores = -1 to run with all the cores. parallel.cores should be == -1 or >= 1.
cl	parallel cores to be passed to this function.
...	(only when mode = "retrain") other parameters (except ntree and mtry) passed to randomForest function.

Value

If mode = "prediction", this function returns a data frame that contains the predicted results.

If mode = "retrain", this function returns a random forest classifier.

If mode = "feature", this function returns a data frame that contains the extracted features.

References

Muppirala UK, Honavar VG, Dobbs D. Predicting RNA-protein interactions using only sequence information. BMC Bioinformatics 2011; 12:489

Examples

```
# Following codes only show how to use this function
# and cannot reflect the genuine performance of tools or classifiers.

data(demoPositiveSeq)
seqRNA <- demoPositiveSeq$RNA.positive
seqPro <- demoPositiveSeq$Pro.positive

# Predicting ncRNA-protein pairs:

Res_RPISeq_1 <- run_RPISeq(seqRNA = seqRNA, seqPro = seqPro, mode = "prediction",
                           label = "Interact", prediction = "web",
                           parallel.cores = 2) # using web server

Res_RPISeq_2 <- run_RPISeq(seqRNA = seqRNA, seqPro = seqPro, mode = "prediction",
                           prediction = "retrained", retrained.model = NULL,
```

```
                          parallel.cores = 2) # using default rebuilt model

# Train a new model:

# Argument "label" which indicates the class of each input pair is required here.
# "label" should correspond to the classes of "seqRNA" and "seqPro".
# "positive.class" should be one of the classes in argument "label" or can be set as "NULL".
# In the latter case, the first label in "label" will be used as the positive class.
# Parameters of random forest, such as "nodesize", can be passed using "..." argument.

RPI_model <- run_RPISeq(seqRNA = seqRNA, seqPro = seqPro, mode = "retrain",
                        label = rep(c("Interact", "Non.Interact"), each = 10),
                        positive.class = "Interact", folds.num = 5,
                        ntree = 300, seed = 1, parallel.cores = 2, nodesize = 2)

# Predicting using new built model by setting "retrained.model = RPI_model":

Res_RPISeq_3 <- run_RPISeq(seqRNA = seqRNA, seqPro = seqPro, mode = "prediction",
                           prediction = "retrained", retrained.model = RPI_model,
                           label = rep(c("Interact", "Non.Interact"), each = 10),
                           parallel.cores = 2)

# Only extracting features:

RPISeq_feature_df <- run_RPISeq(seqRNA = seqRNA, seqPro = seqPro, mode = "feature",
                                label = "Interact", parallel.cores = 2)

# Extracted features can be used to build classifiers using other machine learning
# algorithms, which provides users with more flexibility.
```

Index

aaindex, *12*

computeFreq, 2, *18, 19*
computeMLC, 5
computeMotifs, 6, *20–22*
computePhysChem, 9, *12*, *22–24*
computePhysChem_AAindex, 11
computeStructure, 13, *24, 25*
confusionMatrix, *17*

demoIDX, 16
demoNegativeSeq, 16
demoPositiveSeq, 16

evaluatePrediction, 17

featureFreq, *4*, 18
featureMotifs, *8*, 20
featurePhysChem, *11*, 22
featureStructure, *15*, 24
formatSeq, *16*, 26

LION, *38*

mod_LION, 27
mod_LncADeep, 27
mod_lncPro, 27
mod_rpiCOOL, 28
mod_RPISeq, 28

normalizeData, 28

randomForest, *30–33*, *40*, *42, 43*, *45*, *48*, *51*
randomForest_CV, 29, *32, 33*
randomForest_RFE, *30*, 31, *33*
randomForest_tune, *30*, *32*, 32
read.fasta, *3*, *5, 6*, *9*, *12*, *14*, *18*, *20, 21*, *23, 24*, *26*, *34*, *36*, *38, 39*, *42*, *44*, *48*, *50*
run_confidentPrediction, 37
run_LION, *27*, 39
run_LncADeep, *27*, *38*, 41
run_lncPro, *27*, *38*, 44
run_rpiCOOL, *28*, *38*, 47
run_RPISeq, *28*, *38*, 50
runPredator, *15*, *25*, 34, *37*
runRNAsubopt, *15*, *25*, *35*, 36