Package ‘ftrCOOL’

November 30, 2021

Type Package

Title Feature Extraction from Biological Sequences

Version 2.0.0

Author Sare Amerifar

Maintainer Sare Amerifar <sare.amerifi.01@gmail.com>

Description Extracts features from biological sequences. It contains most features which are presented in related work and also includes features which have never been introduced before. It extracts numerous features from nucleotide and peptide sequences. Each feature converts the input sequences to discrete numbers in order to use them as predictors in machine learning models. There are many features and information which are hidden inside a sequence. Utilizing the package, users can convert biological sequences to discrete models based on chosen properties. References: 'iLearn' Z. Chen et al.' (2019) <DOI:10.1093/bib/bbz041>. 'iFeature' Z. Chen et al.' (2018) <DOI:10.1093/bioinformatics/bty140>. <https://CRAN.R-project.org/package=rDNAse>. 'PseKRAAC' Y. Zuo et al.'PseKRAAC: a flexible web server for generating pseudo K-tuple reduced amino acids composition' (2017) <DOI:10.1093/bioinformatics/btw564>. 'iDNA6mA-PseKNC' P. Feng et al. 'iDNA6mA-PseKNC: Identifying DNA N6-methyladenosine sites by incorporating nucleotide physicochemical properties into PseKNC' (2019) <DOI:10.1016/j.jgyen.2018.01.005>. 'I. Dubchak et al.'Prediction of protein folding class using global description of amino acid sequence' (1995) <DOI:10.1073/pnas.92.19.8700>. 'W. Chen et al.'Identification and analysis of the N6-methyladenosine in the Saccharomyces cerevisiae transcriptome' (2015) <DOI:10.1038/srep13859>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.1

Imports stats, utils

Suggests testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2021-11-29 23:10:02 UTC
R topics documented:

AA2Binary ................................................................. 5
AAindex ................................................................. 7
AAKpartComposition ................................................... 8
AAutoCor ................................................................. 9
AESNN3 ................................................................. 11
alphabetCheck .......................................................... 12
ANF_DNA ............................................................... 13
ANF_RNA ............................................................... 14
APAAPC ................................................................. 15
APkNUCdi_DNA .......................................................... 16
APkNUCdi_RNA .......................................................... 17
APkNUCtri_DNA .......................................................... 19
ASA ................................................................. 20
ASDC ................................................................. 21
ASDC_DNA ............................................................. 22
ASDC_RNA ............................................................. 23
AutoCorDiNUC_DNA ...................................................... 24
AutoCorDiNUC_RNA ...................................................... 25
AutoCorTriNUC_DNA ...................................................... 27
binary_3bit_T1 ........................................................... 28
binary_3bit_T2 ........................................................... 30
binary_3bit_T3 ........................................................... 31
binary_3bit_T4 ........................................................... 32
binary_3bit_T5 ........................................................... 34
binary_3bit_T6 ........................................................... 35
binary_3bit_T7 ........................................................... 36
binary_5bit_T1 ........................................................... 38
binary_5bit_T2 ........................................................... 39
binary_6bit ............................................................. 41
BLOSUM62 ............................................................. 42
CkSAApair .............................................................. 43
CkSGAApair .............................................................. 44
CkSNUCpair_DNA ......................................................... 46
CkSNUCpair_RNA ......................................................... 47
codonAdaptionIndex .................................................... 48
CodonFraction .......................................................... 49
CodonUsage_DNA ......................................................... 50
CodonUsage_RNA ......................................................... 51
conjointTriad ........................................................... 51
conjointTriadKS ......................................................... 52
CTD ................................................................. 53
CTDC ................................................................. 54
CTDD ................................................................. 55
CTDT ................................................................. 56
DDE ................................................................. 57
DiNUC2Binary_DNA ....................................................... 58
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DiNUC2Binary_RNA</td>
<td>59</td>
</tr>
<tr>
<td>DiNUCindex_DNA</td>
<td>60</td>
</tr>
<tr>
<td>DiNUCindex_RNA</td>
<td>62</td>
</tr>
<tr>
<td>DisorderB</td>
<td>63</td>
</tr>
<tr>
<td>DisorderC</td>
<td>64</td>
</tr>
<tr>
<td>DisorderS</td>
<td>65</td>
</tr>
<tr>
<td>DistancePair</td>
<td>66</td>
</tr>
<tr>
<td>DPCP_DNA</td>
<td>67</td>
</tr>
<tr>
<td>DPCP_RNA</td>
<td>68</td>
</tr>
<tr>
<td>EAAComposition</td>
<td>69</td>
</tr>
<tr>
<td>EffectiveNumberCodon</td>
<td>71</td>
</tr>
<tr>
<td>EGAAComposition</td>
<td>72</td>
</tr>
<tr>
<td>EIIP</td>
<td>73</td>
</tr>
<tr>
<td>ENUComposition_DNA</td>
<td>75</td>
</tr>
<tr>
<td>ENUComposition_RNA</td>
<td>76</td>
</tr>
<tr>
<td>ExpectedValKmerNUC_DNA</td>
<td>77</td>
</tr>
<tr>
<td>ExpectedValKmerNUC_RNA</td>
<td>78</td>
</tr>
<tr>
<td>ExpectedValueAA</td>
<td>80</td>
</tr>
<tr>
<td>ExpectedValueGAA</td>
<td>80</td>
</tr>
<tr>
<td>ExpectedValueGKmerAA</td>
<td>81</td>
</tr>
<tr>
<td>ExpectedValueKmerAA</td>
<td>83</td>
</tr>
<tr>
<td>fa.read</td>
<td>84</td>
</tr>
<tr>
<td>fickettScore</td>
<td>85</td>
</tr>
<tr>
<td>GAAKpartComposition</td>
<td>85</td>
</tr>
<tr>
<td>GrpDDE</td>
<td>87</td>
</tr>
<tr>
<td>G_Ccontent_DNA</td>
<td>88</td>
</tr>
<tr>
<td>G_Ccontent_RNA</td>
<td>89</td>
</tr>
<tr>
<td>kAAComposition</td>
<td>90</td>
</tr>
<tr>
<td>kGAAComposition</td>
<td>91</td>
</tr>
<tr>
<td>KNNPeptide</td>
<td>92</td>
</tr>
<tr>
<td>KNNProtein</td>
<td>93</td>
</tr>
<tr>
<td>KNN_DNA</td>
<td>95</td>
</tr>
<tr>
<td>KNN_RNA</td>
<td>96</td>
</tr>
<tr>
<td>kNUComposition_DNA</td>
<td>97</td>
</tr>
<tr>
<td>kNUComposition_RNA</td>
<td>98</td>
</tr>
<tr>
<td>LocalPoSpKAACF</td>
<td>99</td>
</tr>
<tr>
<td>LocalPoSpKNUCF_DNA</td>
<td>101</td>
</tr>
<tr>
<td>LocalPoSpKNUCF_RNA</td>
<td>102</td>
</tr>
<tr>
<td>maxORF</td>
<td>103</td>
</tr>
<tr>
<td>maxORFlength_DNA</td>
<td>104</td>
</tr>
<tr>
<td>maxORFlength_RNA</td>
<td>105</td>
</tr>
<tr>
<td>maxORF_RNA</td>
<td>106</td>
</tr>
<tr>
<td>Mismatch_DNA</td>
<td>107</td>
</tr>
<tr>
<td>Mismatch_RNA</td>
<td>108</td>
</tr>
<tr>
<td>MMI_DNA</td>
<td>109</td>
</tr>
<tr>
<td>MMI_RNA</td>
<td>110</td>
</tr>
<tr>
<td>nameKmer</td>
<td>111</td>
</tr>
<tr>
<td>NCP_DNA</td>
<td>111</td>
</tr>
</tbody>
</table>
R topics documented:

NCP_RNA ................................................................. 113
needleman ................................................................. 115
nonStandardSeq ......................................................... 115
NUC2Binary_DNA ......................................................... 116
NUC2Binary_RNA ........................................................ 118
NUCKpartComposition_DNA ............................................ 119
NUCKpartComposition_RNA ............................................. 120
OPF_10bit ................................................................. 121
OPF_7bit_T1 ............................................................... 122
OPF_7bit_T2 ............................................................... 124
OPF_7bit_T3 ............................................................... 125
PCPseDNC ................................................................. 126
PS2_DNA ................................................................. 128
PS2_RNA ................................................................. 129
PS3_DNA ................................................................. 131
PS3_RNA ................................................................. 132
PS4_DNA ................................................................. 134
PS4_RNA ................................................................. 135
PSEAAC ................................................................. 137
PseEIIP ................................................................. 138
PSEkNUCdi_DNA ......................................................... 139
PSEkNUCdi_RNA ......................................................... 140
PSEkNUCTri_DNA ......................................................... 142
PseKRAAC_T1 ............................................................ 143
PseKRAAC_T10 ........................................................... 145
PseKRAAC_T11 ........................................................... 146
PseKRAAC_T12 ........................................................... 148
PseKRAAC_T13 ........................................................... 149
PseKRAAC_T14 ........................................................... 151
PseKRAAC_T15 ........................................................... 152
PseKRAAC_T16 ........................................................... 154
PseKRAAC_T2 ............................................................ 156
PseKRAAC_T3A ........................................................... 157
PseKRAAC_T3B ........................................................... 159
PseKRAAC_T4 ............................................................ 160
PseKRAAC_T5 ............................................................ 162
PseKRAAC_T6A ........................................................... 163
PseKRAAC_T6B ........................................................... 164
PseKRAAC_T7 ............................................................ 166
PseKRAAC_T8 ............................................................ 167
PseKRAAC_T9 ............................................................ 169
PSSM ................................................................. 170
PSTNPds ................................................................. 171
PSTNPss_DNA ........................................................... 173
PSTNPss_RNA ........................................................... 174
QSortOrder .............................................................. 175
readASAdir .............................................................. 176
readDisDir .............................................................. 177
**AA2Binary**  

*Description*

This function transforms an amino acid to a binary format. The type of the binary format is determined by the `binaryType` parameter. For details about each format, please refer to the description of the `binaryType` parameter.

*Usage*

```r
AA2Binary(
  seqs,
  binaryType = "numBin",
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```
Arguments

`seqs` is a FASTA file with amino acid sequences. Each sequence starts with a ‘>’ character. Also, `seqs` could be a string vector. Each element of the vector is a peptide/protein sequence.

`binaryType` It can take any of the following values: ('strBin','logicBin','numBin'). 'strBin'(String binary): each amino acid is represented by a string containing 20 characters(0-1). For example, A = ALANIN = "1000000...0" 'logicBin'(logical value): Each amino acid is represented by a vector containing 20 logical entries. For example, A = ALANIN = c(T,F,F,F,F,F,...F) 'numBin' (numeric bin): Each amino acid is represented by a numeric (i.e., integer) vector containing 20 numerals. For example, A = ALANIN = c(1,0,0,0,0,0,0,...,0)

`label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

`outFormat` (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

`outputFileDist` shows the path and name of the 'txt' output file.

Value

The output is different depending on the `outFormat` parameter ('mat' or 'txt'). If `outFormat` is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if `binaryType` is 'strBin', the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences)*20. If `outFormat` is 'txt', all binary values will be written to a the output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in `outFormat` for sequences with different lengths. Warning: If `outFormat` is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

```r
ptmSeqsADR<-system.file("extdata/",package="ftrCOOL")
ptmSeqsVect<-as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
mat<-AA2Binary(seqs = ptmSeqsVect, binaryType="numBin",outFormat="mat")
```
**AAindex**

*Amino Acid Index (AAindex)*

**Description**

This function converts the amino acids of a sequence to a list of physicochemical properties in the aaIndex file. For each amino acid, the function uses a numeric vector which shows the aaIndex of the amino acid.

**Usage**

```r
AAindex(
  seqs,
  selectedAAidx = 1:554,
  standardized = TRUE,
  threshold = 1,
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

**Arguments**

- **seqs** is a FASTA file with amino acid sequences. Each sequence starts with a ‘>’ character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- **selectedAAidx** AAindex function works based on physicochemical properties. Users select the properties by their ids or indexes in aaIndex2 file.
- **standardized** is a logical parameter. If it is set to TRUE, amino acid indices will be in the standard format. The default value is TRUE.
- **threshold** is a number between (0, 1]. In selectedAAidx, indices with a correlation higher than the threshold will be deleted. The default value is 1.
- **label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
- **outFormat** (output format) can take two values: ‘mat’ (matrix) and ‘txt’. The default value is ‘mat’.
- **outputFileDist** shows the path and name of the ‘txt’ output file.

**Details**

In this function each amino acid is converted to a numeric vector. Elements of the vector represent a physicochemical property for the amino acid. In the aaIndex database, there are 554 amino acid indices. Users can choose the desired aaindex by specifying aaindexes through their ids or indexes in the aaIndex file, via selectedAAidx parameter.
AAKpartComposition

Value

The output depends on the outFormat parameter which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same length such that the number of columns is (sequence length)*(number of selected amino acid indexes) and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

dir = tempdir()
ptmSeqsADR<-system.file("extdata/",package="ftrCOOL")
ptmSeqsVect<-as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
mat<-AAindex(seqs = ptmSeqsVect, selectedAAidx=1:5,outFormat="mat")
ad<-paste0(dir,"/aaidx.txt")
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
AAindex(seqs = filePrs, selectedAAidx=1:5,standardized=TRUE,threshold=1,outFormat="txt" ,outputFileDist=ad)
unlink("dir", recursive = TRUE)

Description

In this function, each sequence is divided into k equal partitions. The length of each part is equal to ceiling(l(length of the sequence)/k). The last part can have a different length containing the residual amino acids. The amino acid composition is calculated for each part.

Usage

AAKpartComposition(seqs, k = 3, normalized = TRUE, label = c())
AAutoCor

Arguments

- seqs: is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- k: is an integer value. Each sequence should be divided to k partition(s).
- normalized: is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.
- label: is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

A feature matrix with k*20 number of columns. The number of rows is equal to the number of sequences.

Note

Warning: The length of all sequences should be greater than k.

Examples

```r
filePrs <- system.file("extdata/proteins.fasta", package="ftrCOOL")
mat <- AAKpartComposition(seqs = filePrs, k = 5, normalized = FALSE)
```

### AAutoCor

**Amino Acid Autocorrelation-Autocovariance (AAutoCor)**

**Description**

It creates the feature matrix for each function in autocorelation (i.e., Moran, Greay, NormalizeM-Borto) or autocovariance (i.e., AC, CC, ACC). The user can select any combination of the functions too. In this case, the final matrix will contain features of each selected function.

**Usage**

```r
AAutoCor(
  seqs,
  selectedAAidx = list(c("CIDH920105", "BHAR880101", "CHAM820101", "CHAM820102”, 
                         "CHOC760101", "BIGC670101", "CHAM810101", "DAYM780201")),
  maxlag = 3,
  threshold = 1,
  type = c("Moran", "Geary", "NormalizeMBorto", "AC", "CC", "ACC"),
  label = c()
)
```
### Arguments

**seqs**
- is a FASTA file with amino acid sequences. Each sequence starts with a `>` character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

**selectedAAidx**
- Function takes as input the physicochemical properties. Users select the properties by their ids or indices in the aaIndex2 file. This parameter could be a vector or a list of amino acid indices. The default values of the vector are the 'CIDH920105','BHAR880101','CHAM820101','CHAM820102','CHOC760101','BIGC670101','CHAM810101','DAYM780201' ids in the aaIndex2 file.

**maxlag**
- This parameter shows the maximum gap between two amino acids. The gaps change from 1 to maxlag (the maximum lag).

**threshold**
- is a number between (0, 1]. In selectedAAidx, indices with a correlation higher than the threshold will be deleted. The default value is 1.

**type**
- could be 'Moran', 'Greary', 'NormalizeMBorto', 'AC', 'CC', or 'ACC'. Also, it could be any combination of them.

**label**
- is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

### Details

For CC and AAC autocovariance functions, which consider the covariance of the two physicochemical properties, we have provided users with the ability to categorize their selected properties in a list. The binary combination of each group will be taken into account. Note: If all the features are in a group or selectedAAidx parameter is a vector, the binary combination will be calculated for all the physicochemical properties.

### Value

This function returns a feature matrix. The number of columns in the matrix changes depending on the chosen autocorrelation or autocovariance types and nlag parameter. The output is a matrix. The number of rows shows the number of sequences.

### Examples

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat1<-AAutoCor(seqs=filePrs,maxlag=20,threshold=0.9,
               type=c("Moran","Geary","NormalizeMBorto","AC"))

mat2<-AAutoCor(seqs=filePrs,maxlag=20,threshold=0.9,selectedAAidx=list(c('CIDH920105','BHAR880101','CHAM820101','CHAM820102'),c('CHOC760101','BIGC670101'),c('CHAM810101','DAYM780201')),type=c("AC","CC","ACC"))
```
AESNN3  Learn from alignments (AESNN3)

Description

This function replace each amino acid of the sequence with a three-dimensional vector. Values are taken from the three hidden units of the neural network trained on structure alignments. The AESNN3 function can be applied to encode peptides of equal length.

Usage

AESNN3(seqs, label = c(), outFormat = "mat", outputFileDist = "")

Arguments

seqs is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

outFormat (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

outputFileDist shows the path and name of the 'txt' output file.

Value

The output depends on the outFormat parameter which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same length such that the number of columns is (sequence length)*(5) and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat parameter for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes.

References

alphabetCheck

Examples

```r
ptmSeqsADR <- system.file("extdata/", package="ftrCOOL")
ptmSeqsVect <- as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
mat <- AESSN3(seqs = ptmSeqsVect, outFormat = "mat")
```

Description

This function checks the alphabets in a sequence. If one of the following conditions hold, the sequence will be deleted: 1. A peptide sequence containing non-standard amino acids, 2. A DNA sequence with an alphabet other than A, C, G, or T, 3. An RNA sequence having an alphabet other than A, C, G, or U.

Usage

```r
alphabetCheck(sequences, alphabet = "aa", label = c())
```

Arguments

- `sequences` is a string vector. Each element is a peptide, protein, DNA, or RNA sequences.
- `alphabet` This parameter shows the alphabet of sequences. If it is set to 'aa', it indicates the alphabet of amino acids. When it is 'dna', it shows the nucleotide alphabet and in case it equals 'rna', it represents ribonucleotide alphabet.
- `label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

'alphabetCheck' returns a list with two elements. The first element is a vector which contains valid sequences. The second element is a vector which contains the labels of the sequences (if any exists).

Note

This function receives a sequence vector and the label of sequences (if any). It deletes sequences (and their labels) containing non-standard alphabets.

Examples

```r
seq <- alphabetCheck(sequences = c("AGDFLIAACNMLKIVYT", "ADXVGAJK"), alphabet = "aa")
```
Accumulated Nucleotide Frequency (ANF_DNA)

Description
This function replaces nucleotides with a four-length vector. The first three elements represent the nucleotides and the forth holds the frequency of the nucleotide from the beginning of the sequence until the position of the nucleotide in the sequence. 'A' will be replaced with c(1, 1, 1, freq), 'C' with c(0, 1, 0, freq), 'G' with c(1, 0, 0, freq), and 'T' with c(0, 0, 1, freq).

Usage
ANF_DNA(seqs, outFormat = "mat", outputFileDist = "", label = c())

Arguments
- seqs: is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
- outFormat: (output format) can take two values: 'mat' (matrix) and 'txt'. The default value is 'mat'.
- outputFileDist: shows the path and name of the 'txt' output file.
- label: is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value
The output depends on the outFormat parameter which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same length such that the number of columns is (sequence length)*(4) and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.

Note
This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

References
Examples

```r
LNCSeqsADR <- system.file("extdata/", package="ftrCOOL")
LNC50Nuc <- as.vector(read.csv(paste0(LNCSeqsADR,"/LNC50Nuc.csv"))[ , 2])
mat <- ANF_DNA(seqs = LNC50Nuc, outFormat = "mat")
```

Description

This function replaces ribonucleotides with a four-length vector. The first three elements represent
the ribonucleotides and the forth holds the frequency of the ribonucleotide from the beginning of
the sequence until the position of the ribonucleotide in the sequence. 'A' will be replaced with c(1,
1, 1, freq), 'C' with c(0, 1, 0, freq),'G' with c(1, 0, 0, freq), and 'U' with c(0, 0, 1, freq).

Usage

```r
ANF_RNA(seqs, outFormat = "mat", outputFileDist = ",", label = c())
```

Arguments

- **seqs**: is a FASTA file containing ribonucleotide sequences. The sequences start with
  '>'. Also, seqs could be a string vector. Each element of the vector is a ribonu-
  cleotide sequence.
- **outFormat** (output format) can take two values: 'mat'(matrix) and 'txt'. The default value
  is 'mat'.
- **outputFileDist** shows the path and name of the 'txt' output file.
- **label** is an optional parameter. It is a vector whose length is equivalent to the number
  of sequences. It shows the class of each entry (i.e., sequence).

Value

The output depends on the outFormat parameter which can be either 'mat' or 'txt'. If outFormat is
'mat', the function returns a feature matrix for sequences with the same length such that the number
of columns is (sequence length)*4 and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in out-
Format for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences
with different lengths, it returns an error. Also, when output format is 'txt', label information is not
shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if
sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.
References

Examples

```r
fileLNC <- system.file("extdata/Carica_papaya101RNA.txt", package="ftrCOOL")
mat <- ANF_RNA(seqs = fileLNC, outFormat="mat")
```

Description
This function calculates the amphiphilic pseudo amino acid composition (Series) for each sequence.

Usage

```
APAAC(
  seqs,
  aaIDX = c("ARGP820101", "HOPT810101"),
  lambda = 30,
  w = 0.05,
  l = 1,
  threshold = 1,
  label = c()
)
```

Arguments

- **seqs**
  - is a FASTA file with amino acid sequences. Each sequence starts with a `>` character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

- **aaIDX**
  - is a vector of IDs or indexes of the user-selected physicochemical properties in the aaIndex2 database. The default values of the vector are the hydrophobicity ids and hydrophilicity ids in the amino acid index file.

- **lambda**
  - is a tuning parameter. Its value indicates the maximum number of spaces between amino acid pairs. The number changes from 1 to lambda.

- **w**
  - (weight) is a tuning parameter. It changes from 0 to 1. The default value is 0.05.

- **l**
  - This parameter keeps the value of l in lmer composition. The lmers form the first 20^l elements of the APAAC descriptor.

- **threshold**
  - is a number between (0, 1]. In aaIDX, indices with a correlation higher than the threshold will be deleted. The default value is 1.

- **label**
  - is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
Details

This function computes the pseudo amino acid composition for each physicochemical property. We have provided users with the ability to choose among different properties (i.e., not confined to hydrophobicity or hydrophilicity).

Value

A feature matrix such that the number of columns is \(20^l+\text{(number of chosen aalIndex}\times\text{lambda})\) and the number of rows equals the number of sequences.

Examples

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat<-APAAC(seqs=filePrs,l=2,lambda=3,threshold=1)
```

---

### Description

This function calculates the amphiphilic pseudo k nucleotide composition(Di) (Series) for each sequence.

Usage

```r
APAkNUCdi_DNA(
  seqs,
  selectedIdx = c("Rise", "Roll", "Shift", "Slide", "Tilt", "Twist"),
  lambda = 3,
  w = 0.05,
  l = 2,
  ORF = FALSE,
  reverseORF = TRUE,
  threshold = 1,
  label = c()
)
```

Arguments

- `seqs` is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
selectedIdx is a vector of Ids or indices of the desired physicochemical properties of dinucleotides. Users can choose the desired indices by their ids or their names in the DI_DNA index file. The default value of this parameter is a vector with ("Rise", "Roll", "Shift", "Slide", "Tilt", "Twist") ids.

lambda is a tuning parameter. This integer value shows the maximum limit of spaces between dinucleotide pairs. The Number of spaces changes from 1 to lambda.

w (weight) is a tuning parameter. It changes in the range of 0 to 1. The default value is 0.05.

l This parameter keeps the value of l in lmer composition. The lmers form the first $4^l$ elements of the APkNCdi descriptor.

ORF (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

reverseORF is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

threshold is a number between (0 to 1]. In selectedIdx, indices with a correlation higher than the threshold will be deleted. The default value is 1.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details

This function computes the pseudo nucleotide composition for each physicochemical property of dinucleotides. We have provided users with the ability to choose among the 148 properties in the di-nucleotide index database.

Value

It is a feature matrix. The number of columns is $4^l + (number of the chosen indices \times \lambda$) and the number of rows is equal to the number of sequences.

Examples

```r
fileLNC<-'system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
mat<-APkNUCdi_DNA(seqs=fileLNC,ORF=TRUE,threshold=1)
```

---

**Description**

This function calculates the amphiphilic pseudo k ribonucleotide composition(Di) (Series) for each sequence.
Usage

APkNUCdi_RNA(
  seqs,
  selectedIdx = c("Rise (RNA)", "Roll (RNA)", "Shift (RNA)", "Slide (RNA)",
  "Tilt (RNA)", "Twist (RNA)"),
  lambda = 3,
  w = 0.05,
  l = 2,
  ORF = FALSE,
  reverseORF = TRUE,
  threshold = 1,
  label = c()
)

Arguments

seqs is a FASTA file containing ribonucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.

selectedIdx is a vector of Ids or indices of the desired physicochemical properties of di-ribonucleotides. Users can choose the desired indices by their ids or their names in the DI_RNA index file. The default value of this parameter is a vector with ("Rise (RNA)", "Roll (RNA)", "Shift (RNA)", "Slide (RNA)", "Tilt (RNA)", "Twist (RNA)") ids.

lambda is a tuning parameter. This integer value shows the maximum limit of spaces between di-ribonucleotide pairs. The Number of spaces changes from 1 to lambda.

w (weight) is a tuning parameter. It changes in the range of 0 to 1. The default value is 0.05.

l This parameter keeps the value of l in lmer composition. The lmers form the first $4^l$ elements of the APkNCdi descriptor.

ORF (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

reverseORF is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

threshold is a number between (0 to 1]. In selectedIdx, indices with a correlation higher than the threshold will be deleted. The default value is 1.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details

This function computes the pseudo ribonucleotide composition for each physicochemical property of di-ribonucleotides. We have provided users with the ability to choose among the 22 properties in the di-ribonucleotide index database.
Value

It is a feature matrix. The number of columns is \(4^\lambda\) + (number of the chosen indices \(\times \lambda\)) and the number of rows is equal to the number of sequences.

Examples

```r
fileLNC <- system.file("extdata/Carica_papaya101RNA.txt", package="ftrCOOL")
mat <- APkNUCdi_RNA(seqs=fileLNC, ORF=TRUE, threshold=0.8)
```

---

**APkNUCTri_DNA**  
Amphiphilic Pseudo-k Nucleotide Composition-Tri(series) (**APkNUC-Tri_DNA**)  

**Description**

This function calculates the amphiphilic pseudo k nucleotide composition(Tri) (Series) for each sequence.

**Usage**

```r
APkNUCTri_DNA(
  seqs,
  selectedIdx = c("Dnase I", "Bendability (DNAse)"),
  lambda = 3,
  w = 0.05,
  l = 3,
  ORF = FALSE,
  reverseORF = TRUE,
  threshold = 1,
  label = c()
)
```

**Arguments**

- **seqs**: is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
- **selectedIdx**: is a vector of IDs or indices of the desired physicochemical properties of trinucleotides. Users can choose the desired indices by their IDs or their names in the TRI_DNA index file. The default value of the parameter is a vector with ("Dnase I", "Bendability (DNAse)") IDs.
- **lambda**: is a tuning parameter. This integer value shows the maximum limit of spaces between trinucleotide pairs. The Number of spaces changes from 1 to lambda.
- **w**: (weight) is a tuning parameter. It changes in the range of 0 to 1. The default value is 0.05.
ASA

1  This parameter keeps the value of 1 in lmer composition. The lmers form the first $4^l$ of the APkNCTri descriptor.

ORF  (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

reverseORF  is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

threshold  is a number between (0, 1]. In selectedIdx, indices with a correlation higher than the threshold will be deleted. The default value is 1.

label  is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details

This function computes the pseudo nucleotide composition for each physicochemical property of trinucleotides. We have provided users with the ability to choose among the 12 properties in the tri-nucleotide index database.

Value

It is a feature matrix. The number of columns is $4^l + \text{(number of the chosen indices}*\lambda)$ and the number of rows is equal to the number of sequences.

Examples

```r
fileLNC<system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
mat<-APkNUCTri_DNA(seqs=fileLNC,l=3,threshold=1)
```

ASA  

**Accessible Solvent Accessibility (ASA)**

Description

ASA represents an amino acid by a numeric value. This function extracts the ASA from the output of SPINE-X software which predicts ASA for each amino acid in a peptide or protein sequence. The output of SPINE-X is a tab-delimited file. ASAs are in the 11th column of the file.

Usage

```r
ASA(dirPath, outFormat = "mat", outputFileDist = "")
```
Arguments

dirPath Path of the directory which contains all output files of SPINE-X. Each file belongs to a sequence.

outFormat It can take two values: 'mat' (which stands for matrix) and 'txt'. The default value is 'mat'.

outputFileDist It shows the path and name of the 'txt' output file.

Value

The output depends on the outFormat which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same lengths such that the number of columns is equal to the length of the sequences and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

dir = tempdir()
ad<-paste0(dir,"/asa.txt")

PredASAdir<-system.file("testForder",package="ftrCOOL")
PredASAdir<-paste0(PredASAdir,"/ASAdir/"
ASA(PredASAdir,outFormat="txt",outputFileDist=ad)
unlink("dir", recursive = TRUE)

ASDC Adaptive skip dipeptide composition (ASDC)

Description

This descriptor sufficiently considers the correlation information present not only between adjacent residues but also between intervening residues. This function calculates frequency of pair amino acids omitting gaps between them. Then this function normalizes each value through dividing each frequency by summition(frequencies).

Usage

ASDC(seqs, label = c())
Arguments

seqs is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

The function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 400 (all possible amino acid pairs).

References


Examples

```r
filePrs <- system.file("extdata/proteins.fasta", package="ftrCOOL")
mat <- ASDC(seqs = filePrs)
```

Description

This descriptor sufficiently considers the correlation information present not only between adjacent nucleotides but also between intervening nucleotides. This function calculates frequency of pair nucleotides omitting gaps between them. Then this function normalizes each value through dividing each frequency by summition(frequencies).

Usage

```
ASDC_DNA(seqs, ORF = FALSE, reverseORF = TRUE, label = c())
```

Arguments

seqs is a FASTA file containing nucleotide sequences. The sequences start with '>'. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

ORF (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).
reverseORF is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

The function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 16 (All possible nucleotide pairs).

References


Examples

```r
fileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
fileLNC<-fa.read(file=fileLNC,alphabet="dna")[1:5]
mat1<-ASDC_DNA(seqs=fileLNC,ORF=TRUE,reverseORF=FALSE)
```

ASDC_RNA

Adaptive skip di-ribonucleotide composition (ASDC_RNA)

Description

This descriptor sufficiently considers the correlation information present not only between adjacent ribo ribonucleotides but also between intervening nucleotides This function calculates frequency of pair ribonucleotides omitting gaps between them. Then this function normalizes each value through dividing each frequency by summition(frequencies).

Usage

```r
ASDC_RNA(seqs, ORF = FALSE, reverseORF = TRUE, label = c())
```

Arguments

- `seqs` is a FASTA file containing ribonucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.
- `ORF` (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).
reverseORF is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

The function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 16 (All posible ribonucleotide pairs).

References


Examples

```r
ptmSeqsADR<-system.file("extdata/",package="ftrCOOL")
fileLNC<-fa.read(file=paste0(ptmSeqsADR,"/testSeq2RNA51.txt"),alphabet="rna")
mat1<-ASDC_RNA(seqs=fileLNC)
```

<table>
<thead>
<tr>
<th>AutoCorDiNUC_DNA</th>
<th>Di Nucleotide Autocorrelation-Autocovariance (AutoCorDiNUC_DNA)</th>
</tr>
</thead>
</table>

Description

It creates the feature matrix for each function in autocorelation (i.e., Moran, Greay, NormalizeMBorto) or autocovariance (i.e., AC, CC, ACC). The user can select any combination of the functions too. In this case, the final matrix will contain features of each selected function.

Usage

```r
AutoCorDiNUC_DNA(
  seqs, 
  selectedIdx = c("Rise", "Roll", "Shift", "Slide", "Tilt", "Twist"),
  maxlag = 3,
  threshold = 1,
  type = c("Moran", "Geary", "NormalizeMBorto", "AC", "CC", "ACC"),
  label = c()
)
```
AutoCorDiNUC_RNA

Arguments

seqs  is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

selectedIdx  function takes as input the physicochemical properties. Users select the properties by their ids or indices in the DI_DNA file. This parameter could be a vector or a list of dinucleotide indices. The default value of this parameter is a vector with ("Rise", "Roll", "Shift", "Slide", "Tilt", "Twist") ids.

maxlag  This parameter shows the maximum gap between two dinucleotide pairs. The gaps change from 1 to maxlag (the maximum lag).

threshold  is a number between (0 to 1]. In selectedIdx, indices with a correlation higher than the threshold will be deleted. The default value is 1.

type  could be 'Moran', 'Greay', 'NormalizeMBorto', 'AC', 'CC', or 'ACC'. Also, it could be any combination of them.

label  is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details

For CC and AAC autocovariance functions, which consider the covariance of the two physicochemical properties, we have provided users with the ability to categorize their selected properties in a list. The binary combination of each group will be taken into account. Note: If all the features are in a group or selectedAAIdx parameter is a vector, the binary combination will be calculated for all the physicochemical properties.

Value

This function returns a feature matrix. The number of columns in the matrix changes depending on the chosen autocorrelation or autocovariance types and nlag parameter. The output is a matrix. The number of rows shows the number of sequences.

Examples

```r
fileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
mat2<-AutoCorDiNUC_DNA(seqs=fileLNC,selectedIdx=list(10,c(1,3),6:13,c(2:7)),
,maxlag=15,type="CC")
```

| AutoCorDiNUC_RNA | Di riboNucleotide Autocorrelation-Autocovariance (AutoCorDiNUC_RNA) |
Description

It creates the feature matrix for each function in autocorelation (i.e., Moran, Greay, NormalizeMBorto) or autocovariance (i.e., AC, CC, ACC). The user can select any combination of the functions too. In this case, the final matrix will contain features of each selected function.

Usage

AutoCorDiNUC_RNA(
  seqs,
  selectedIdx = c("Rise (RNA)", "Roll (RNA)", "Shift (RNA)", "Slide (RNA)",
  "Tilt (RNA)", "Twist (RNA)"),
  maxlag = 3,
  threshold = 1,
  type = c("Moran", "Geary", "NormalizeMBorto", "AC", "CC", "ACC"),
  label = c()
)

Arguments

seqs is a FASTA file containing ribonucleic acid(RNA) sequences. The sequences start with '>'. Also, seqs could be a string vector. Each element of the vector is a RNA sequence.

selectedIdx function takes as input the physicochemical properties. Users select the properties by their ids or indices in the DI_RNA file. This parameter could be a vector or a list of di-ribonucleotide indices. The default value of this parameter is a vector with ("Rise (RNA)", "Roll (RNA)", "Shift (RNA)", "Slide (RNA)", "Tilt (RNA)", "Twist (RNA)") ids.

maxlag This parameter shows the maximum gap between two di-ribonucleotide pairs. The gaps change from 1 to maxlag (the maximum lag).

threshold is a number between (0 to 1]. In selectedIdx, indices with a correlation higher than the threshold will be deleted. The default value is 1.

type could be 'Moran', 'Greay', 'NormalizeMBorto', 'AC', 'CC', or 'ACC'. Also, it could be any combination of them.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details

For CC and AAC autocovariance functions, which consider the covariance of the two physicochemical properties, we have provided users with the ability to categorize their selected properties in a list. The binary combination of each group will be taken into account. Note: If all the features are in a group or selectedAAidx parameter is a vector, the binary combination will be calculated for all the physicochemical properties.
Value

This function returns a feature matrix. The number of columns in the matrix changes depending on the chosen autocorrelation or autocovariance types and nlag parameter. The output is a matrix. The number of rows shows the number of sequences.

Examples

```r
fileLNC <- system.file("extdata/Carica_papaya101RNA.txt", package="ftrCOOL")
fileLNC <- fa.read(fileLNC, alphabet="rna")
fileLNC <- fileLNC[1:20]
mat1 <- AutoCorDiNUC_RNA(seqs=fileLNC,maxlag=20,type=c("Moran"))
```

<table>
<thead>
<tr>
<th>AutoCorTriNUC_DNA</th>
<th>Tri Nucleotide</th>
<th>Autocorrelation-Autocovariance</th>
<th>(AutoCor-TriNUC_DNA)</th>
</tr>
</thead>
</table>

Description

It creates the feature matrix for each function in autocorelation (i.e., Moran, Geary, NormalizeMBorto) or autocovariance (i.e., AC, CC, ACC). The user can select any combination of the functions too. In this case, the final matrix will contain features of each selected function.

Usage

```r
AutoCorTriNUC_DNA(
  seqs,
  selectedNucIdx = c("Dnase I", "Bendability (DNAse)"),
  maxlag = 3,
  threshold = 1,
  type = c("Moran", "Geary", "NormalizeMBorto", "AC", "CC", "ACC"),
  label = c()
)
```

Arguments

- **seqs**: is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
- **selectedNucIdx**: function takes as input the physicochemical properties. Users select the properties by their ids or indices in the TRI_DNA file. This parameter could be a vector or a list of trinucleotide indices. The default value of this parameter is a vector with ("Dnase I", "Bendability (DNase)") ids.
- **maxlag**: This parameter shows the maximum gap between two tri-nucleotide pairs. The gaps change from 1 to maxlag (the maximum lag).
threshold is a number between (0 to 1]. In selectedNucIdx, indices with a correlation higher than the threshold will be deleted. The default value is 1.

type could be 'Moran', 'Geary', 'NormalizeMBorto', 'AC', 'CC', or 'ACC'. Also, it could be any combination of them.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details

For CC and AAC autocovriance functions, which consider the covariance of the two physicochemical properties, we have provided users with the ability to categorize their selected properties in a list. The binary combination of each group will be taken into account. Note: If all the features are in a group or selectedAAidx parameter is a vector, the binary combination will be calculated for all the physicochemical properties.

Value

This function returns a feature matrix. The number of columns in the matrix changes depending on the chosen autocorrelation or autocovariance types and nlag parameter. The output is a matrix. The number of rows shows the number of sequences.

Examples

```r
fileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
mat1<-AutoCorTriNUC_DNA(seqs=fileLNC,selectedNucIdx=c(1:7),maxlag=20,type=c("Moran","Geary"))
mat2<-AutoCorTriNUC_DNA(seqs=fileLNC,selectedNucIdx=list(c(1,3),6:10,c(2:7)),
maxlag=15,type=c("AC","CC"))
```

---

**binary_3bit_T1**  
**Binary - 3bit - Type 1 (binary_3bit_T1)**

Description

This group of functions(binary_3bit_T1-T7) categorizes amino acids in 3 groups based on the type. Then represent group of amino acids by a three dimensional vector. The type of the binary format is determined by the binaryType parameter. For details about each format, please refer to the description of the binaryType parameter.

Usage

```r
binary_3bit_T1(
  seqs,
  binaryType = "numBin",
  label = c(),
  outFormat = "mat",
)```
Arguments

**seqs** is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

**binaryType** It can take any of the following values: ('strBin','logicBin','numBin'). 'strBin'(String binary): each amino acid is represented by a string containing 20 characters(0-1). For example, A = ALANIN = "1000000...0" 'logicBin'(logical value): Each amino acid is represented by a vector containing 20 logical entries. For example, A = ALANIN = c(T,F,F,F,F,...F) 'numBin’ (numeric bin): Each amino acid is represented by a numeric (i.e., integer) vector containing 20 numerals. For example, A = ALANIN = c(1,0,0,0,0,0,0,...,0)

**label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

**outFormat** (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

**outputFileDist** shows the path and name of the 'txt' output file.

Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences)*3. If outFormat is 'txt', all binary values will be written to a the output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

```r
ptmSeqsADR<-system.file("extdata/",package="ftrCOOL")
ptmSeqsVect<-as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[2])
mat<-binary_3bit_T1(seqs = ptmSeqsVect, binaryType="numBin",outFormat="mat")
```
Description

This group of functions (binary_3bit_T1-T7) categorizes amino acids in 3 groups based on the type. Then represent group of amino acids by a three dimensional vector. The type of the binary format is determined by the binaryType parameter. For details about each format, please refer to the description of the binaryType parameter.

Usage

```r
binary_3bit_T2(
  seqs,  
  binaryType = "numBin",  
  label = c(),  
  outFormat = "mat",  
  outputFileDist = ""
)
```

Arguments

- **seqs** is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

- **binaryType** It can take any of the following values: ('strBin', 'logicBin', 'numBin'). 'strBin'(String binary): each amino acid is represented by a string containing 20 characters(0-1). For example, A = ALANIN = "1000000...0" 'logicBin'(logical value): Each amino acid is represented by a vector containing 20 logical entries. For example, A = ALANIN = c(T,F,F,F,F,F,...F) 'numBin' (numeric bin): Each amino acid is represented by a numeric (i.e., integer) vector containing 20 numerals. For example, A = ALANIN = c(1,0,0,0,0,0,0,...,0)

- **label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

- **outFormat** (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

- **outputFileDist** shows the path and name of the 'txt' output file.

Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences)*3. If outFormat is 'txt', all binary values will be written to a the output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.
**Note**

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

**Examples**

```r
ptmSeqsADR<-system.file("extdata/",package="ftrCOOL")
ptmSeqsVect<-as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
mat<-binary_3bit_T2(seqs = ptmSeqsVect, binaryType="numBin",outFormat="mat")
```

**Description**

This group of functions(binary_3bit_T1-T7) categorizes amino acids in 3 groups based on the type. Then represent group of amino acids by a three dimensional vector. The type of the binary format is determined by the binaryType parameter. For details about each format, please refer to the description of the binaryType parameter.

**Usage**

```r
binary_3bit_T3(
  seqs,
  binaryType = "numBin",
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

**Arguments**

- `seqs` is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- `binaryType` It can take any of the following values: ('strBin','logicBin','numBin'). 'strBin'(String binary): each amino acid is represented by a string containing 20 characters(0-1). For example, A = ALANIN = "1000000...0" 'logicBin'(logical value): Each amino acid is represented by a vector containing 20 logical entries. For example, A = ALANIN = c(T,F,F,F,F,F,...F) 'numBin' (numeric bin): Each amino acid is represented by a numeric (i.e., integer) vector containing 20 numerals. For example, A = ALANIN = c(1,0,0,0,0,0,...,0)
label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

outFormat (output format) can take two values: 'mat' (matrix) and 'txt'. The default value is 'mat'.

outputFileDist shows the path and name of the 'txt' output file.

Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences)*3. If outFormat is 'txt', all binary values will be written to a file. The output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

```r
ptmSeqsADR<-system.file("extdata/",package="ftrCOOL")
ptmSeqsVect<-as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
mat<-binary_3bit_T3(seqs = ptmSeqsVect, binaryType="numBin",outFormat="mat")
```

---

**binary_3bit_T4**

**Binary - 3bit - Type4 (binary_3bit_T4)**

**Description**

This group of functions (binary_3bit_T1-T7) categorizes amino acids in 3 groups based on the type. Then represent group of amino acids by a three dimensional vector. The type of the binary format is determined by the binaryType parameter. For details about each format, please refer to the description of the binaryType parameter.

**Usage**

```r
binary_3bit_T4(
  seqs,
  binaryType = "numBin",
  label = c(),
```
binary_3bit_T4

```r
outFormat = "mat",
outputFileDist = ""
)
```

**Arguments**

*seqs* is a FASTA file with amino acid sequences. Each sequence starts with a ‘>’ character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

*binaryType* It can take any of the following values: (`strBin`, `logicBin`, `numBin`). `strBin` (String binary): each amino acid is represented by a string containing 20 characters (0-1). For example, A = ALANIN = "1000000...0" `logicBin` (logical value): Each amino acid is represented by a vector containing 20 logical entries. For example, A = ALANIN = c(T,F,F,F,F,...F) `numBin` (numeric bin): Each amino acid is represented by a numeric (i.e., integer) vector containing 20 numerals. For example, A = ALANIN = c(1,0,0,0,0,0,0,...,0)

*label* is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

*outFormat* (output format) can take two values: `mat` (matrix) and `txt`. The default value is `mat`.

*outputFileDist* shows the path and name of the 'txt' output file.

**Value**

The output is different depending on the outFormat parameter (`mat` or `txt`). If outFormat is set to `mat`, it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is `strBin`, the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences) * 3. If outFormat is `txt`, all binary values will be written to a the output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

**Note**

This function is provided for sequences with the same lengths. Users can use `txt` option in outFormat for sequences with different lengths. Warning: If outFormat is set to `mat` for sequences with different lengths, it returns an error. Also, when output format is `txt`, label information is not shown in the text file. It is noteworthy that `txt` format is not usable for machine learning purposes if sequences have different sizes. Otherwise `txt` format is also usable for machine learning purposes.

**Examples**

```r
ptmSeqsADR<-system.file("extdata/",package="ftrCOOL")
ptmSeqsVect<-as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
mat<-binary_3bit_T4(seqs = ptmSeqsVect, binaryType="numBin",outFormat="mat")
```
**Description**

This group of functions (binary_3bit_T1-T7) categorizes amino acids in 3 groups based on the type. Then represent group of amino acids by a three dimensional vector. The type of the binary format is determined by the binaryType parameter. For details about each format, please refer to the description of the binaryType parameter.

**Usage**

```r
binary_3bit_T5(
  seqs,
  binaryType = "numBin",
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

**Arguments**

- **seqs**
  is a FASTA file with amino acid sequences. Each sequence starts with a ‘>’ character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

- **binaryType**
  It can take any of the following values: ('strBin','logicBin','numBin'), 'strBin'(String binary): each amino acid is represented by a string containing 20 characters(0-1). For example, A = ALANIN = "1000000...0" 'logicBin'(logical value): Each amino acid is represented by a vector containing 20 logical entries. For example, A = ALANIN = c(T,F,F,F,F,F,....F) 'numBin' (numeric bin): Each amino acid is represented by a numeric (i.e., integer) vector containing 20 numerals. For example, A = ALANIN = c(1,0,0,0,0,0,0,...,0)

- **label**
  is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

- **outFormat**
  (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

- **outputFileDist**
  shows the path and name of the 'txt' output file.

**Value**

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences)*3. If outFormat is 'txt', all binary values will be written to a the output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.
Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

```r
ptmSeqsADR <- system.file("extdata/", package="ftrCOOL")
ptmSeqsVect <- as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
mat <- binary_3bit_T6(seqs = ptmSeqsVect, binaryType="numBin", outFormat="mat")
```

Description

This group of functions(binary_3bit_T1-T7) categorizes amino acids in 3 groups based on the type. Then represent group of amino acids by a three dimentional vector. The type of the binary format is determined by the binaryType parameter. For details about each format, please refer to the description of the binaryType parameter.

Usage

```r
binary_3bit_T6(
  seqs,
  binaryType = "numBin",
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

Arguments

- **seqs** is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

- **binaryType** It can take any of the following values: ('strBin', 'logicBin', 'numBin'). 'strBin'(String binary): each amino acid is represented by a string containing 20 characters(0-1). For example, A = ALANIN = "1000000...0" 'logicBin'(logical value): Each amino acid is represented by a vector containing 20 logical entries. For example, A = ALANIN = c(T,F,F,F,F,F,...F) 'numBin' (numeric bin): Each amino acid is represented by a numeric (i.e., integer) vector containing 20 numerals. For example, A = ALANIN = c(1,0,0,0,0,0,...,0)
is an optional parameter. It is a vector whose length is equivalent to the number
of sequences. It shows the class of each entry (i.e., sequence).
outFormat (output format) can take two values: 'mat' (matrix) and 'txt'. The default value
is 'mat'.
outputFileDist shows the path and name of the 'txt' output file.

Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to
'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal
to the number of sequences and if binaryType is 'strBin', the number of columns is the length of
the sequences. Otherwise, it is equal to (length of the sequences)*3. If outFormat is 'txt', all binary
values will be written to a the output is written to a tab-delimited file. Each line in the file shows
the binary format of a sequence.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in out-
Format for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences
with different lengths, it returns an error. Also, when output format is 'txt', label information is not
shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if
sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

```
ptmSeqsADR<-system.file("extdata/",package="ftrCOOL")
ptmSeqsVect<-as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
mat<-binary_3bit_T6(seqs = ptmSeqsVect, binaryType="numBin",outFormat="mat")
```

```
binary_3bit_T7 Binary - 3bit - Type7 (binary_3bit_T7)
```

Description

This group of functions (binary_3bit_T1-T7) categorizes amino acids in 3 groups based on the type.
Then represent group of amino acids by a three dimensional vector. The type of the binary for-
mat is determined by the binaryType parameter. For details about each format, please refer to the
description of the binaryType parameter.

Usage

```
binary_3bit_T7(
  seqs,
  binaryType = "numBin",
  label = c(),
```

outFormat = "mat",
outputFileDist = ""
)

### Arguments

- **seqs**: A FASTA file with amino acid sequences. Each sequence starts with a `>` character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- **binaryType**: It can take any of the following values: ('strBin', 'logicBin', 'numBin'). 'strBin'(String binary): each amino acid is represented by a string containing 20 characters (0-1). For example, A = ALANIN = "1000000...0" 'logicBin'(logical value): Each amino acid is represented by a vector containing 20 logical entries. For example, A = ALANIN = c(T,F,F,F,F,...F) 'numBin' (numeric bin): Each amino acid is represented by a numeric (i.e., integer) vector containing 20 numerals. For example, A = ALANIN = c(1,0,0,0,0,0,...,0)
- **label**: It is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
- **outFormat** (output format) can take two values: ‘mat’ (matrix) and 'txt'. The default value is 'mat'.
- **outputFileDist**: It shows the path and name of the 'txt' output file.

### Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences)*3. If outFormat is 'txt', all binary values will be written to a the output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

### Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

### Examples

ptmSeqsADR<- system.file("extdata/",package="ftrCOOL")
ptmSeqsVect<-as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
mat<-binary_3bit_T7(seqs = ptmSeqsVect, binaryType="numBin", outFormat="mat")
binary_5bit_T1

Binary - 5bit - Type1 (binary_5bit_T1)

**Description**

This function categorizes amino acids in 5 groups. Then represent group of amino acids by a 5 dimensional vector i.e., e1, e2, e3, e4, e5. e1=G, A, V, L, M, I, e2=F, Y, W, e3=K, R, H, e4=D, E, e5=S, T, C, P, N, Q. e1 is encoded by 10000 e2 is encoded by 01000 and ... and e5 is encoded by 00001.

**Usage**

```r
binary_5bit_T1(
  seqs,
  binaryType = "numBin",
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

**Arguments**

- `seqs` is a FASTA file with amino acid sequences. Each sequence starts with a ' >' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

- `binaryType` It can take any of the following values: ('strBin', 'logicBin', 'numBin'). 'strBin'(String binary): each amino acid is represented by a string containing 20 characters (0-1). For example, A = ALANIN = "1000000...0"'logicBin'(logical value): Each amino acid is represented by a vector containing 20 logical entries. For example, A = ALANIN = c(T,F,F,F,F,F,...F)'numBin'(numeric bin): Each amino acid is represented by a numeric (i.e., integer) vector containing 20 numerals. For example, A = ALANIN = c(1,0,0,0,0,0,0,...0)

- `label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

- `outFormat` (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

- `outputFileDist` shows the path and name of the 'txt' output file.

**Details**

The type of the binary format is determined by the binaryType parameter. For details about each format, please refer to the description of the binaryType parameter.
Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences)*5. If outFormat is 'txt', all binary values will be written to a the output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

```r
ptmSeqsADR<-system.file("extdata/",package="ftrCOOL")
ptmSeqsVect<-as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
mat<-binary_5bit_T1(seqs = ptmSeqsVect, binaryType="numBin",outFormat="mat")
```

Description

The idea behind this function is: We have 20 amino acids and we can show them with at least 5 bits. A is encoded by (00011), C (00101), D (00110), E (00111), F (01001), G (01010), H (01011), I (01100), K (01101), L (01110), M (10001), N (10010), P (10011), Q (10100), R (10101), S (10110), T (11000), V (11001), W (11010), Y (11100). This function transforms an amino acid to a binary format. The type of the binary format is determined by the binaryType parameter. For details about each format, please refer to the description of the binaryType parameter.

Usage

```r
binary_5bit_T2(
  seqs,
  binaryType = "numBin",
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```
Arguments

seqs is a FASTA file with amino acid sequences. Each sequence starts with a ‘>’ character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

binaryType It can take any of the following values: ('strBin', 'logicBin', 'numBin'). 'strBin'(String binary): each amino acid is represented by a string containing 20 characters(0-1). For example, A = ALANIN = "1000000...0" 'logicBin'(logical value): Each amino acid is represented by a vector containing 20 logical entries. For example, A = ALANIN = c(T,F,F,F,F,F,...F) 'numBin'(numeric bin): Each amino acid is represented by a numeric (i.e., integer) vector containing 20 numerals. For example, A = ALANIN = c(1,0,0,0,0,0,0,...,0)

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

outFormat (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

outputFileDist shows the path and name of the 'txt' output file.

Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences)*5. If outFormat is 'txt', all binary values will be written to a the output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

ptmSeqsADR<system.file("extdata/",package="ftrCOOL")
ptmSeqsVect<-as.vector(read.csv(paste0(ptmSeqsADR, "/ptmVect101AA.csv"))[,2])
mat<-binary_5bit_T2(seqs = ptmSeqsVect, binaryType="numBin",outFormat="mat")
Description

This function categorizes amino acids in 6 groups. Then represent group of amino acids by a 6 dimentional vector i.e. e1, e2, e3, e4, e5, e6. e1=H, R, K, e2=D, E, N, D, e3=C, e4=S, T, P, A, G, e5=M, I, L, V, e6=F, Y, W. e1 is ecoded by 100000 e2 is encoded by 010000 and ... and e6 is encoded by 000001.

Usage

```
binary_6bit(
  seqs,
  binaryType = "numBin",
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

Arguments

- `seqs` is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

- `binaryType` It can take any of the following values: ('strBin', 'logicBin', 'numBin'). 'strBin'(String binary): each amino acid is represented by a string containing 20 characters(0-1). For example, A = ALANIN = "1000000...0" 'logicBin'(logical value): Each amino acid is represented by a vector containing 20 logical entries. For example, A = ALANIN = c(T,F,F,F,F,F,F,...F) 'numBin' (numeric bin): Each amino acid is represented by a numeric (i.e., integer) vector containing 20 numerals. For example, A = ALANIN = c(1,0,0,0,0,0,0,...,0)

- `label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

- `outFormat` (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

- `outputFileDist` shows the path and name of the 'txt' output file.

Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences)*6. If outFormat is 'txt', all binary values will be written to a the output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.
Note
This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

```r
ptmSeqsADR <- system.file("extdata/", package="ftrCOOL")
ptmSeqsVect <- as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
mat <- binary_6bit(seqs = ptmSeqsVect, binaryType = "numBin", outFormat = "mat")
```

Description
This function creates a 20-dimensional numeric vector for each amino acid of a sequence. Each entry of the vector contains the similarity score of the amino acid with other amino acids including itself. The score is extracted from the Blosum62 matrix.

Usage

```r
BLOSUM62(seqs, label = c(), outFormat = "mat", outputFileDist = "")
```

Arguments

- **seqs** is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- **label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
- **outFormat** (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.
- **outputFileDist** shows the path and name of the 'txt' output file.

Value
The output depends on the outFormat parameter which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same length such that the number of columns is (sequence length)*20 and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.
Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

```r
dir = tempdir()
ptmSeqsADR<-system.file("extdata/",package="ftrCOOL")
filePr<-system.file("extdata/protein.fasta",package="ftrCOOL")
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
ad<-paste0(dir,"/blosum62.txt")
vect<-BLOSUM62(seqs = filePr,outFormat="mat")
BLOSUM62(seqs = filePrs,outFormat="txt",outputFileDist=ad)
unlink("dir", recursive = TRUE)
```

CkSAApair  Composition of k-Spaced Amino Acids pairs (CkSAApair)

Description

This function calculates the composition of k-spaced amino acid pairs. In other words, it computes the frequency of all amino acid pairs with k spaces.

Usage

CkSAApair(seqs, rng = 3, upto = FALSE, normalized = TRUE, label = c())

Arguments

- **seqs** is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- **rng** This parameter can be a number or a vector. Each element of the vector shows the number of spaces between amino acid pairs. For each k in the rng vector, a new vector (whose size is 400) is created which contains the frequency of pairs with k gaps.
- **upto** It is a logical parameter. The default value is FALSE. If rng is a number and upto is set to TRUE, rng is converted to a vector with values from [0 to rng].
CkSGAApair is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

The function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 400*(length of rng vector).

Note

'upto' is enabled only when rng is a number and not a vector.

Examples

```r
filePrs <- system.file("extdata/proteins.fasta", package="ftrCOOL")
mat1 <- CkSAApair(seqs=filePrs, rng=2, upto=TRUE, normalized=TRUE)
mat2 <- CkSAApair(seqs=filePrs, rng=c(1,3,5))
```

CkSGAApair Composition of k-Spaced Grouped Amino Acids pairs (CkSGAApair)

Description

In this function, amino acids are first grouped into a category which is defined by the user. Later, the composition of the k-spaced grouped amino acids is computed. Please note that this function differs from CkSAApair which works on individual amino acids.

Usage

```r
ckSGAApair(
    seqs,
    rng = 3,
    upto = FALSE,
    normalized = TRUE,
    Grp = "locFus",
    label = c()
)
```
Arguments

**seqs**
is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

**rng**
This parameter can be a number or a vector. Each element of the vector shows the number of spaces between amino acid pairs. For each k in the rng vector, a new vector (whose size is (number of categorizes)^2) is created which contains the frequency of pairs with k gaps.

**upto**
It is a logical parameter. The default value is FALSE. If rng is a number and upto is set to TRUE, rng is converted to a vector with values from [1 to rng].

**normalized**
is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.

**Grp**
is a list of vectors containing amino acids. Each vector represents a category. Users can define a customized amino acid grouping, provided that the sum of all amino acids is 20 and there is no repeated amino acid in the groups. Also, users can choose 'cTriad'(conjointTriad), 'locFus', or 'aromatic'. Each option provides specific information about the type of an amino acid grouping.

**label**
is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details

Column names in the feature matrix follow G(?ss?). For example, G(1ss2) means Group1**Group2, where '*' is a wild character.

Value

This function returns a feature matrix. Row length is equal to the number of sequences and the number of columns is ((number of categorizes)^2)*(length of rng vector).

Note

'upto' is enabled only when rng is a number and not a vector.

Examples

filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat1<-CkSGAApair(seqs=filePrs,rng=2,upto=TRUE,Grp="aromatic")

mat2<-CkSGAApair(seqs=filePrs,rng=c(1,3,5),upto=FALSE,Grp=
CkSNUCpair_DNA  Composition of k-Spaced Nucleotides Pairs (CkSNUCpair_DNA)

Description

This function calculates the composition of k-spaced nucleotide pairs. In other words, it computes the frequency of all nucleotide pairs with k spaces.

Usage

CkSNUCpair_DNA(
  seqs,
  rng = 3,
  upto = FALSE,
  ORF = FALSE,
  reverseORF = TRUE,
  normalized = TRUE,
  label = c()
)

Arguments

seqs  is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

rng  This parameter can be a number or a vector. Each element of the vector shows the number of spaces between nucleotide pairs. For each k in the rng vector, a new vector (whose size is 16) is created which contains the frequency of pairs with k gaps.

upto  It is a logical parameter. The default value is FALSE. If rng is a number and upto is set to TRUE, rng is converted to a vector with values from [0 to rng].

ORF  (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

reverseORF  is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

normalized  is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.

label  is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

The function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 16*(length of rng vector).
Note

'upto' is enabled only when rng is a number and not a vector.

Examples

```r
fileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
mat1<-CkSNUCpair_DNA(seqs=fileLNC,rng=2,upto=TRUE,ORF=TRUE,reverseORF=FALSE)
mat2<-CkSNUCpair_DNA(seqs=fileLNC,rng=c(1,3,5))
```

CkSNUCpair_RNA  Composition of k-Spaced riboNucleotides Pairs (CkSNUCpair_RNA)

Description

This function calculates the composition of k-spaced ribonucleotide pairs. In other words, it computes the frequency of all ribonucleotide pairs with k spaces.

Usage

```r
CkSNUCpair_RNA(
  seqs,
  rng = 3,
  upto = FALSE,
  ORF = FALSE,
  reverseORF = TRUE,
  normalized = TRUE,
  label = c()
)
```

Arguments

- `seqs` is a FASTA file containing ribonucleotide sequences. The sequences start with '>'. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.
- `rng` This parameter can be a number or a vector. Each element of the vector shows the number of spaces between ribonucleotide pairs. For each k in the rng vector, a new vector (whose size is 16) is created which contains the frequency of pairs with k gaps.
- `upto` It is a logical parameter. The default value is FALSE. If rng is a number and upto is set to TRUE, rng is converted to a vector with values from [0 to rng].
- `ORF` (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).
reverseORF is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

normalized is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

The function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 16*(length of rng vector).

Note

'upto' is enabled only when rng is a number and not a vector.

Examples

```r
fileLNC<-system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
mat1<-CkSNUCpair_RNA(seqs=fileLNC,rng=2,upto=TRUE,ORF=TRUE,reverseORF=FALSE)
mat2<-CkSNUCpair_RNA(seqs=fileLNC,rng=c(1,3,5))
```

codonAdaptionIndex  Codon Adaption Index (codonAdaptionIndex)

Description

This function calculates the codon adaption index for each sequence.

Usage

codonAdaptionIndex(seqs, ORF = FALSE, reverseORF = TRUE, label = c())

Arguments

seqs is a FASTA file containing nucleotide sequences. The sequences start with '>'. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

ORF (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

reverseORF is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).
**CodonFraction**

**Description**

This function calculates the codon fraction for each sequence.

**Usage**

CodonFraction(seqs, ORF = FALSE, reverseORF = TRUE, label = c())

**Arguments**

- **seqs** is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

- **ORF** (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

- **reverseORF** is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

- **label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

**Value**

A feature matrix such that the number of columns is $4^3$ and the number of rows is equal to the number of sequences.
CodonUsage_DNA

Examples

```r
fileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
mat<-CodonFraction(seqs=fileLNC,ORF=TRUE,reverseORF=FALSE)
```

CodonUsage_DNA

Codon Usage in DNA (CodonUsage_DNA)

Description

This function calculates the codon usage for each sequence.

Usage

```r
CodonUsage_DNA(seqs, ORF = FALSE, reverseORF = TRUE, label = c())
```

Arguments

- `seqs`: is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, `seqs` could be a string vector. Each element of the vector is a nucleotide sequence.
- `ORF` (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).
- `reverseORF` is a logical parameter. It is enabled only if `ORF` is true. If `reverseORF` is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).
- `label`: is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

A feature matrix such that the number of columns is $4^3$ and the number of rows is equal to the number of sequences.

Examples

```r
fileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
mat<-CodonUsage_DNA(fileLNC,ORF=TRUE,reverseORF=FALSE)
```
CodonUsage_RNA

Description
This function calculates the codon usage for each sequence.

Usage
CodonUsage_RNA(seqs, ORF = FALSE, reverseORF = TRUE, label = c())

Arguments
seqs
is a FASTA file containing ribonucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.

ORF
(Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

reverseORF
is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

label
is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value
A feature matrix such that the number of columns is $4^3$ and the number of rows is equal to the number of sequences.

Examples
```r
fileLNC <- system.file("extdata/Carica_papaya101RNA.txt", package="ftrCOOL")
mat <- CodonUsage_RNA(seqs = fileLNC, ORF = TRUE, reverseORF = FALSE)
```

conjointTriad

Conjoint Triad (conjointTriad)

Description
This function calculates the grouped tripeptide composition with the conjoint triad grouping type.

Usage
conjointTriad(seqs, normalized = TRUE, label = c())
Arguments

- `seqs` is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, `seqs` could be a string vector. Each element of the vector is a peptide/protein sequence.
- `normalized` is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.
- `label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

This function returns a feature matrix. The number of rows equals to the number of sequences and the number of columns is $7^3$.

Examples

```r
filePrs <- system.file("extdata/proteins.fasta", package = "ftrCOOL")
mat1 <- conjointTriad(seqs = filePrs)
```

---

**Description**

This function calculates the grouped tripeptide composition with conjoint triad grouping type. For each $k$, it creates a $7^3$ feature vector. $K$ is the space between the first and the second amino acids and the second and the third amino acids of the tripeptide.

**Usage**

```r
conjointTriadKS(seqs, rng = 3, upto = FALSE, normalized = FALSE, label = c())
```

**Arguments**

- `seqs` is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, `seqs` could be a string vector. Each element of the vector is a peptide/protein sequence.
- `rng` This parameter can be a number or a vector. Each element of the vector shows the number of spaces between the first and the second amino acids and the second and the third amino acids of the tripeptide. For each $k$ in the `rng` vector, a new vector (whose size is $7^3$) is created which contains the frequency of tri-amino acid with $k$ gaps.
upto 

It is a logical parameter. The default value is FALSE. If rng is a number and upto is set to TRUE, rng is converted to a vector with values from 0 to rng.

normalized 

is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.

label 

is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details

A tripeptide with k spaces looks like AA1(ss..s)AA2(ss..s)AA3. AA stands for amino acids and s means space.

Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is (7*3)*(length rng vector).

Note

'upto' is enabled only when rng is a number and not a vector.

Examples

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat1<-conjointTriadKS(filePrs,rng=2,upto=TRUE,normalized=TRUE)
mat2<-conjointTriadKS(filePrs,rng=c(1,3,5))
```

---

**CTD** Composition_Transition_Distribution (CTD)

Description

This function calculates the composition, transition, and distribution for each sequence.

Usage

```r
CTD(seqs, normalized = FALSE, label = c())
```
Arguments

seqs is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

normalized is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

Output is a combination of three different matrices: Composition, Transition, and Distribution. You can obtain any of the three matrices by executing the corresponding function, i.e., CTDC, CTDT, and CTDD.

References


Examples

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
CTDtotal<-CTD(seqs=filePrs,normalized=FALSE)
```

CTDC

CTD Composition (CTDC)

Description

This function computes the composition part of CTD. Thirteen properties are defined in this function. Each property categorizes the amino acids of the sequences into three groups. The grouped amino acid composition is calculated for each property. For more information, please check the references.

Usage

```r
CTDC(seqs, normalized = FALSE, label = c())
```
Arguments

**seqs**

is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

**normalized**

is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.

**label**

is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 3*7, where three is the number of groups and thirteen is the number of properties.

References


Examples

```r
filePrs <- system.file("extdata/proteins.fasta", package="ftrCOOL")
CTD_C <- CTDC(seqs=filePrs, normalized=FALSE, label=c())
```

---

**CTDD**

*CTD Distribution (CTDD)*

Description

This function computes the distribution part of CTD. It calculates fifteen values for each property. For more information, please check the references.

Usage

```r
CTDD(seqs, label = c())
```

Arguments

**seqs**

is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

**label**

is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
Value
This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 15*7.

References

Examples

```
filePrs <- system.file("extdata/proteins.fasta", package="ftrCOOL")
CTD_D <- CTD(seqs = filePrs)
```

<table>
<thead>
<tr>
<th>CTDT</th>
<th>CTD Transition (CTDT)</th>
</tr>
</thead>
</table>

Description
This function computes the transition part of CTD. Thirteen properties are defined in this function. Each property categorizes the amino acids of a sequence into three groups. For each property, the grouped amino acid transition (i.e., transitions 1-2, 1-3, and 2-3) is calculated. For more information, please check the references.

Usage

```
CTDT(seqs, normalized = FALSE, label = c())
```

Arguments

- **seqs**
  - is a FASTA file with amino acid sequences. Each sequence starts with a ‘>’ character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

- **normalized**
  - is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.

- **label**
  - is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value
This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 3*7, where three is the number of transition types (i.e., 1-2, 1-3, and 2-3) and thirteen is the number of properties.
References


Examples

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
CTD_T<-CTDT(seqs=filePrs,normalized=FALSE)
```

---

### DDE

Dipeptide Deviation from Expected Mean value (DDE)

**Description**

This function computes the dipeptide deviation from the expected mean value.

**Usage**

```r
DDE(seqs, label = c())
```

**Arguments**

- `seqs` is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- `label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

**Value**

A feature matrix with $20^2=400$ number of columns. The number of rows is equal to the number of sequences.

**Examples**

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat<-DDE(seqs=filePrs)
```
DiNUC2Binary_DNA

Dinucleotide To Binary DNA (DiNUC2Binary_DNA)

Description

This function transforms a dinucleotide to a binary number with four bits which is enough to represent all the possible types of dinucleotides. The type of the binary format is determined by the binaryType parameter. For details about each format, please refer to the description of the binaryType parameter.

Usage

DiNUC2Binary_DNA(
  seqs,
  binaryType = "numBin",
  outFormat = "mat",
  outputFileDist = "",
  label = c()
)

Arguments

seqs is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

binaryType It can take any of the following values: ('strBin', 'logicBin', 'numBin'). 'strBin' (String binary): each dinucleotide is represented by a string containing 4 characters (0-1). For example, AA = "0000" AC = "0001" ... TT = "1111". 'logicBin' (logical value): Each dinucleotide is represented by a vector containing 4 logical entries. For example, AA = c(F,F,F,F) AC = c(F,F,F,T) ... TT = c(T,T,T,T). 'numBin' (numeric bin): Each dinucleotide is represented by a numeric (i.e., integer) vector containing 4 numeric entries. For example, AA = c(0,0,0,0) AC = c(0,0,0,1) ... TT = c(1,1,1,1)

outFormat (output format) can take two values: 'mat' (matrix) and 'txt'. The default value is 'mat'.

outputFileDist shows the path and name of the 'txt' output file.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is the (length of the sequences-1). Otherwise, it is equal to (length of the sequences-1)*4. If outFormat is 'txt', all binary values will be written to a tab-delimited file. Each line in the file shows the binary format of a sequence.
Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

```
LNCSeqsADR<-system.file("extdata/",package="ftrCOOL")
LNC50Nuc<-as.vector(read.csv(paste0(LNCSeqsADR,"/LNC50Nuc.csv"))[,2])
mat<-DiNUC2Binary_DNA(seqs = LNC50Nuc, binaryType="numBin",outFormat="mat")
```

DiNUC2Binary_RNA  

Di riboNucleotide To Binary RNA (DiNUC2Binary_RNA)

Description

This function transforms a di-ribonucleotide to a binary number with four bits which is enough to represent all the possible types of di-ribonucleotides. The type of the binary format is determined by the binaryType parameter. For details about each format, please refer to the description of the binaryType parameter.

Usage

```
DiNUC2Binary_RNA(
  seqs,
  binaryType = "numBin",
  outFormat = "mat",
  outputFileDist = "",
  label = c()
)
```

Arguments

- **seqs**: is a FASTA file containing ribonucleotide sequences. The sequences start with '>'. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.
- **binaryType**: It can take any of the following values: ('strBin','logicBin','numBin'). 'strBin' (String binary): each di-ribonucleotide is represented by a string containing 4 characters(0-1). For example, AA = "0000" AC="0001" ... TT="1111" 'logicBin' (logical value): Each di-ribonucleotide is represented by a vector containing 4 logical entries. For example, AA = c(F,F,F,F) AC=c(F,F,T) ... TT=c(T,T,T,T) 'numBin' (numeric bin): Each di-ribonucleotide is represented by a numeric (i.e., integer) vector containing 4 numeric entries. For example, AA = c(0,0,0,0) AC = c(0,0,0,1) ... TT = c(1,1,1,1)
DiNUCindex_DNA

(output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

outputFileDist shows the path and name of the 'txt' output file.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is (length of the sequences-1). Otherwise, it is equal to (length of the sequences-1)*4. If outFormat is 'txt', all binary values will be written to a 'txt' file. Each line in the file shows the binary format of a sequence.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

```
fileLNC<-system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
mat<-DiNUC2Binary_RNA(seqs = fileLNC, binaryType="numBin",outFormat="mat")
```

DiNUCindex_DNA

Di Nucleotide Index (DiNUCindex_DNA)

Description

This function replaces dinucleotides in a sequence with their physicochemical properties in the dinucleotide index file.

Usage

```
DiNUCindex_DNA(
  seqs, 
  selectedIdx = c("Rise", "Roll", "Shift", "Slide", "Tilt", "Twist"), 
  threshold = 1, 
  label = c(), 
  outFormat = "mat", 
  outputFileDist = ""
)
```
DiNUCindex_DNA

Arguments

seqs is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

selectedIdx DiNUCindex_DNA function works based on physicochemical properties. Users, select the properties by their ids or indexes in DI_DNA index file. The default value of this parameter is a vector with ("Rise", "Roll", "Shift", "Slide", "Tilt", "Twist") entries.

threshold is a number between (0 , 1]. In selectedIdx, indices with a correlation higher than the threshold will be deleted. The default value is 1.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

outFormat (output format) can take two values: ‘mat’ (matrix) and 'txt'. The default value is ‘mat’.

outputFileDist shows the path and name of the 'txt' output file.

Details

There are 148 physicochemical indexes in the dinucleotide database.

Value

The output depends on the outFormat parameter which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same length such that the number of columns is (sequence length-1)*(number of selected di-nucleotide indexes) and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

fileLNC<-system.file("extdata/Athaliana1.fa",package="ftrCOOL")
vect<-DiNUCindex_DNA(seqs = fileLNC,outFormat="mat")
DiNUCindex_RNA

Di riboNucleotide Index (DiNUCindex_RNA)

Description

This function replaces di-ribonucleotides in a sequence with their physicochemical properties in the di-ribonucleotide index file.

Usage

```r
DiNUCindex_RNA(
  seqs,
  selectedIdx = c("Rise (RNA)", "Roll (RNA)", "Shift (RNA)", "Slide (RNA)",
    "Tilt (RNA)", "Twist (RNA)"),
  threshold = 1,
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

Arguments

- `seqs` is a FASTA file containing ribonucleotide sequences. The sequences start with '>'. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.
- `selectedIdx` DiNucIndex function works based on physicochemical properties. Users, select the properties by their ids or indexes in DI_RNA file. The default value of this parameter is a vector with ("Rise (RNA)", "Roll (RNA)", “Shift (RNA)”, “Slide (RNA)”, “Tilt (RNA)”, “Twist (RNA)”) entries.
- `threshold` is a number between (0, 1]. In selectedIdx, indices with a correlation higher than the threshold will be deleted. The default value is 1.
- `label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
- `outFormat` (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.
- `outputFileDist` shows the path and name of the 'txt' output file.

Details

There are 22 physicochemical indexes in the di-ribonucleotide database.

Value

The output depends on the outFormat parameter which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same length such that the number of columns is (sequence length-1)*(number of selected di-ribonucleotide indexes) and the number
of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.

Note
This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

```
fileLNC<-system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
vect<DiNUCindex_RNA(seqs = fileLNC,outFormat="mat")
```

<table>
<thead>
<tr>
<th>DisorderB</th>
<th>disorder Binary (DisorderB)</th>
</tr>
</thead>
</table>

Description
This function extracts the ordered and disordered amino acids in protein or peptide sequences. The input to the function is provided by VSL2 software. Also, the function converts order amino acids to ’10’ and disorder amino acids to ’01’.

Usage

```
DisorderB(  
  dirPath,  
  binaryType = "numBin",  
  outFormat = "mat",  
  outputFileDist = ""
)
```

Arguments

dirPath Path of the directory which contains all output files of VSL2. Each file belongs to a sequence.

binaryType It can take any of the following values: (’strBin’,’logicBin’,’numBin’), ’strBin’ (String binary): each amino acid is represented by a string containing 2 characters(0-1). order = "10" disorder="01". ’logicBin’ (logical value): Each amino acid is represented by a vector containing 2 logical entries. order = c(TRUE,FALSE) disorder=c(FALSE,TRUE). ’numBin’ (numeric bin): Each amino acid is represented by a numeric (i.e., integer) vector containing 2 numeric entries. order = c(1,0) disorder=c(0,1).

outFormat It can take two values: 'mat' (which stands for matrix) and 'txt'. The default value is 'mat'.

outputFileDist It shows the path and name of the 'txt' output file.
Value

The output is different depending on the outFormat parameter (‘mat’ or ‘txt’). If outFormat is set
to ‘mat’, it returns a feature matrix for sequences with the same lengths. The number of rows is
equal to the number of sequences and if binaryType is ‘strBin’, the number of columns is the length
of the sequences. Otherwise, it is equal to (length of the sequences)*2. If outFormat is ‘txt’, all
binary values will be written to a tab-delimited file. Each line in the file shows the binary format of
a sequence.

Note

This function is provided for sequences with the same lengths. Users can use ‘txt’ option in out-
Format for sequences with different lengths. Warning: If outFormat is set to ‘mat’ for sequences
with different lengths, it returns an error. Also, when output format is ‘txt’, label information is not
shown in the text file. It is noteworthy that ‘txt’ format is not usable for machine learning purposes if
sequences have different sizes. Otherwise ‘txt’ format is also usable for machine learning purposes.

Examples

```r
dir = tempdir()
PredDisdir<-system.file("testForder",package="ftrCOOL")
PredDisdir<-paste0(PredDisdir,"/Disdir/"
ad1<-paste0(dir,"/disorderB.txt")

DisorderB(PredDisdir,binaryType="numBin",outFormat="txt",outputFileDist=ad1)
unlink("dir", recursive = TRUE)
```

<table>
<thead>
<tr>
<th>DisorderC</th>
<th>disorder Content (DisorderC)</th>
</tr>
</thead>
</table>

Description

This function extracts ordered and disordered amino acids in protein or peptide sequences. The
input to the function is provided by VSL2 software. Also, the function returns number of order and
disorder amino acids in the sequence.

Usage

`DisorderC(dirPath)`

Arguments

- `dirPath` : Path of the directory which contains all output files of VSL2. Each file belongs
to a sequence.
DisorderS

Value

The output is a feature matrix with 2 columns. The number of rows is equal to the number of sequences.

Examples

```r
dir = tempdir()
PredDisdir<-system.file("testForder",package="ftrCOOL")
PredDisdir<-paste0(PredDisdir,"/Disdir/")

mat<-DisorderC(PredDisdir)
```

DisorderS  disorder Simple (DisorderS)

Description

This function extracts ordered and disordered amino acids in protein or peptide sequences. The input to the function is provided by VSL2 software. The function represent order amino acids by 'O' and disorder amino acids by 'D'.

Usage

```
DisorderS(dirPath, outFormat = "mat", outputFileDist = "")
```

Arguments

- **dirPath**: Path of the directory which contains all output files of VSL2. Each file belongs to a sequence.
- **outFormat**: It can take two values: 'mat' (which stands for matrix) and 'txt'. The default value is 'mat'.
- **outputFileDist**: It shows the path and name of the 'txt' output file.

Value

The output depends on the outFormat which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same lengths such that the number of columns is equal to the length of the sequences and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.
Examples

```r
dir = tempdir()

PredDisdir<-system.file("testFolder",package="ftrCOOL")
PredDisdir<-paste0(PredDisdir,"/Disdir/")
ad1<-paste0(dir,"/disorderS.txt")

DisorderS(PredDisdir, outFormat="txt",outputFileDist=ad1)

unlink("dir", recursive = TRUE)
```

DistancePair

**PseAAC of distance-pairs and reduced alphabet (DistancePair)**

Description

In this function, first amino acids are grouped into a category which is one of ‘cp13’, ‘cp14’, ‘cp19’, ‘cp20’. Users choose one of these terms to categorize amino acids. Then DistancePair function computes frequencies of all grouped residues and also all grouped-paired residues with [0,rng] distance. ‘rng’ is a parameter which already was set by the user.

Usage

```r
DistancePair(seqs, rng = 3, normalized = TRUE, Grp = "cp14", label = c())
```

Arguments

- `seqs` is a FASTA file with amino acid sequences. Each sequence starts with a ‘>’ character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- `rng` This parameter is a number. It shows maximum number of spaces between amino acid pairs. For each k in the rng vector, a new vector (whose size is (number of categorizes)^2) is created which contains the frequency of pairs with k gaps.
- `normalized` is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.
- `Grp` for this parameter users can choose between these items: ‘cp13’, ‘cp14’, ‘cp19’, or ‘cp20’.
- `label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

This function returns a feature matrix. Row length is equal to the number of sequences and the number of columns is (number of categorizes)+((number of categorizes)^2)*(rng+1).
**Examples**

```r
gpr <- system.file("extdata/proteins.fasta", package="ftrCOOL")
mat1 <- DistancePair(seqs=gpr, rng=2, Grp="cp14")
```

---

**DPCP_DNA**

*Dinucleotide physicochemical properties (DPCP_DNA)*

**Description**

This function replaces dinucleotides in a sequence with their physicochemical properties which is multiplied by normalized frequency of that di-nucleotide.

**Usage**

```r
DPCP_DNA(
  seqs,
  selectedIdx = c("Rise", "Roll", "Shift", "Slide", "Tilt", "Twist"),
  threshold = 1,
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

**Arguments**

- **seqs**
  - is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

- **selectedIdx**
  - DPCP_DNA function works based on physicochemical properties. Users, select the properties by their ids or indexes in DI_DNA index file. The default value of this parameter is a vector with ("Rise", "Roll", "Shift", "Slide", "Tilt", "Twist") entries.

- **threshold**
  - is a number between (0 , 1]. In selectedIdx, indices with a correlation higher than the threshold will be deleted. The default value is 1.

- **label**
  - is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

- **outFormat**
  - (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

- **outputFileDist**
  - shows the path and name of the 'txt' output file.

**Details**

There are 148 physicochemical indexes in the dinucleotide database.
The output depends on the `outFormat` parameter which can be either 'mat' or 'txt'. If `outFormat` is 'mat', the function returns a feature matrix for sequences with the same length such that the number of columns is \((\text{sequence length}-1)\times(\text{number of selected di-nucleotide indexes})\) and the number of rows is equal to the number of sequences. If the `outFormat` is 'txt', the output is written to a tab-delimited file.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in `outFormat` for sequences with different lengths. Warning: If `outFormat` is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

```r
fileLNC<-system.file("extdata/Athaliana1.fa",package="ftrCOOL")
vect<-DPCP_DNA(seqs = fileLNC,outFormat="mat")
```

---

**DPCP_RNA**

_Di-ribonucleotide physicochemical properties (DPCP_RNA)_

Description

This function replaces di-ribonucleotides in a sequence with their physicochemical properties which is multiplied by normalized frequency of that di-ribonucleotide.

Usage

```r
DPCP_RNA(
  seqs,
  selectedIdx = c("Rise (RNA)", "Roll (RNA)", "Shift (RNA)", "Slide (RNA)", "Tilt (RNA)", "Twist (RNA)"),
  threshold = 1,
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

Arguments

- `seqs` is a FASTA file containing ribonucleotide sequences. The sequences start with ‘>’. Also, `seqs` could be a string vector. Each element of the vector is a ribonucleotide sequence.
**EAAComposition**

**selectedIdx**  
DiNucIndex function works based on physicochemical properties. Users, select the properties by their ids or indexes in DI_RNA file. The default value of this parameter is a vector with ("Rise (RNA)", "Roll (RNA)", "Shift (RNA)", "Slide (RNA)", "Tilt (RNA)", "Twist (RNA)") entries.

**threshold**  
is a number between (0, 1]. In selectedAAidx, indices with a correlation higher than the threshold will be deleted. The default value is 1.

**label**  
is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

**outFormat**  
(output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

**outputFileDist**  
shows the path and name of the 'txt' output file.

**Details**

There are 22 physicochemical indexes in the di-ribonucleotide database.

**Value**

The output depends on the outFormat parameter which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same length such that the number of columns is (sequence length-1)*(number of selected di-ribonucleotide indexes) and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.

**Note**

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

**Examples**

```r
fileLNC<-system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
vect<-DPCP_RNA(seqs = fileLNC,outFormat="mat")
```

---

**EAAComposition**  
*Enhanced Amino Acid Composition (EAAComposition)*

**Description**

This function slides a window over the input sequence(s). Also, it computes the composition of amino acids that appears within the limits of the window.
Usage

EAAComposition(
    seqs,
    winSize = 50,
    overLap = TRUE,
    label = c(),
    outFormat = "mat",
    outputFileDist = ""
)

Arguments

seqs is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

winSize is a number which shows the size of the window.

overLap This parameter shows how the window moves over the sequence. If overlap is set to FALSE, the window slides over the sequence in such a way that every time the window moves, it covers a unique portion of the sequence. Otherwise, portions of the sequence which appear within the window limits have "winSize-1" amino acids in common.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

outFormat (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

outputFileDist shows the path and name of the 'txt' output file.

Details

Column names in the output matrix are Wi(aa), where aa shows an amino acid type ("A", "C", "D"..., "Y") and i indicates the number of times that the window has moved over the sequence(s).

Value

The output depends on the outFormat parameter which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same length such that the number of columns is (20 * number of partitions displayed by the window) and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

When overlap is FALSE, the last partition represented by the window may have a different length with other parts.
EffectiveNumberCodon

References


Examples

dir = tempdir()
ptmSeqsADR<-system.file("extdata/",package="ftrCOOL")
ptmSeqsVect<-as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
mat<-EAAComposition(seqs = ptmSeqsVect,winSize=50, overLap=FALSE,outFormat='mat')

ad<-paste0(dir,"/EaaCompos.txt")
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
EAAComposition(seqs = filePrs,winSize=50, overLap=FALSE,outFormat="txt",outputFileDist=ad)
unlink("dir", recursive = TRUE)

EffectiveNumberCodon

Effective Number of Codon (EffectiveNumberCodon)

Description

This function calculates the effective number of codon for each sequence.

Usage

EffectiveNumberCodon(seqs, ORF = FALSE, reverseORF = TRUE, label = c())

Arguments

seqs is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

ORF (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

reverseORF is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

The function returns a feature vector. The length of the vector is equal to the number of sequences. Each entry in the vector contains the effective number of codon.
EGAAComposition

Examples

```r
fileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
vect<-EffectiveNumberCodon(seqs=fileLNC,ORF=TRUE,reverseORF=FALSE)
```

---

EGAAComposition | Enhanced Grouped Amino Acid Composition (EGAAComposition)

Description

In this function, amino acids are first grouped into user-defined categories. Then, enhanced grouped amino acid composition is computed. For details about the enhanced feature, please refer to function EAAACComposition. Please note that this function differs from function EAAACComposition which works on individual amino acids.

Usage

```r
EGAAComposition(
  seqs,                        # is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
  winSize = 50,                # shows the size of sliding window. It is a numeric value.
  overLap = TRUE,              # This parameter shows how the window moves on the sequence. If the overlap is set to TRUE, the next window would have distance 1 with the previous window. Otherwise, the next window will start from the next amino acid after the previous window. There is no overlap between the next and previous windows.
  Grp = "locFus",             # is a list of vectors containing amino acids. Each vector represents a category. Users can define a customized amino acid grouping, provided that the sum of all amino acids is 20 and there is no repeated amino acid in the groups. Also, users can choose 'cTriad'(conjointTriad), 'locFus', or 'aromatic'. Each option provides specific information about the type of an amino acid grouping.
  label = c(),                # is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
  outFormat = "mat",          #
  outputFileDist = ""         #
)
```

Arguments

- **seqs**: is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- **winSize**: shows the size of sliding window. It is a numeric value.
- **overLap**: This parameter shows how the window moves on the sequence. If the overlap is set to TRUE, the next window would have distance 1 with the previous window. Otherwise, the next window will start from the next amino acid after the previous window. There is no overlap between the next and previous windows.
- **Grp**: is a list of vectors containing amino acids. Each vector represents a category. Users can define a customized amino acid grouping, provided that the sum of all amino acids is 20 and there is no repeated amino acid in the groups. Also, users can choose 'cTriad'(conjointTriad), 'locFus', or 'aromatic'. Each option provides specific information about the type of an amino acid grouping.
- **label**: is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
EIIP

Description
This function replaces each nucleotide in the input sequence with its electron-ion interaction value. The resulting sequence is represented by a feature vector whose length is equal to the length of the sequence. Please check the references for more information.
Usage

EIIP(seqs, outFormat = "mat", outputFileDist = "", label = c())

Arguments

- seqs: a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
- outFormat: (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.
- outputFileDist: shows the path and name of the 'txt' output file.
- label: is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

The output depends on the outFormat parameter which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same length such that the number of columns is equal to the length of the sequences and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat parameter for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

References


Examples

LNCSeqsADR<- system.file("extdata/", package="ftrCOOL")
LNC50Nuc<-as.vector(read.csv(paste0(LNCSeqsADR,"/LNC50Nuc.csv"))[,2])
mat<-EIIP(seqs = LNC50Nuc, outFormat="mat")
**ENUComposition_DNA**

**Enhanced Nucleotide Composition (ENUComposition_DNA)**

**Description**

This function slides a window over the input sequence(s). Also, it computes the composition of nucleotides that appears within the limits of the window.

**Usage**

```r
ENUComposition_DNA(
  seqs,
  winSize = 50,
  overLap = TRUE,
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

**Arguments**

- `seqs` is a FASTA file containing nucleotide sequences. The sequences start with `'>'. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
- `winSize` is a number which shows the size of the window.
- `overLap` This parameter shows how the window moves on the sequence. If the overlap is set to TRUE, the next window would have distance 1 with the previous window. Otherwise, the next window will start from the next nucleotide after the previous window. There is no overlap between the next and previous windows.
- `label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
- `outFormat` (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.
- `outputFileDist` shows the path and name of the 'txt' output file.

**Value**

The output depends on the outFormat parameter which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same length such that the number of columns is (4 * number of partitions displayed by the window) and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.
Note
This function is provided for sequences with the same lengths. Users can use ‘txt’ option in out-
Format for sequences with different lengths. Warning: If outFormat is set to ‘mat’ for sequences
with different lengths, it returns an error. Also, when output format is ‘txt’, label information is not
shown in the text file. It is noteworthy that ‘txt’ format is not usable for machine learning purposes if
sequences have different sizes. Otherwise ‘txt’ format is also usable for machine learning purposes.

Examples

```r
dir = tempdir()
LNCSeqsADR<-system.file("extdata/",package="ftrCOOL")
LNC50Nuc<-as.vector(read.csv(paste0(LNCSeqsADR,"/LNC50Nuc.csv"))[,2])
mat<-ENUComposition_DNA(seqs = LNC50Nuc, winSize=20,outFormat="mat")
ad<-paste0(dir,"/ENUCcompos.txt")
fileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
ENUComposition_DNA(seqs = fileLNC,outFormat="txt",winSize=20
, outputFileDist=ad,overLap=FALSE)

unlink("dir", recursive = TRUE)
```

ENUComposition_RNA

Enhanced riboNucleotide Composition (ENUComposition_RNA)

Description
This function slides a window over the input sequence(s). Also, it computes the composition of
ribonucleotides that appears within the limits of the window.

Usage

```r
ENUComposition_RNA(
  seqs,
  winSize = 50,
  overLap = TRUE,
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

Arguments

- `seqs` is a FASTA file containing ribonucleotide sequences. The sequences start with
  ‘>’. Also, seqs could be a string vector. Each element of the vector is a ribo-
nucleotide sequence.
- `winSize` is a number which shows the size of the window.
overLap

This parameter shows how the window moves on the sequence. If the overlap is set to TRUE, the next window would have distance 1 with the previous window. Otherwise, the next window will start from the next ribonucleotide after the previous window. There is no overlap between the next and previous windows.

label

is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

outFormat

(output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

outputFileDist

shows the path and name of the 'txt' output file.

Value

The output depends on the outFormat parameter which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same length such that the number of columns is (4 * number of partitions displayed by the window) and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

dir = tempdir()
LNCSeqsADR<-system.file("extdata/",package="ftrCOOL")
fileLNC<-system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
mat<-ENUComposition_RNA(seqs = fileLNC, winSize=20,outFormat="mat")

ad<-paste0(dir,"/ENUCcompos.txt")
ENUComposition_RNA(seqs = fileLNC,outFormat="txt",winSize=20 ,outputFileDist=ad,overLap=FALSE)

unlink("dir", recursive = TRUE)

Expected ValKmerNUC_DNA

**Expected Value for K-mer Nucleotide (ExpectedValKmerNUC_DNA)**

Description

This function is introduced by this package for the first time. It computes the expected value for each k-mer in a sequence. ExpectedValue(k-mer) = freq(k-mer) / ( freq(nucleotide1) * freq(nucleotide2) * ... * freq(nucleotidek) )
Usage

ExpectedValKmerNUC_DNA(
    seqs,
    k = 4,
    ORF = FALSE,
    reverseORF = TRUE,
    normalized = TRUE,
    label = c()
)

Arguments

seqs is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

k is an integer value. The default is four.

ORF (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

reverseORF is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

normalized is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

The function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is \((4^k)\).

Examples

```r
fileLNC<system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
mat<ExpectedValKmerNUC_DNA(seqs=fileLNC,k=4,ORF=TRUE,reverseORF=FALSE)
```
ExpectedValKmerNUC_RNA

Description

This function is introduced by this package for the first time. It computes the expected value for each k-mer in a sequence. ExpectedValue(k-mer) = freq(k-mer) / ( freq(ribonucleotide1) * freq(ribonucleotide2) * ... * freq(ribonucleotidem) )

Usage

ExpectedValKmerNUC_RNA(
  seqs,  
  k = 4,  
  ORF = FALSE,  
  reverseORF = TRUE,  
  normalized = TRUE,  
  label = c()
)

Arguments

seqs is a FASTA file containing ribonucleotide sequences. The sequences start with '>'. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.

k is an integer value. The default is four.

ORF (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

reverseORF is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

normalized is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

The function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is (4^k).

Examples

fileLNC<-system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
mat<-ExpectedValKmerNUC_RNA(seqs=fileLNC,k=4,ORF=TRUE,reverseORF=FALSE)
ExpectedValueAA

**Description**

This function is introduced by this package for the first time. It computes the expected value for each k-mer in a sequence. ExpectedValue(k-mer) = freq(k-mer) / (c_1 * c_2 * ... * c_k), where c_i is the number of codons that encrypt the i'th amino acid in the k-mer.

**Usage**

```
ExpectedValueAA(seqs, k = 2, normalized = TRUE, label = c())
```

**Arguments**

- `seqs` is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- `k` is an integer value. The default is two.
- `normalized` is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.
- `label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

**Value**

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 20^k.

**Examples**

```
filePrs <- system.file("extdata/proteins.fasta", package="ftrCOOL")
mat<-ExpectedValueAA(seqs=filePrs,k=2,normalized=FALSE)
```

ExpectedValueGAA

**Description**

This function is introduced by this package for the first time. In this function, amino acids are first grouped into user-defined categories. Later, the expected value of grouped amino acids is computed. Please note that this function differs from Function `ExpectedValueAA` which works on individual amino acids.
Usage

ExpectedValueGAA(seqs, k = 3, Grp = "locFus", normalized = TRUE, label = c())

Arguments

seqs
is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

k
is an integer value. The default is three.

Grp
is a list of vectors containing amino acids. Each vector represents a category. Users can define a customized amino acid grouping, provided that the sum of all amino acids is 20 and there is no repeated amino acid in the groups. Also, users can choose 'cTriad'(conjointTriad), 'locFus', or 'aromatic'. Each option provides specific information about the type of an amino acid grouping.

normalized
is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.

label
is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details

for more information about ExpectedValueGAA, please refer to function ExpectedValueKmer.

Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is (number of categories)^k.

Examples

filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat1<-ExpectedValueGAA(seqs=filePrs,k=2,Grp="locFus")

mat2<-ExpectedValueGAA(seqs=filePrs,k=1,Grp=list(Grp1=c("G","A","V","L","M","I","F","Y","W"),Grp2=c("K","R","H","D","E"),Grp3=c("S","T","C","P","N","Q"))))

<table>
<thead>
<tr>
<th>ExpectedValueGKmerAA</th>
<th>Expected Value for Grouped K-mer Amino Acid(ExpectedValueGKmerAA)</th>
</tr>
</thead>
</table>

---
Description

This function is introduced by this package for the first time. In this function, amino acids are first grouped into user-defined categories. Later, the expected value of grouped k-mer is computed. Please note that this function differs from Function ExpectedValueKmerAA which works on individual amino acids.

Usage

```
ExpectedValueGKmerAA(
    seqs,
    k = 2,
    Grp = "locFus",
    normalized = TRUE,
    label = c()
)
```

Arguments

- **seqs** is a FASTA file with amino acid sequences. Each sequence starts with a ‘>’ character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- **k** is an integer. The default value is two.
- **Grp** is a list of vectors containing amino acids. Each vector represents a category. Users can define a customized amino acid grouping, provided that the sum of all amino acids is 20 and there is no repeated amino acid in the groups. Also, users can choose 'cTriad'(conjointTriad), 'locFus', or 'aromatic'. Each option provides specific information about the type of an amino acid grouping.
- **normalized** is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.
- **label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is (number of categorizes)^k.

Examples

```
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat1<-ExpectedValueGKmerAA(seqs=filePrs,k=2,Grp="locFus")

mat2<-ExpectedValueGKmerAA(seqs=filePrs,k=1,Grp=list(Grp1=c("G", "A", "V", "L", "M"),
                                                 Grp2=c("K", "R", "H", "D", "E"),
                                                 Grp3=c("S", "T", "C", "P", "N", "Q")))
```
**ExpectedValueKmerAA**

*Expected Value for K-mer Amino Acid (ExpectedValueKmerAA)*

**Description**

This function computes the expected value of each k-mer by dividing the frequency of the kmer to multiplying frequency of each amino acid of the k-mer in the sequence.

**Usage**

```r
ExpectedValueKmerAA(seqs, k = 2, normalized = TRUE, label = c())
```

**Arguments**

- **seqs** is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- **k** is an integer value and it shows the size of kmer in the kmer composition. The default value is 2.
- **normalized** is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.
- **label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

**Details**

```
ExpectedValue(k-mer) = freq(k-mer) / ( freq(aminoacid1) * freq(aminoacid2) * ... * freq(aminoacidk) )
```

**Value**

This function returns a feature matrix. The number of rows equals the number of sequences and the number of columns if upto set false, is 20^k.

**Examples**

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat<-ExpectedValueKmerAA(filePrs,k=2,normalized=FALSE)
```
Description

This function reads a FASTA file. Each sequence starts with '>' in the file. This is a general function which can be applied to all types of sequences (i.e., protein/peptide, dna, and rna).

Usage

fa.read(file, legacy.mode = TRUE, seqonly = FALSE, alphabet = "aa")

Arguments

- **file**: The address of the FASTA file.
- **legacy.mode**: comments all lines which start with ";".
- **seqonly**: if it is set to true, the function will return sequences with no description.
- **alphabet**: is a vector which contains amino acid, RNA, or DNA alphabets.

Value

a string vector such that each element is a sequence.

References

https://cran.r-project.org/web/packages/rDNAse/index.html

Examples

fileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
sequenceVectLNC<-fa.read(file=fileLNC,alphabet="dna")

filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
sequenceVectPROX<-fa.read(file=filePrs,alphabet="aa")
fickettScore

Fickett Score (fickettScore)

Description
This function calculates the ficket score of each sequence.

Usage
fickettScore(seqs, ORF = FALSE, reverseORF = TRUE, label = c())

Arguments
- **seqs**: is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
- **ORF** (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).
- **reverseORF**: is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).
- **label**: is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value
The function returns a feature vector. The length of the vector is equal to the number of sequences. Each entry in the vector contains the value of the fickett score.

Examples
```r
codelnc <- system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
vec <- fickettScore(seqs=codelnc,ORF=TRUE,reverseORF=FALSE)
```

GAAKpartComposition
Grouped Amino Acid K Part Composition (GAAKpartComposition)

Description
In this function, amino acids are first grouped into user-defined categories. Later, the composition of the grouped amino acid k part is computed. Please note that this function differs from AAKpartComposition which works on individual amino acids.
GAAKpartComposition

Usage

GAAKpartComposition(
  seqs,  
  k = 5, 
  normalized = TRUE, 
  Grp = "locFus", 
  label = c()
)

Arguments

seqs is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

k is an integer. Each sequence should be divided to k partition(s).

normalized is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.

Grp is a list of vectors containing amino acids. Each vector represents a category. Users can define a customized amino acid grouping, provided that the sum of all amino acids is 20 and there is no repeated amino acid in the groups. Also, users can choose 'cTriad'(conjointTriad), 'locFus', or 'aromatic'. Each option provides specific information about the type of an amino acid grouping.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

a feature matrix with k*(number of categorizes) number of columns. The number of rows is equal to the number of sequences.

Note

Warning: The length of all sequences should be greater than k.

Examples

filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat1<-GAAKpartComposition(seqs=filePrs,k=5,Grp="aromatic")

mat2<-GAAKpartComposition(seqs=filePrs,k=3,normalized=FALSE,Grp=
Grp3=c("S", "T", "C", "P", "N", "Q")))
Description

This function is introduced by this package for the first time. In this function, amino acids are first grouped into user-defined categories. Later, DDE is applied to grouped amino acids. Please note that this function differs from DDE which works on individual amino acids.

Usage

GrpDDE(seqs, Grp = "locFus", label = c())

Arguments

seqs is a FASTA file with amino acid sequences. Each sequence starts with a ‘>’ character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

Grp is a list of vectors containing amino acids. Each vector represents a category. Users can define a customized amino acid grouping, provided that the sum of all amino acids is 20 and there is no repeated amino acid in the groups. Also, users can choose ‘cTriad’(conjointTriad), 'locFus', or 'aromatic'. Each option provides specific information about the type of an amino acid grouping.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

A feature matrix with (number of categorizes)^2 number of columns. The number of rows is equal to the number of sequences.

Examples

filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat1<-GrpDDE(seqs=filePrs,Grp="aromatic")

mat2<-GrpDDE(seqs=filePrs,Grp=list(Grp1=c("G","A","V","L","M","I","F","Y","W"),Grp2=c("K","R","H","D","E"),Grp3=c("S","T","C","P","N","Q")))
G_Ccontent_DNA

G_C content in DNA (G_Ccontent_DNA)

Description

This function calculates G-C content of each sequence.

Usage

G_Ccontent_DNA(
  seqs,
  ORF = FALSE,
  reverseORF = TRUE,
  normalized = TRUE,
  label = c()
)

Arguments

seqs is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

ORF (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

reverseORF is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

normalized is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

The function returns a feature vector. The length of the vector is equal to the number of sequences. Each entry in the vector contains G-C content of a sequence.

Examples

fileLNC<system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
vect<G_Ccontent_DNA(seqs=fileLNC,ORF=TRUE,reverseORF=FALSE)
Description

This function calculates G-C content of each sequence.

Usage

G_Ccontent_RNA(
  seqs,
  ORF = FALSE,
  reverseORF = TRUE,
  normalized = TRUE,
  label = c()
)

Arguments

seqs is a FASTA file containing ribonucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.

ORF (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

reverseORF is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

normalized is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

The function returns a feature vector. The length of the vector is equal to the number of sequences. Each entry in the vector contains G-C content of a sequence.

Examples

fileLNC<-system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
vect<-G_Ccontent_RNA(seqs=fileLNC,ORF=TRUE,reverseORF=FALSE)
Description

This function calculates the frequency of all k-mers in the sequence(s).

Usage

```r
kAAComposition(seqs, rng = 3, upto = FALSE, normalized = TRUE, label = c())
```

Arguments

- `seqs` is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- `rng` This parameter can be a number or a vector. Each entry of the vector holds the value of k in the k-mer composition.
- `upto` It is a logical parameter. The default value is FALSE. If rng is a number and upto is set to TRUE, rng is converted to a vector with values from 1 to rng.
- `normalized` is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.
- `label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns depends on rng vector. For each value k in the vector, \((20)^k\) columns are created in the matrix.

Examples

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat1<-kAAComposition(seqs=filePrs,rng=3,upto=TRUE)
mat2<-kAAComposition(seqs=filePrs,rng=c(1,3),upto=TRUE)
```
**kGAAComposition**

**k Grouped Amino Acid Composition (kGAAComposition)**

**Description**

In this function, amino acids are first grouped into user-defined categories. Later, the composition of the k grouped amino acids is computed. Please note that this function differs from **kAAComposition** which works on individual amino acids.

**Usage**

```r
kGAAComposition(
  seqs,
  rng = 3,
  upto = FALSE,
  normalized = TRUE,
  Grp = "locFus",
  label = c()
)
```

**Arguments**

- `seqs` is a FASTA file with amino acid sequences. Each sequence starts with a ‘>’ character. Also, `seqs` could be a string vector. Each element of the vector is a peptide/protein sequence.
- `rng` This parameter can be a number or a vector. Each entry of the vector holds the value of k in the k-mer composition. For each k in the `rng` vector, a new vector (whose size is 20^k) is created which contains the frequency of k-mers.
- `upto` It is a logical parameter. The default value is FALSE. If `rng` is a number and `upto` is set to TRUE, `rng` is converted to a vector with values from 1 to `rng`.
- `normalized` is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.
- `Grp` is a list of vectors containing amino acids. Each vector represents a category. Users can define a customized amino acid grouping, provided that the sum of all amino acids is 20 and there is no repeated amino acid in the groups. Also, users can choose 'cTriad'(conjointTriad), 'locFus', or 'aromatic'. Each option provides specific information about the type of an amino acid grouping.
- `label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

**Details**

for more details, please refer to **kAAComposition**
KNNPeptide

Value
This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is \((\text{number of categories})^k\)\(^{\text{*}}\text{length of rng vector}\).

Examples

```r
cfilePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
cmat1<-CkSGApair(seqs=filePrs,rng=2,upto=TRUE,Grp="aromatic")
cmat2<-CkSGApair(seqs=filePrs,rng=c(1,3,5),Grp=
Grp3=c("S", "T", "C", "P", "N", "Q")))
```

KNNPeptide

\textit{K-Nearest Neighbor for Peptides (KNNPeptide)}

Description

This function needs an extra training data set and a label. We compute the similarity score of each input sequence with all sequences in the training data set. We use the BLOSUM62 matrix to compute the similarity score. The label shows the class of each sequence in the training data set. KNNPeptide finds the label of 1. It reports the frequency of each class for each k.

Usage

```
KNNPeptide(seqs, trainSeq, percent = 30, label = c(), labeltr = c())
```

Arguments

- **seqs** is a fasta file with amino acids sequences. Each sequence starts with a '>' character or it is a string vector such that each element is a peptide or protein sequence.
- **trainSeq** is a fasta file with amino acids sequences. Each sequence starts with a '>' character. Also it could be a string vector such that each element is a peptide sequence. Each sequence in the training set is associated with a label. The label is found in the parameter labeltr.
- **percent** determines the threshold which is used to identify sequences (in the training set) which are similar to the input sequence.
- **label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
- **labeltr** This parameter is a vector whose length is equivalent to the number of sequences in the training set. It shows class of each sequence in the training set.

Value

This function returns a feature matrix such that number of columns is number of classes multiplied by percent and number of rows is equal to the number of the sequences.
**KNNProtein**

**Note**

This function is usable for amino acid sequences with the same length in both training data set and the set of sequences.

**References**


**Examples**

```r
ptmSeqsADR <- system.file("extdata/", package="ftrCOOL")
ptmSeqsVect <- as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
posSeqs <- as.vector(read.csv(paste0(ptmSeqsADR,"/poSeqPTM101.csv"))[1:10])
negSeqs <- as.vector(read.csv(paste0(ptmSeqsADR,"/negSeqPTM101.csv"))[1:10])

posSeqs <- posSeqs[1:10]
negSeqs <- negSeqs[1:10]

trainSeq <- c(posSeqs, negSeqs)

labelPos <- rep(1, length(posSeqs))
labelNeg <- rep(0, length(negSeqs))

labeltr <- c(labelPos, labelNeg)

KNNPeptide(seqs = ptmSeqsVect, trainSeq = trainSeq, percent = 10, labeltr = labeltr)
```

---

**KNNProtein**  
*K-Nearest Neighbor for Protein (KNNProtein)*

**Description**

This function is like KNNPeptide with the difference that similarity score is computed by Needleman-Wunsch algorithm.

**Usage**

```
KNNProtein(seqs, trainSeq, percent = 30, labeltr = c(), label = c())
```
Arguments

**seqs** is a fasta file with amino acids sequences. Each sequence starts with a '>' character. Also it could be a string vector such that each element is a protein sequence.

**trainSeq** is a fasta file with amino acids sequences. Each sequence starts with a '>' character. Also it could be a string vector such that each element is a protein sequence. Each sequence in the training set is associated with a label. The label is found in the parameter `labeltr`.

**percent** determines the threshold which is used to identify sequences (in the training set) which are similar to the input sequence.

**labeltr** This parameter is a vector whose length is equivalent to the number of sequences in the training set. It shows class of each sequence in the training set.

**label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

This function returns a feature matrix such that number of columns is number of classes multiplied by percent and number of rows is equal to the number of the sequences.

References


Examples

```r
ptmSeqsADR <- system.file("extdata/", package="FtrCOOL")
ptmSeqsVect <- as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
ptmSeqsVect <- sapply(ptmSeqsVect, function(seq){substr(seq,1,31)})

posSeqs <- as.vector(read.csv(paste0(ptmSeqsADR,"/poSeqPTM101.csv"))[,2])
negSeqs <- as.vector(read.csv(paste0(ptmSeqsADR,"/negSeqPTM101.csv"))[,2])

posSeqs <- sapply(posSeqs, function(seq){substr(seq,1,31)})
negSeqs <- sapply(negSeqs, function(seq){substr(seq,1,31)})

trainSeq <- c(posSeqs, negSeqs)
labelPos <- rep(1, length(posSeqs))
labelNeg <- rep(0, length(negSeqs))

labeltr <- c(labelPos, labelNeg)

mat <- KNNProtein(seqs = ptmSeqsVect, trainSeq = trainSeq, percent = 5, labeltr = labeltr)
```
**KNN_DNA**

**K-Nearest Neighbor_DNA (KNN_DNA)**

**Description**

This function is like KNNPeptide with the difference that similarity score is computed by Needleman-Wunsch algorithm.

**Usage**

\[
\text{KNN_DNA(seqs, trainSeq, percent = 30, labeltr = c(), label = c())}
\]

**Arguments**

- **seqs** is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
- **trainSeq** is a fasta file with nucleotide sequences. Each sequence starts with a ‘>’ character. Also it could be a string vector such that each element is a nucleotide sequence. Each sequence in the training set is associated with a label. The label is found in the parameter labeltr.
- **percent** determines the threshold which is used to identify sequences (in the training set) which are similar to the input sequence.
- **labeltr** This parameter is a vector whose length is equivalent to the number of sequences in the training set. It shows class of each sequence in the training set.
- **label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

**Value**

This function returns a feature matrix such that number of columns is number of classes multiplied by percent and number of rows is equal to the number of the sequences.

**References**


**Examples**

```r
ptmSeqsADR<system.file("extdata/", package="ftrCOOL")
seqs<-fa.read(file=paste0(ptmSeqsADR,"/testData51.txt"),alphabet="dna")

posSeqs<-fa.read(file=paste0(ptmSeqsADR,"/posData51.txt"),alphabet="dna")
negSeqs<-fa.read(file=paste0(ptmSeqsADR,"/negData51.txt"),alphabet="dna")
```
trainSeq<-c(posSeqs,negSeqs)
labelPos<-rep(1,length(posSeqs))
labelNeg<-rep(0,length(negSeqs))
labeltr<-c(labelPos,labelNeg)
KNN_DNA(seqs=seqs,trainSeq=trainSeq,percent=5,labeltr=labeltr)

KNN_RNA

K-Nearest Neighbor RNA (KNN_RNA)

Description
This function is like KNNPeptide with the difference that similarity score is computed by Needleman-Wunsch algorithm.

Usage
KNN_RNA(seqs, trainSeq, percent = 30, labeltr = c(), label = c())

Arguments
seqs is a FASTA file containing ribonucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.

trainSeq is a fasta file with ribonucleotide sequences. Each sequence starts with a ‘>’ character. Also it could be a string vector such that each element is a ribonucleotide sequence. Each sequence in the training set is associated with a label. The label is found in the parameter labeltr.

percent determines the threshold which is used to identify sequences (in the training set) which are similar to the input sequence.

labeltr This parameter is a vector whose length is equivalent to the number of sequences in the training set. It shows class of each sequence in the training set.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value
This function returns a feature matrix such that number of columns is number of classes multiplied by percent and number of rows is equal to the number of the sequences.

References
**Examples**

```r
ptmSeqsADR <- system.file("extdata/", package="ftrCOOL")
posSeqs <- fa.read(file=paste0(ptmSeqsADR,"/pos2RNA51.txt"), alphabet="rna")
negSeqs <- fa.read(file=paste0(ptmSeqsADR,"/neg2RNA51.txt"), alphabet="rna")
seqs <- fa.read(file=paste0(ptmSeqsADR,"/testSeq2RNA51.txt"), alphabet="rna")

trainSeq <- c(posSeqs, negSeqs)

labelPos <- rep(1, length(posSeqs))
labelNeg <- rep(0, length(negSeqs))

labeltr <- c(labelPos, labelNeg)

KNN_RNA(seqs=seqs, trainSeq=trainSeq, percent=10, labeltr=labeltr)
```

---

**kNUComposition_DNA**

**k Nucleotide Composition (kNUComposition_DNA)**

**Description**

This function calculates the frequency of all k-mers in the sequence.

**Usage**

```r
kNUComposition_DNA(
  seqs, 
  rng = 3, 
  reverse = FALSE, 
  upto = FALSE, 
  normalized = TRUE, 
  ORF = FALSE, 
  reverseORF = TRUE, 
  label = c()
)
```

**Arguments**

- `seqs` is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, `seqs` could be a string vector. Each element of the vector is a nucleotide sequence.
- `rng` This parameter can be a number or a vector. Each entry of the vector holds the value of k in the k-mer composition. For each k in the `rng` vector, a new vector (whose size is $4^k$) is created which contains the frequency of kmers.
reverse
It is a logical parameter which assumes the reverse complement of the sequence.

upto
It is a logical parameter. The default value is FALSE. If rng is a number and
upto is set to TRUE, rng is converted to a vector with values from 1 to rng.

normalized
is a logical parameter. When it is FALSE, the return value of the function does
not change. Otherwise, the return value is normalized using the length of the
sequence.

ORF
(Open Reading Frame) is a logical parameter. If it is set to true, ORF region of
each sequence is considered instead of the original sequence (i.e., 3-frame).

reverseORF
is a logical parameter. It is enabled only if ORF is true. If reverseORF is true,
ORF region will be searched in the sequence and also in the reverse complement
of the sequence (i.e., 6-frame).

label
is an optional parameter. It is a vector whose length is equivalent to the number
of sequences. It shows the class of each entry (i.e., sequence).

Value
This function returns a feature matrix. The number of rows is equal to the number of sequences and
the number of columns depends on the rng vector. For each value k in the vector, \((4)^k\) columns are
created in the matrix.

Examples

```r
defileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL"
mat<-kNUComposition_DNA(seqs=defileLNC,rng=c(1,3))
```
Arguments

seqs is a FASTA file containing ribonucleotide sequences. The sequences start with '>' . Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.

rng This parameter can be a number or a vector. Each entry of the vector holds the value of k in the k-mer composition. For each k in the rng vector, a new vector (whose size is \(4^k\)) is created which contains the frequency of kmers.

reverse It is a logical parameter which assumes the reverse complement of the sequence.

upto It is a logical parameter. The default value is FALSE. If rng is a number and upto is set to TRUE, rng is converted to a vector with values from 1 to rng.

normalized is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.

ORF (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

reverseORF is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns depends on the rng vector. For each value k in the vector, \(4^k\) columns are created in the matrix.

Examples

```r
geneLNC<-system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
mat<-kNUComposition_RNA(seqs=geneLNC,rng=c(1,3))
```

LocalPoSpKAFF

<table>
<thead>
<tr>
<th>Local PoSpKAFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Position Specific k Amino Acids Frequency (LocalPoSpKAFF)</td>
</tr>
</tbody>
</table>

Description

For each sequence, this function creates a feature vector denoted as \((f_1, f_2, f_3, \ldots, f_N)\), where \(f_i = \text{freq}(i^{th} \text{ k-mer of the sequence}) / i\). It should be applied to sequences with the same length.

Usage

```r
LocalPoSpKAFF(seqs, k = 2, label = c(), outFormat = "mat", outputFileDist = "")
```
Arguments

seqs    is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

k       is a numeric value which holds the value of k in the k-mers.

label   is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

outFormat (output format) can take two values: 'mat' (matrix) and 'txt'. The default value is 'mat'.

outputFileDist shows the path and name of the 'txt' output file.

Value

The output depends on the outFormat parameter which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same length such that the number of columns is (sequence length-k+1) and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

dir = tempdir()
ptmSeqsADR<-system.file("extdata/",package="ftrCOOL")
ptmSeqsVect<-as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
mat<-LocalPoSpKAAF(seqs = ptmSeqsVect, k=2,outFormat="mat")
ad<-paste0(dir,"/LocalPoSpKaaF.txt")
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
LocalPoSpKAAF(seqs = filePrs, k=1,outFormat="txt" ,outputFileDist=ad)
unlink("dir", recursive = TRUE)
Local PoSpKNUCF_DNA

Local Position Specific k Nucleotide Frequency (LocalPoSpKNUCF_DNA)

Description

For each sequence, this function creates a feature vector denoted as \((f_1, f_2, f_3, \ldots, f_N)\), where \(f_i = \text{freq}(i^{th} \text{k-mer of the sequence}) / i\). It should be applied to sequences with the same length.

Usage

LocalPoSpKNUCF_DNA(
  seqs,
  k = 2,
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)

Arguments

seqs is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

k is a numeric value which holds the value of k in the k-mers.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

outFormat (output format) can take two values: ‘mat’ (matrix) and ‘txt’. The default value is ‘mat’.

outputFileDist shows the path and name of the ‘txt’ output file.

Value

The output depends on the outFormat parameter which can be either ‘mat’ or ‘txt’. If outFormat is ‘mat’, the function returns a feature matrix for sequences with the same length such that the number of columns is (sequence length-k+1) and the number of rows is equal to the number of sequences. If the outFormat is ‘txt’, the output is written to a tab-delimited file.

Note

This function is provided for sequences with the same lengths. Users can use ‘txt’ option in outFormat for sequences with different lengths. Warning: If outFormat is set to ‘mat’ for sequences with different lengths, it returns an error. Also, when output format is ‘txt’, label information is not shown in the text file. It is noteworthy that ‘txt’ format is not usable for machine learning purposes if sequences have different sizes. Otherwise ‘txt’ format is also usable for machine learning purposes.
LocalPoSpKNUCF_RNA

Examples

```r
dir = tempdir()
LNCSeqsADR <- system.file("extdata/", package="ftrCOOL")
LNC50Nuc <- as.vector(read.csv(paste0(LNCSeqsADR,"/LNC50Nuc.csv"))[,2])
mat <- LocalPoSpKNUCF_DNA(seqs = LNC50Nuc, k=2, outFormat="mat")

ad <- paste0(dir,"/LocalPoSpKnucF.txt")
fileLNC <- system.file("extdata/Athaliana_LNCRNA.fa", package="ftrCOOL")
LocalPoSpKNUCF_DNA(seqs = fileLNC, k=1, outFormat="txt", outputFileDist = ad)

unlink("dir", recursive = TRUE)
```

LocalPoSpKNUCF_RNA  
Local Position Specific k riboNucleotide Frequency (LocalPoSpKNUCF_RNA)

Description

For each sequence, this function creates a feature vector denoted as \( (f_1, f_2, f_3, \ldots, f_N) \), where \( f_i = \frac{\text{freq}(i^{\text{th}} \text{k-mer of the sequence})}{i} \). It should be applied to sequences with the same length.

Usage

```r
LocalPoSpKNUCF_RNA(
  seqs,
  k = 2,
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

Arguments

- **seqs** is a FASTA file containing ribonucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.
- **k** is a numeric value which holds the value of k in the k-mers.
- **label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
- **outFormat** (output format) can take two values: ‘mat’ (matrix) and ‘txt’. The default value is ‘mat’.
- **outputFileDist** shows the path and name of the ‘txt’ output file.
The output depends on the outFormat parameter which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same length such that the number of columns is (sequence length-k+1) and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

```r
# Define the directory
dir = tempdir()
# Load the file
fileLNC = system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
# Execute the function
mat <- LocalPoSpKNUCF_RNA(seqs = fileLNC, k=2,outFormat="mat")
# Output file path
ad <- paste0(dir,"/LocalPoSpKnucF.txt")
# Run the function with 'txt' output format and label information
LocalPoSpKNUCF_RNA(seqs = fileLNC,k=1,outFormat="txt",outputFileDist=ad)
# Clean up
unlink("dir", recursive = TRUE)
```

---

### maxORF

**Maximum Open Reading Frame in DNA (maxORF)**

**Description**

This function gets a sequence as the input. If reverse is true, the function extracts the max Open Reading Frame in the sequence and its reverse complement (hint: Six frames). Otherwise, only the sequence is searched (hint: Three frames).

**Usage**

```r
maxORF(seqs, reverse = TRUE, label = c())
```

**Arguments**

- **seqs** is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
- **reverse** is a logical parameter which assumes the reverse complement of the sequence.
- **label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
maxORFlength_DNA

Value

A vector containing a subsequence for each given sequences. The subsequence is the maximum ORF of the sequence.

Note

If a sequence does not contain ORF, the function deletes the sequence.

Examples

```r
fileLNC <- system.file("extdata/Athaliana_LNCRNA.fa", package="ftrCOOL")
ORF <- maxORF(seqs = fileLNC, reverse = FALSE)
```

maxORFlength_DNA

Maximum Open Reading Frame length in DNA (maxORFlength_DNA)

Description

This function returns the length of the maximum Open Reading Frame for each sequence. If reverse is FALSE, ORF region will be searched in a sequence. Otherwise, it will be searched both in the sequence and its reverse complement.

Usage

```r
maxORFlength_DNA(seqs, reverse = TRUE, normalized = FALSE, label = c())
```

Arguments

- `seqs` is a FASTA file containing nucleotide sequences. The sequences start with '>'.
  Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
- `reverse` is a logical parameter which assumes the reverse complement of the sequence.
- `normalized` is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.
- `label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

A vector containing the lengths of maximum ORFs for each sequence.
Examples

```r
fileLNC <- system.file("extdata/Athaliana_LNCRNA.fa", package = "ftrCOOL")
vect <- maxORFlength_DNA(seqs = fileLNC, reverse = TRUE, normalized = TRUE)
```

---

**Description**

This function returns the length of the maximum Open Reading Frame for each sequence. If reverse is FALSE, ORF region will be searched in a sequence. Otherwise, it will be searched both in the sequence and its reverse complement.

**Usage**

```r
maxORFlength_RNA(seqs, reverse = TRUE, normalized = FALSE, label = c())
```

**Arguments**

- `seqs`: is a FASTA file containing ribonucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.
- `reverse`: It is a logical parameter which assumes the reverse complement of the sequence.
- `normalized`: is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.
- `label`: is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

**Value**

A vector containing the lengths of maximum ORFs for each sequence.

**Examples**

```r
fileLNC <- system.file("extdata/Carica_papaya101RNA.txt", package = "ftrCOOL")
vect <- maxORFlength_RNA(seqs = fileLNC, reverse = TRUE, normalized = TRUE)
```
maxORF_RNA

Maximum Open Reading Frame in RNA (maxORF_RNA)

Description

This function gets a sequence as the input. If reverse is true, the function extracts the max Open Reading Frame in the sequence and its reverse complement (hint: Six frames). Otherwise, only the sequence is searched (hint: Three frames).

Usage

maxORF_RNA(seqs, reverse = TRUE, label = c())

Arguments

seqs  
is a FASTA file containing ribonucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.

reverse  
It is a logical parameter which assumes the reverse complement of the sequence.

label  
is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

A vector containing a subsequence for each given sequences. The subsequence is the maximum ORF of the sequence.

Note

If a sequence does not contain ORF, the function deletes the sequence.

Examples

fileLNC<-system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
ORF<-maxORF_RNA(seqs=fileLNC,reverse=FALSE)
Mismatch_DNA

**Description**

This function also calculates the frequencies of all k-mers in the sequence but allows maximum m mismatches. m<k.

**Usage**

Mismatch_DNA(seqs, k = 3, m = 2, label = c())

**Arguments**

- **seqs** is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
- **k** This parameter can be a number which shows kmer.
- **m** This parameter shows maximum number of mismatches.
- **label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

**Value**

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns depends on the rng vector. For each value k in the vector, \(4^k\) columns are created in the matrix.

**References**


**Examples**

```r
fileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
mat<-Mismatch_DNA(seqs=fileLNC)
```
Mismatch_RNA

Description

This function also calculates the frequencies of all k-mers in the sequence but allows maximum m mismatches. m<k.

Usage

Mismatch_RNA(seqs, k = 3, m = 2, label = c())

Arguments

seqs is a FASTA file containing ribonucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.

k This parameter can be a number which shows kmer.

m This parameter shows maximum number of mismatches.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns depends on the rng vector. For each value k in the vector, (4)^k columns are created in the matrix.

References


Examples

fileLNC<- system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
mat<-Mismatch_RNA(seqs=fileLNC)
**MMI_DNA**

**Multivariate Mutual Information_DNA (MMI_DNA)**

**Description**

MMI computes mutual information based on 2-mers $T_2 = AA, AC, AG, AT, CC, CG, CT, GG, GT, TT$ and 3-mers $T_3 = AAA, AAC, AAG, AAT, ACC, ACG, ACT, AGG, AGT, ATT, CCC, CCG, CCT, CGG, CGT, CTT, GGG, GGT, GTT$ and TTT for more information please check the reference part.

**Usage**

```r
MMI_DNA(seqs, label = c())
```

**Arguments**

- **seqs**: is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
- **label**: is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

**Value**

It is a feature matrix. The number of columns is 30 and the number of rows is equal to the number of sequences.

**References**

Zhen Chen, Pei Zhao, Chen Li, Fuyi Li, Dongxu Xiang, Yong-Zi Chen, Tatsuya Akutsu, Roger J Daly, Geoffrey I Webb, Quanzhi Zhao, Lukasz Kurgan, Jiangning Song. iLearnPlus: a comprehensive and automated machine-learning platform for nucleic acid and protein sequence analysis, prediction and visualization, Nucleic Acids Research (2021).

**Examples**

```r
fileLNC <- system.file("extdata/Athaliana_LNCRNA.fa", package="ftrCOOL")
mat <- MMI_DNA(seqs = fileLNC)
```
Description

MMI computes mutual information based on 2-mers $T_2 = \text{AA, AC, AG, AU, CC, CG, CU, GG, GU, U}$ and 3-mers $T_3 = \text{AAA, AAC, AAG, AAU, ACC, ACG, ACU, AGG, AGU, AUU, CCC, CCG, CCU, CGG, CGU, CUU, GGG, GGU, GUU and UUU}$ for more information please check the reference part.

Usage

```r
MMI_RNA(seqs, label = c())
```

Arguments

- `seqs`: is a FASTA file containing ribonucleotide sequences. The sequences start with ‘>’. Also, `seqs` could be a string vector. Each element of the vector is a ribonucleotide sequence.
- `label`: is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

It is a feature matrix. The number of columns is 30 and the number of rows is equal to the number of sequences.

References

Zhen Chen, Pei Zhao, Chen Li, Fuyi Li, Dongxu Xiang, Yong-Zi Chen, Tatsuya Akutsu, Roger J Daly, Geoffrey I Webb, Quanzhi Zhao, Lukasz Kurgan, Jiangning Song. iLearnPlus: a comprehensive and automated machine-learning platform for ribonucleic acid and protein sequence analysis, prediction and visualization, Nucleic Acids Research (2021).

Examples

```r
cat <- system.file("extdata/Carica_papaya101RNA.txt", package="ftrCOOL")
mat <- MMI_RNA(seqs=cat)
```
nameKmer  

**nameKmer (nameKmer)**

**Description**

This function creates all possible k-combinations of the given alphabets.

**Usage**

```r
nameKmer(k = 3, type = "aa", num = 0)
```

**Arguments**

- **k** is a numeric value.
- **type** can be one of "aa", "rna", "dna", or "num".
- **num** When type is set to "num", it shows the numeric alphabet(1,...,num).

**Value**

a string vector of length \(20^k\) for 'aa' type), \(4^k\) for 'dna' type), \(4^k\) for 'rna' type), and \(num^k\) for 'num' type).

**Examples**

```r
all_kmersAA<-nameKmer(k=2,type="aa")
all_kmersDNA<-nameKmer(k=3,type="dna")
all_kmersNUM<-nameKmer(k=3,type="num",num=2)
```

---

NCP_DNA  

Nucleotide Chemical Property (NCP_DNA)

**Description**

This function replaces nucleotides with a three-length vector. The vector represent the nucleotides such that 'A' will be replaced with c(1, 1, 1), 'C' with c(0, 1, 0), 'G' with c(1, 0, 0), and 'T' with c(0, 0, 1).
Usage

NCP_DNA(
  seqs,
  binaryType = "numBin",
  outFormat = "mat",
  outputFileDist = "",
  label = c()
)

Arguments

seqs is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

binaryType It can take any of the following values: (‘strBin’, ‘logicBin’, ‘numBin’). ‘strBin’ (String binary): each nucleotide is represented by a string containing 4 characters (0-1). A = "0001", C = "0010", G = "0100", T = "1000" ‘logicBin’ (logical value): Each nucleotide is represented by a vector containing 4 logical entries. A = c(F,F,F,T), ... , T = c(T,F,F,F) ‘numBin’ (numeric bin): Each nucleotide is represented by a numeric (i.e., integer) vector containing 4 numerals. A = c(0,0,0,1), ... , T = c(1,0,0,0)

outFormat (output format) can take two values: ‘mat’ (matrix) and ‘txt’. The default value is ‘mat’.

outputFileDist shows the path and name of the ‘txt’ output file.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

The output is different depending on the outFormat parameter (‘mat’ or ‘txt’). If outFormat is set to ‘mat’, it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is ‘strBin’, the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences)*3. If outFormat is ‘txt’, all binary values will be written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

Note

This function is provided for sequences with the same lengths. Users can use ‘txt’ option in outFormat for sequences with different lengths. Warning: If outFormat is set to ‘mat’ for sequences with different lengths, it returns an error. Also, when output format is ‘txt’, label information is not shown in the text file. It is noteworthy that ‘txt’ format is not usable for machine learning purposes if sequences have different sizes. Otherwise ‘txt’ format is also usable for machine learning purposes.
References


Examples

```r
dir = tempdir()
LNCSeqsADR<-system.file("extdata/",package="ftrCOOL")
LNC50Nuc<-as.vector(read.csv(paste0(LNCSeqsADR,"/LNC50Nuc.csv"))[2])
mat<-NCP_DNA(seqs = LNC50Nuc,binaryType="strBin",outFormat="mat")

ad<-paste0(dir,"/NCP.txt")
fileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
NCP_DNA(seqs = fileLNC,binaryType="numBin",outFormat="txt",outputFileDist=ad)
unlink("dir", recursive = TRUE)
```

NCP_RNA

### riboNucleotide Chemical Property (NCP_RNA)

**Description**

This function replaces ribonucleotides with a three-length vector. The vector represent the ribonucleotides such that 'A' will be replaced with c(1, 1, 1), 'C' with c(0, 1, 0), 'G' with c(1, 0, 0), and 'U' with c(0, 0, 1).

**Usage**

```r
NCP_RNA(
  seqs,
  binaryType = "numBin",
  outFormat = "mat",
  outputFileDist = ",",
  label = c()
)
```

**Arguments**

- `seqs` is a FASTA file containing ribonucleotide sequences. The sequences start with '>'. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.
- `binaryType` It can take any of the following values: ('strBin','logicBin','numBin'). 'strBin'(String binary): each ribonucleotide is represented by a string containing 4 characters(0-1). A = "0001", C = "0010", G = "0100", T = "1000" 'logicBin'(logical value): Each ribonucleotide is represented by a vector containing 4 logical entries. A
NCP_RNA

= c(F,F,F,T) , ... , T = c(T,F,F,F) 'numBin' (numeric bin): Each ribonucleotide is represented by a numeric (i.e., integer) vector containing 4 numerals. A = c(0,0,0,1) , ... , T = c(1,0,0,0)

outFormat (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

outputFileDist shows the path and name of the 'txt' output file.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences)*3. If outFormat is 'txt', all binary values will be written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

References


Examples

dir = tempdir()
fileLNC<-system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
mat<-NCP_RNA(seqs = fileLNC,binaryType="strBin",outFormat="mat")
ad<-paste0(dir,"/NCP.txt")
NCP_RNA(seqs = fileLNC,binaryType="numBin",outFormat="txt",outputFileDist=ad)
unlink("dir", recursive = TRUE)
**needleman**  

Needleman-Wunsch (needleman)

---

**Description**

This function works based on Needleman-Wunsch algorithm which computes similarity score of two sequences.

**Usage**

```r
needleman(seq1, seq2, gap = -1, mismatch = -1, match = 1)
```

**Arguments**

- `seq1` (sequence1) is a string.
- `seq2` (sequence2) is a string.
- `gap` The penalty for gaps in sequence alignment. Usually, it is a negative value.
- `mismatch` The penalty for the mismatch in the sequence alignment. Usually, it is a negative value.
- `match` A score for the match in sequence alignment. Usually, it is a positive value.

**Value**

The function returns a number which indicates the similarity between `seq1` and `seq2`.

**References**

https://gist.github.com/juliuskittler/ed53696ac1e590b413aac2dddf0457f6

**Examples**

```r
simScore<-needleman(seq1="Hello",seq2="Hello",gap=-1,mismatch=-2,match=1)
```

---

**nonStandardSeq**  

nonStandard sequence (nonStandardSeq)

---

**Description**

This function returns sequences which contain at least one non-standard alphabet.

**Usage**

```r
nonStandardSeq(file, legacy.mode = TRUE, seqonly = FALSE, alphabet = "aa")
```
Arguments

file  The address of fasta file which contains all the sequences.
legacy.mode  It comments all lines starting with ";"
seqonly  If it is set to true, the function returns sequences with no description.
alphabet  It is a vector which contains the amino acid, RNA, or DNA alphabets.

Value

This function returns a string vector. Each element of the vector is a sequence which contains at least one non-standard alphabet.

Examples

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
nonStandardPrSeq<-nonStandardSeq(file = filePrs,alphabet="aa")
fileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
nonStandardNUCSeq<-nonStandardSeq(file = fileLNC, alphabet="dna")
```

NUC2Binary_DNA  Nucleotide To Binary (NUC2Binary_DNA)

Description

This function transforms a nucleotide to a binary format. The type of the binary format is determined by the binaryType parameter. For details about each format, please refer to the description of the binaryType parameter.

Usage

```r
NUC2Binary_DNA(
  seqs,
  binaryType = "numBin",
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

Arguments

seqs  is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
NUC2Binary_DNA

binaryType It can take any of the following values: ('strBin','logicBin','numBin'). 'strBin'(String binary): each nucleotide is represented by a string containing 4 characters(0-1). A = "0001", C = "0010", G = "0100", T = "1000" 'logicBin'(logical value): Each nucleotide is represented by a vector containing 4 logical entries. A = c(F,F,F,T), ... , T = c(T,F,F,F) 'numBin' (numeric bin): Each nucleotide is represented by a numeric (i.e., integer) vector containing 4 numerals. A = c(0,0,0,1), ... , T = c(1,0,0,0)

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

outFormat (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

outputFileDist shows the path and name of the 'txt' output file.

Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences)*4. If outFormat is 'txt', all binary values will be written to a 'txt' file. Each line in the file shows the binary format of a sequence.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat parameter for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes.

Examples

dir = tempdir()
LNCSeqsADR<-system.file("extdata/",package="ftrCOOL")
LNC50Nuc<-as.vector(read.csv(paste0(LNCSeqsADR,"/LNC50Nuc.csv"))[,2])
mat<-NUC2Binary_DNA(seqs = LNC50Nuc,outFormat="mat")
ad<-paste0(dir,"/NUC2Binary.txt")
fileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
NUC2Binary_DNA(seqs = fileLNC,binaryType="numBin",outFormat="txt",outputFileDist=ad)
unlink("dir", recursive = TRUE)
Description

This function transforms a ribonucleotide to a binary format. The type of the binary format is determined by the binaryType parameter. For details about each format, please refer to the description of the binaryType parameter.

Usage

```r
NUC2Binary_RNA(
  seqs,
  binaryType = "numBin",
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

Arguments

- `seqs` is a FASTA file containing ribonucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.
- `binaryType` can take any of the following values: ('strBin', 'logicBin', 'numBin'). 'strBin' (String binary): each ribonucleotide is represented by a string containing 4 characters (0-1). A = "0001", C = "0010", G = "0100", U = "1000". 'logicBin' (logical value): Each ribonucleotide is represented by a vector containing 4 logical entries. A = c(F,F,F,T), ... , U = c(T,F,F,F). 'numBin' (numeric bin): Each ribonucleotide is represented by a numeric (i.e., integer) vector containing 4 numerals. A = c(0,0,0,1), ... , U = c(1,0,0,0)
- `label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
- `outFormat` (output format) can take two values: 'mat' (matrix) and 'txt'. The default value is 'mat'.
- `outputFileDist` shows the path and name of the 'txt' output file.

Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences)*4. If outFormat is 'txt', all binary values will be written to a 'txt' file. Each line in the file shows the binary format of a sequence.
Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in out-
Format parameter for sequences with different lengths. Warning: If outFormat is set to 'mat' for
sequences with different lengths, it returns an error. Also, when output format is 'txt', label infor-
mation is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine
learning purposes.

Examples

```r
dir = tempdir()
fileLNC<-system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
mat<-NUC2Binary_RNA(seqs = fileLNC,outFormat="mat")
ad<-paste0(dir,"/NUC2Binary.txt")
NUC2Binary_RNA(seqs = fileLNC,binaryType="numBin",outFormat="txt",outputFileDist=ad)

unlink("dir", recursive = TRUE)
```

NUCKpartComposition_DNA

Nucleotide to K Part Composition (NUCKpartComposition_DNA)

Description

In this function, each sequence is divided into k equal partitions. The length of each part is equal to
ceiling(l(lenght of the sequence)/k). The last part can have a different length containing the residual
nucleotides. The nucleotide composition is calculated for each part.

Usage

```r
NUCKpartComposition_DNA(
    seqs,
    k = 5,
    ORF = FALSE,
    reverseORF = TRUE,
    normalized = TRUE,
    label = c()
)
```

Arguments

- **seqs**: is a FASTA file containing nucleotide sequences. The sequences start with ‘>’.
  Also, seqs could be a string vector. Each element of the vector is a nucleotide
  sequence.
- **k**: is an integer value. Each sequence should be divided to k partition(s).
NUCKpartComposition_RNA

ORF (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

reverseORF is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

normalized is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

a feature matrix with k*4 number of columns. The number of rows is equal to the number of sequences.

Note

Warning: The length of all sequences should be greater than k.

Examples

```r
fileLNC <- system.file("extdata/Athaliana_LNCRNA.fa", package="ftrCOOL")
mat <- NUCKpartComposition_DNA(seqs = fileLNC, k = 5, ORF = TRUE, reverseORF = FALSE, normalized = FALSE)
```

Description

In this function, each sequence is divided into k equal partitions. The length of each part is equal to ceiling(length of the sequence)/k). The last part can have a different length containing the residual ribonucleotides. The ribonucleotide composition is calculated for each part.

Usage

```r
NUCKpartComposition_RNA(
  seqs,
  k = 5,
  ORF = FALSE,
  reverseORF = TRUE,
  normalized = TRUE,
  label = c()
)
```
**Arguments**

- **seqs** is a FASTA file containing ribonucleotide sequences. The sequences start with '>' . Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.
- **k** is an integer value. Each sequence should be divided to k partition(s).
- **ORF** (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).
- **reverseORF** is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).
- **normalized** is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.
- **label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

**Value**

A feature matrix with k*4 number of columns. The number of rows is equal to the number of sequences.

**Note**

Warning: The length of all sequences should be greater than k.

**Examples**

```r
fileLNC<system.file("extdata/Carica_papaya101RNA.txt",package="frtCOOL")
mat<NUCKpartComposition_RNA(seqs=fileLNC,k=5,ORF=TRUE,reverseORF=FALSE,normalized=FALSE)
```

**Description**

This group of functions (OPF Group) categorize amino acids in different groups based on the type. This function includes 10 amino acid properties. OPF_10bit substitutes each amino acid with a 10-dimensional vector. Each element of the vector shows if that amino acid locates in a special property category or not. '0' means that amino acid is not located in that property group and '1' means it is located.

**Usage**

```r
OPF_10bit(seqs, label = c(), outFormat = "mat", outputFileDist = "")
```
Arguments

segs is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, segs could be a string vector. Each element of the vector is a peptide/protein sequence.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

outFormat (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

outputFileDist shows the path and name of the 'txt' output file.

Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. Number of columns for this feature matrix is equal to (length of the sequences)*10 and number of rows is equal to the number of sequences. If outFormat is 'txt', all binary values will be written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

References


Examples

```
ptmSeqsADR<-system.file("extdata/",package="ftrCOOL")
ptmSeqsVect<-as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
mat<-OPF_10bit(seqs = ptmSeqsVect,outFormat="mat")
```
**OPF_7bit_T1**

**Description**

This group of functions (OPF Group) categorize amino acids in different groups based on the type. This function includes 7 amino acid properties. OPF_7bit_T1 substitutes each amino acid with a 7-dimensional vector. Each element of the vector shows if that amino acid locates in a special property category or not. '0' means that amino acid is not located in that property group and '1' means it is located. The only difference between OPF_7bit type1, type2, and type3 is in localization of amino acids in the properties groups.

**Usage**

```
OPF_7bit_T1(seqs, label = c(), outFormat = "mat", outputFileDist ="")
```

**Arguments**

- **seqs** is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- **label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
- **outFormat** (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.
- **outputFileDist** shows the path and name of the 'txt' output file.

**Value**

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. Number of columns for this feature matrix is equal to (length of the sequences)*7 and number of rows is equal to the number of sequences. If outFormat is 'txt', all binary values will be written to a the output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

**Note**

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

**References**

Examples

```r
ptmSeqsADR <- system.file("extdata/", package="ftrCOOL")
ptmSeqsVect <- as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
mat <- OPF_7bit_T2(seqs = ptmSeqsVect, outFormat = "mat")
```

**Description**

This group of functions (OPF Group) categorize amino acids in different groups based on the type. This function includes 7 amino acid properties. OPF_7bit_T2 substitutes each amino acid with a 7-dimensional vector. Each element of the vector shows if that amino acid locates in a special property category or not. '0' means that amino acid is not located in that property group and '1' means it is located. The only difference between OPF_7bit type1, type2, and type3 is in localization of amino acids in the properties groups.

**Usage**

```r
OPF_7bit_T2(seqs, label = c(), outFormat = "mat", outputFileDist = "")
```

**Arguments**

- `seqs` is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- `label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
- `outFormat` (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.
- `outputFileDist` shows the path and name of the 'txt' output file.

**Value**

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. Number of columns for this feature matrix is equal to (length of the sequences)*7 and number of rows is equal to the number of sequences. If outFormat is 'txt', all binary values will be written to a the output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.
Note
This function is provided for sequences with the same lengths. Users can use 'txt' option in out-
Format for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences
with different lengths, it returns an error. Also, when output format is 'txt', label information is not
shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if
sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

References
Wei,L., Zhou,C., Chen,H., Song,J. and Su,R. ACPred-FL: a sequence-based predictor using effec-

Examples

```
ptmSeqsADR<-system.file("extdata/",package="ftrCOOL")
ptmSeqsVect<-as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
mat<-OPF_7bit_T3(seqs = ptmSeqsVect,outFormat="mat")
```

**OPF_7bit_T3**

Overlapping property features_7bit_T3 (OPF_7bit_T3)

Description
This group of functions (OPF Group) categorize amino acids in different groups based on the type.
This function includes 7 amino acid properties. OPF_7bit_T3 substitutes each amino acid with
a 7-dimensional vector. Each element of the vector shows if that amino acid locates in a special
property category or not. '0' means that amino acid is not located in that property group and '1'
means it is located. The only difference between OPF_7bit type1, type2, and type3 is in localization
of amino acids in the properties groups.

Usage

```
OPF_7bit_T3(seqs, label = c(), outFormat = "mat", outputFileDist = "")
```

Arguments

- **seqs** is a FASTA file with amino acid sequences. Each sequence starts with a '>'
  character. Also, seqs could be a string vector. Each element of the vector is a
  peptide/protein sequence.
- **label** is an optional parameter. It is a vector whose length is equivalent to the number
  of sequences. It shows the class of each entry (i.e., sequence).
- **outFormat** (output format) can take two values: 'mat'(matrix) and 'txt'. The default value
  is 'mat'.
- **outputFileDist** shows the path and name of the 'txt' output file.
The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. Number of columns for this feature matrix is equal to (length of the sequences)*7 and number of rows is equal to the number of sequences. If outFormat is 'txt', all binary values will be written to a the output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

This function is provided for sequences with the same lengths. Users can use 'txt' option in out-Format for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

This function works like PSEkNUCdi_DNA except that the default value of selectedIdx parameter is different.

This function works like PSEkNUCdi_DNA except that the default value of selectedIdx parameter is different.
Arguments

seqs is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

selectedIdx is a vector of Ids or indices of the desired physicochemical properties of dinucleotides. Users can choose the desired indices by their ids or their names in the DI_DNA index file. Default value of this parameter is a vector with ("Base stacking","Protein induced deformability","B-DNA twist","A-philicity","Propeller twist","Duplex stability:(freeenergy)","DNA denaturation","Bending stiffness","Protein DNA twist","Aida_BA_transition","Breslauer_dG","Breslauer_dH","Electron interaction","Hartman_trans_free_energy","Helix-Coil_transition","Lisser_BZ_transition","Polar_interaction","SantaLucia_dG","SantaLucia_dS","Sarai_flexibility","Stability","Sugimoto_dG","Sugimoto_dH","Sugimoto_dS","Duplex tability(disruptenergy)","Stabilising energy of Z-DNA","Breslauer_dS","Ivanov_BA_transition","SantaLucia_dH","Stacking_energy","Watson-Crick_interaction","Dinucleotide GC Content","Rise","Roll","Shift","Slide","Tilt","Twist") entries.

lambda is a tuning parameter. This integer value shows the maximum limit of spaces between dinucleotide pairs. The Number of spaces changes from 1 to lambda.

w (weight) is a tuning parameter. It changes in the range of 0 to 1. The default value is 0.05.

l This parameter keeps the value of l in Imer composition. The Imers form the first 4^l elements of the APkNCdi descriptor.

ORF (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

reverseORF is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

threshold is a number between (0 , 1]. In selectedIdx, indices with a correlation higher than the threshold will be deleted. The default value is 1.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
Details

This function computes the pseudo nucleotide composition for each physicochemical property of di-nucleotides. We have provided users with the ability to choose among the 148 properties in the di-nucleotide index database.

Value

a feature matrix such that the number of columns is $4^l+\lambda$ and the number of rows is equal to the number of sequences.

Examples

```r
fileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
mat<-PSEkNUCdi_DNA(seqs=fileLNC,l=2,ORF=TRUE,threshold=0.8)
```

Usage

```r
PS2_DNA(
  seqs,
  binaryType = "numBin",
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

Arguments

- `seqs` is a FASTA file containing nucleotide sequences. The sequences start with '>'. Also, `seqs` could be a string vector. Each element of the vector is a nucleotide sequence.
- `binaryType` It can take any of the following values: ('strBin','logicBin','numBin'). 'strBin'(String binary): each di-nucleotide is represented by a string containing 16 characters(0-1). For example, 'AA' = "1000000000000000", 'AC' = "0100000000000000", ..., 'TT'= "0000000000000001" 'logicBin'(logical value): Each amino acid is represented by a vector containing 16 logical entries. For example, 'AA' = c(T,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F), ... 'numBin' (numeric bin): Each amino acid is represented by a numeric (i.e., integer) vector containing 16 numerals. For example, 'AA' = c(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
PS2_RNA

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

outFormat (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

outputFileDist shows the path and name of the 'txt' output file.

Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences-1)*16. If outFormat is 'txt', all binary values will be written to a the output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

References

Zhen Chen, Pei Zhao, Chen Li, Fuyi Li, Dongxu Xiang, Yong-Zi Chen, Tatsuya Akutsu, Roger J Daly, Geoffrey I Webb, Quanzhi Zhao, Lukasz Kurgan, Jiangning Song, iLearnPlus: a comprehensive and automated machine-learning platform for nucleic acid and protein sequence analysis, prediction and visualization, Nucleic Acids Research, (2021).

Examples

LNCSeqsADR<-system.file("extdata/",package="ftrCOOL")
LNC50Nuc<-as.vector(read.csv(paste0(LNCSeqsADR,"/LNC50Nuc.csv"))[,2])
mat<-PS2_DNA(seqs = LNC50Nuc,outFormat="mat")
Usage

```r
PS2_RNA(
  seqs,
  binaryType = "numBin",
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

Arguments

- **seqs**: is a FASTA file containing nucleotide sequences. The sequences start with `>`. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

- **binaryType**: It can take any of the following values: (`strBin`, `logicBin`, `numBin`). `strBin` (String binary): each di-nucleotide is represented by a string containing 16 characters (0-1). For example, `'AA' = "1000000000000000"`, `'AC' = "0100000000000000"`, `...`, `'TT' = "0000000000000001"` `logicBin` (logical value): Each amino acid is represented by a vector containing 16 logical entries. For example, `'AA' = c(T,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F)`, `...` `numBin` (numeric bin): Each amino acid is represented by a numeric (i.e., integer) vector containing 16 numerals. For example, `'AA' = c(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)

- **label**: is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

- **outFormat** (output format) can take two values: ‘mat’ (matrix) and ‘txt’. The default value is ‘mat’.

- **outputFileDist**: shows the path and name of the ‘txt’ output file.

Value

The output is different depending on the `outFormat` parameter (`'mat'` or `'txt'`). If `outFormat` is set to `mat`, it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if `binaryType` is `'strBin'`, the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences-1)*16. If `outFormat` is `'txt'`, all binary values will be written to a the output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

Note

This function is provided for sequences with the same lengths. Users can use ‘txt’ option in `outFormat` for sequences with different lengths. Warning: If `outFormat` is set to ‘mat’ for sequences with different lengths, it returns an error. Also, when output format is ‘txt’, label information is not shown in the text file. It is noteworthy that ‘txt’ format is not usable for machine learning purposes if sequences have different sizes. Otherwise ‘txt’ format is also usable for machine learning purposes.
References

Zhen Chen, Pei Zhao, Chen Li, Fuyi Li, Dongxu Xiang, Yong-Zi Chen, Tatsuya Akutsu, Roger J Daly, Geoffrey I Webb, Quanzhi Zhao, Lukasz Kurgan, Jiangning Song, iLearnPlus: a comprehensive and automated machine-learning platform for nucleic acid and protein sequence analysis, prediction and visualization, Nucleic Acids Research, (2021).

Examples

```r
fileLNC <- system.file("extdata/Carica_papaya101RNA.txt", package="ftrCOOL")
mat <- PS2_RNA(seqs = fileLNC, binaryType="numBin", outFormat="mat")
```

---

**PS3_DNA**

**Position-specific of three nucleotide_DNA (PS3_DNA)**

**Description**

This function transforms each tri-nucleotide of the sequence to a binary format. The type of the binary format is determined by the binaryType parameter. For details about each format, please refer to the description of the binaryType parameter.

**Usage**

```r
PS3_DNA(
  seqs,
  binaryType = "numBin",
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

**Arguments**

- **seqs** is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

- **binaryType** It can take any of the following values: ('strBin', 'logicBin', 'numBin'). 'strBin'(String binary): each di-nucleotide is represented by a string containing 64 characters (63 times '0' and one '1'). For example, 'AAA' = "1000000000000000...0", .... 'logicBin'(logical value): Each amino acid is represented by a vector containing 64 logical entries (63 times F and one T). For example, 'AA' = c(T,F,F,F,F,F,F,F,F,F,F,F,F,F,...,F), ... 'numBin' (numeric bin): Each amino acid is represented by a numeric (i.e., integer) vector containing 64 numerals (63 times '0' and one '1'). For example, 'AA' = c(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,...,0)

- **label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
outFormat (output format) can take two values: 'mat' (matrix) and 'txt'. The default value is 'mat'.

outputFileDist shows the path and name of the 'txt' output file.

Value
The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences-2)*64. If outFormat is 'txt', all binary values will be written to a the output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

Note
This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

References
Zhen Chen, Pei Zhao, Chen Li, Fuyi Li, Dongxu Xiang, Yong-Zi Chen, Tatsuya Akutsu, Roger J Daly, Geoffrey I Webb, Quanzhi Zhao, Lukasz Kurgan, Jiangning Song, iLearnPlus: a comprehensive and automated machine-learning platform for nucleic acid and protein sequence analysis, prediction and visualization, Nucleic Acids Research, (2021).

Examples
LNCSeqsADR<-system.file("extdata/",package="ftrCOOL")
LNC50Nuc<-as.vector(read.csv(paste0(LNCSeqsADR,"/LNC50Nuc.csv"))[,2])
mat<-PS3_DNA(seqs = LNC50Nuc,outFormat="mat")

---

PS3_RNA  

**Position-specific of three ribonucleotide RNA (PS3_RNA)**

**Description**
This function transforms each tri-ribonucleotide of the sequence to a binary format. The type of the binary format is determined by the binaryType parameter. For details about each format, please refer to the description of the binaryType parameter.
Usage

```r
PS3_RNA(
  seqs,
  binaryType = "numBin",
  label = c(),
  outFormat = "mat",
  outputFileDist ="
)
```

Arguments

- `seqs` is a FASTA file containing ribonucleotide sequences. The sequences start with '>'. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.
- `binaryType` It can take any of the following values: ('strBin', 'logicBin', 'numBin'). 'strBin' (String binary): each di-ribonucleotide is represented by a string containing 64 characters (63 times '0' and one '1'). For example, 'AAA' = "1000000000000000...0", .... 'logicBin' (logical value): Each amino acid is represented by a vector containing 64 logical entries (63 times F and one T). For example, 'AA' = c(T,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,...,F), ... 'numBin' (numeric bin): Each amino acid is represented by a numeric (i.e., integer) vector containing 64 numerals (63 times '0' and one '1'). For example, 'AA' = c(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,...,0)
- `label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
- `outFormat` (output format) can take two values: 'mat' (matrix) and 'txt'. The default value is 'mat'.
- `outputFileDist` shows the path and name of the 'txt' output file.

Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences-2)*64. If outFormat is 'txt', all binary values will be written to a the output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.
References
Zhen Chen, Pei Zhao, Chen Li, Fuyi Li, Dongxu Xiang, Yong-Zi Chen, Tatsuya Akutsu, Roger J Daly, Geoffrey I Webb, Quanzhi Zhao, Lukasz Kurgan, Jiangning Song, iLearnPlus: a comprehensive and automated machine-learning platform for nucleic acid and protein sequence analysis, prediction and visualization, Nucleic Acids Research, (2021).

Examples

```r
fileLNC<-system.file("extdata/Carica_papaya101RNA.txt",package="frtC0OOL")
mat<-PS3_RNA(seqs = fileLNC, binaryType="numBin",outFormat="mat")
```

---

**PS4_DNA**

Position-specific of four nucleotide DNA (PS4_DNA)

**Description**
This function transforms each 4-nucleotide of the sequence to a binary format. The type of the binary format is determined by the binaryType parameter. For details about each format, please refer to the description of the binaryType parameter.

**Usage**

```r
PS4_DNA(
  seqs,
  binaryType = "numBin",
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

**Arguments**

- **seqs** is a FASTA file containing nucleotide sequences. The sequences start with '>'.
  Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

- **binaryType** It can take any of the following values: ('strBin', 'logicBin', 'numBin'). 'strBin' (String binary): each di-nucleotide is represented by a string containing 256 characters (255 times '0' and one '1'). For example, 'AAA' = "1000000000000000...0", etc.
  'numBin' (numeric bin): Each amino acid is represented by a numeric (i.e., integer) vector containing 256 numerals (255 times '0' and one '1'). For example, 'AA' = c(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,...,0)
PS4_RNA

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

outFormat (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

outputFileDist shows the path and name of the 'txt' output file.

Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences-3)*256. If outFormat is 'txt', all binary values will be written to a the output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

References

Zhen Chen, Pei Zhao, Chen Li, Fuyi Li, Dongxu Xiang, Yong-Zi Chen, Tatsuya Akutsu, Roger J Daly, Geoffrey I Webb, Quanzhi Zhao, Lukasz Kurgan, Jiangning Song, iLearnPlus: a comprehensive and automated machine-learning platform for nucleic acid and protein sequence analysis, prediction and visualization, Nucleic Acids Research, (2021).

Examples

LNCSeqsADR<-system.file("extdata/",package="ftrCOOL")
LNC50Nuc<-as.vector(read.csv(paste0(LNCSeqsADR,"/LNC50Nuc.csv"))[,2])
mat<-PS4_DNA(seqs = LNC50Nuc,outFormat="mat")

PS4_RNA

Position-specific of four ribonucleotide (PS4_RNA)

Description

This function transforms each 4-ribonucleotide of the sequence to a binary format. The type of the binary format is determined by the binaryType parameter. For details about each format, please refer to the description of the binaryType parameter.
Usage

```r
PS4_RNA(
  seqs,
  binaryType = "numBin",
  label = c(),
  outFormat = "mat",
  outputFileDist = ""
)
```

Arguments

- **seqs** is a FASTA file containing ribonucleotide sequences. The sequences start with '>' . Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.

- **binaryType** It can take any of the following values: ('strBin','logicBin','numBin'). 'strBin'(String binary): each di-ribonucleotide is represented by a string containing 256 characters (255 times '0' and one '1'). For example, 'AAA' = "1000000000000000...0", .... 'logicBin'(logical value): Each amino acid is represented by a vector containing 256 logical entries (255 times F and one T). For example, 'AA' = c(T,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,...,F), ... 'numBin' (numeric bin): Each amino acid is represented by a numeric (i.e., integer) vector containing 256 numerals (255 times '0' and one '1'). For example, 'AA' = c(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,...,0)

- **label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

- **outFormat** (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.

- **outputFileDist** shows the path and name of the 'txt' output file.

Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences-3)*256. If outFormat is 'txt', all binary values will be written to a the output is written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.
### References

Zhen Chen, Pei Zhao, Chen Li, Fuyi Li, Dongxu Xiang, Yong-Zi Chen, Tatsuya Akutsu, Roger J Daly, Geoffrey I Webb, Quanzhi Zhao, Lukasz Kurgan, Jiangning Song, iLearnPlus: a comprehensive and automated machine-learning platform for nucleic acid and protein sequence analysis, prediction and visualization, Nucleic Acids Research, (2021).

### Examples

```r
fileLNC <- system.file("extdata/Carica_papaya101RNA.txt", package = "ftrCOOL")
mat <- PS4_RNA(seqs = fileLNC, binaryType = "numBin", outFormat = "mat")
```

---

### PSEAAC

**Pseudo-Amino Acid Composition (Parallel) (PSEAAC)**

#### Description

This function calculates the pseudo amino acid composition (parallel) for each sequence.

#### Usage

```r
PSEAAC(
  seqs,
  aaIDX = c("ARGP820101", "HOPT810101", "Mass"),
  lambda = 30,
  w = 0.05,
  l = 1,
  threshold = 1,
  label = c()
)
```

#### Arguments

- **seqs**: is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- **aaIDX**: is a vector of Ids or indexes of the user-selected physicochemical properties in the aaIndex2 database. The default values of the vector are the hydrophobicity ids and hydrophilicity ids and Mass of residual in the amino acid index file.
- **lambda**: is a tuning parameter. Its value indicates the maximum number of spaces between amino acid pairs. The number changes from 1 to lambda.
- **w** (weight): is a tuning parameter. It changes in from 0 to 1. The default value is 0.05.
l

This parameter keeps the value of l in lmer composition. The lmers form the first $20^l$ elements of the APAAC descriptor.

threshold

is a number between $(0, 1]$. It deletes aaIndexes which have a correlation bigger than the threshold. The default value is 1.

label

is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

A feature matrix such that the number of columns is $20^l+\lambda$ and the number of rows is equal to the number of sequences.

Examples

```
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat<-PSEAAAC(seqs=filePrs,l=2)
```

<table>
<thead>
<tr>
<th>PseEIIP</th>
<th>Pseudo Electron-Ion Interaction Pseudopotentials of Trinucleotide (PseEIIP)</th>
</tr>
</thead>
</table>

Description

This function calculates the pseudo electron-ion interaction for each sequence. It creates a feature vector for each sequence. The vector contains a value for each for each tri-nucleotide. The value is computed by multiplying the aggregate value of electron-ion interaction of each nucleotide

Usage

```
PseEIIP(seqs, label = c())
```

Arguments

- **seqs** is a FASTA file containing nucleotide sequences. The sequences start with '>'. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
- **label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

This function returns a feature matrix which the number of rows is equal to the number of sequences and the number of columns is $4^3=64$.

References

PSEkNUCdi_DNA

Examples

```r
LNCSeqsADR <- system.file("extdata/", package="ftrCOOL")
LNC50Nuc <- as.vector(read.csv(paste0(LNCSeqsADR,"/LNC50Nuc.csv"))[,2])
mat <- PseEIIP(seqs = LNC50Nuc)
```

PSEkNUCdi_DNA  
_Pseudo k Nucleotide Composition-Di(Parallel) (PSEkNUCdi_DNA)_

Description

This function calculates the pseudo-k nucleotide composition(Di) (Parallel) for each sequence.

Usage

```r
PSEkNUCdi_DNA(
  seqs,
  selectedIdx = c("Rise", "Roll", "Shift", "Slide", "Tilt", "Twist"),
  lambda = 3,
  w = 0.05,
  l = 2,
  ORF = FALSE,
  reverseORF = TRUE,
  threshold = 1,
  label = c()
)
```

Arguments

- **seqs** is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
- **selectedIdx** is a vector of Ids or indices of the desired physicochemical properties of dinucleotides. Users can choose the desired indices by their ids or their names in the DI_DNA file. The default values of the parameter is a vector with ("Rise", "Roll", "Shift", "Slide", "Tilt", "Twist") ids.
- **lambda** is a tuning parameter. This integer value shows the maximum limit of spaces between dinucleotide pairs. The Number of spaces changes from 1 to lambda.
- **w** (weight) is a tuning parameter. It changes in the range of 0 to 1. The default value is 0.05.
- **l** This parameter keeps the value of l in lmer composition. The lmers form the first $4^l$ elements of the APkNCdi descriptor.
- **ORF** (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).
reverseORF is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

threshold is a number between (0, 1]. In selectedIdx, indices with a correlation higher than the threshold will be deleted. The default value is 1.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details

This function computes the pseudo nucleotide composition for each physicochemical property of di-nucleotides. We have provided users with the ability to choose among the 148 properties in the di-nucleotide index database.

Value

A feature matrix such that the number of columns is $4^l+\lambda$ and the number of rows is equal to the number of sequences.

Examples

```r
fileLNC <- system.file("extdata/Athaliana_LNCRNA.fa", package="ftrCOOL")
mat <- PSEkNUCdi_DNA(seqs = fileLNC, l = 2, ORF = TRUE, threshold = 0.8)
```

<table>
<thead>
<tr>
<th>PSEkNUCdi_RNA</th>
<th>Pseudo k ribonucleotide Composition-Di(Parallel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSEkNUCdi_RNA</td>
<td>(PSEkNUCdi_RNA)</td>
</tr>
</tbody>
</table>

Description

This function calculates the pseudo-k ribonucleotide composition(Di) (Parallel) for each sequence.

Usage

```r
PSEkNUCdi_RNA(
    seqs,
    selectedIdx = c("Rise (RNA)", "Roll (RNA)", "Shift (RNA)", "Slide (RNA)", "Tilt (RNA)", "Twist (RNA)",
    lambda = 3,
    w = 0.05,
    l = 2,
    ORF = FALSE,
    reverseORF = TRUE,
    threshold = 1,
    label = c()
)
```
Arguments

`seqs` is a FASTA file containing ribonucleotide sequences. The sequences start with ‘>’. Also, `seqs` could be a string vector. Each element of the vector is a ribonucleotide sequence.

`selectedIdx` is a vector of Ids or indices of the desired physicochemical properties of di-ribonucleotides. Users can choose the desired indices by their ids or their names in the DI_RNA properties file. The default value of this parameter is a vector with ("Rise (RNA)", "Roll (RNA)", "Shift (RNA)", "Slide (RNA)", "Tilt (RNA)", "Twist (RNA)") ids.

`lambda` is a tuning parameter. This integer value shows the maximum limit of spaces between di-ribonucleotide pairs. The Number of spaces changes from 1 to lambda.

`w` (weight) is a tuning parameter. It changes in the range of 0 to 1. The default value is 0.5.

`l` This parameter keeps the value of l in lmer composition. The lmers form the first 4^l elements of the APkNCdi descriptor.

`ORF` (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

`reverseORF` is a logical parameter. It is enabled only if ORF is true. If `reverseORF` is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

`threshold` is a number between (0, 1]. In `selectedIdx`, indices with a correlation higher than the threshold will be deleted. The default value is 1.

`label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details

This function computes the pseudo ribonucleotide composition for each physicochemical property of di-ribonucleotides. We have provided users with the ability to choose among the 22 properties in the di-ribonucleotide index database.

Value

A feature matrix such that the number of columns is 4^l+lambda and the number of rows is equal to the number of sequences.

Examples

```r
fileLNC<-'system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
mat<-PSEkNUCdi_RNA(seqs=fileLNC,l=2,ORF=TRUE,threshold=0.8)
```
Description

This function calculates pseudo-k nucleotide composition (Tri) (Parallel) for each sequence.

Usage

\[
PSEkNUCTri_DNA(
  seqs,
  selectedIdx = c("Dnase I", "Bendability (DNase)"),
  lambda = 3,
  w = 0.05,
  l = 3,
  ORF = FALSE,
  reverseORF = TRUE,
  threshold = 1,
  label = c()
)
\]

Arguments

- **seqs**: is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

- **selectedIdx**: is a vector of Ids or indices of the desired physicochemical properties of trinucleotides. Users can choose the desired indices by their ids or their names in the TRI_DNA index file. The default value of this parameter is a vector with ("Dnase I", "Bendability (DNase)") ids.

- **lambda**: is a tuning parameter. This integer value shows the maximum limit of spaces between Tri-nucleotide pairs. The Number of spaces changes from 1 to lambda.

- **w**: (weight) is a tuning parameter. It can take a value in the range 0 to 1. The default value is 0.05.

- **l**: This parameter keeps the value of l in lmer composition. The lmers form the first 4^l elements of the APkNCTri descriptor.

- **ORF**: (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

- **reverseORF**: is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

- **threshold**: is a number between (0 , 1]. In selectedIdx, indices with a correlation higher than the threshold will be deleted. The default value is 1.

- **label**: is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
Details

This function computes the pseudo nucleotide composition for each physicochemical property of trinucleotides. We have provided users with the ability to choose among the 12 properties in the tri-nucleotide index database.

Value

a feature matrix such that the number of columns is \(4^l+\lambda\) and the number of rows is equal to the number of sequences.

Examples

```r
fileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
mat<-PSEkNUCTri_DNA(seqs=fileLNC, l=2,ORF=TRUE,threshold=0.8)
```

<table>
<thead>
<tr>
<th>PseKRAAC_T1</th>
<th>Pseudo K_tuple Reduced Amino Acid Composition Type-1 (PseKRAAC_T1)</th>
</tr>
</thead>
</table>

Description

There are 16 types of PseKRAAC function. In the functions, a (user-selected) grouping of the amino acids might be used to reduce the amino acid alphabet. Also, the functions have a type parameter. The parameter determines the protein sequence analyses which can be either gap or lambda-correlation. PseKRAAC_type1(PseKRAAC_T1) contains Grp 2 to 20.

Usage

```r
PseKRAAC_T1(
  seqs,
  type = "gap",
  Grp = 5,
  GapOrLambdaValue = 2,
  k = 2,
  label = c()
)
```

Arguments

- **seqs** is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- **type** This parameter has two valid value "lambda" and "gap". "lambda" calls lambda_model function and "gap" calls gap_model function.
- **Grp** is a numeric value. It shows the id of an amino acid group. Please find the available groups in the detail section.
GapOrLambdaValue is an integer. If type is gap, this value shows number of gaps between two k-mers. If type is lambda, the value of GapOrLambdaValue shows the number of gaps between each two amino acids of k-mers.

k This parameter keeps the value of k in k-mer.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details


Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is $(Grp)^k$.

References


Examples

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat1<-PseKRAAC_T1(seqs=filePrs,type="gap",Grp=4,GapOrLambdaValue=3,k=2)
mat2<-PseKRAAC_T1(seqs=filePrs,type="lambda",Grp=4,GapOrLambdaValue=3,k=2)
```
PseKRAAC_T10

**Pseudo K_tuple Reduced Amino Acid Composition Type-10**
**(PseKRAAC_T10)**

**Description**

There are 16 types of PseKRAAC function. In the functions, a (user-selected) grouping of the amino acids might be used to reduce the amino acid alphabet. Also, the functions have a type parameter. The parameter determines the protein sequence analyses which can be either gap or lambda-correlation. PseKRAAC_type10(PseKRAAC_T10) contains Grp 2-20.

**Usage**

```r
PseKRAAC_T10(
  seqs,
  type = "gap",
  Grp = 5,
  GapOrLambdaValue = 2,
  k = 4,
  label = c()
)
```

**Arguments**

- **seqs**
  is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

- **type**
  This parameter has two valid value "lambda" and "gap". "lambda" calls lambda_model function and "gap" calls gap_model function.

- **Grp**
  is a numeric value. It shows the id of an amino acid group. Please find the available groups in the detail section.

- **GapOrLambdaValue**
  is an integer. If type is gap, this value shows number of gaps between two k-mers. If type is lambda, the value of GapOrLambdaValue shows the number of gaps between each two amino acids of k-mers.

- **k**
  This parameter keeps the value of k in k-mer.

- **label**
  is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

**Details**

PseKRAAC_T11

Pseudo K_tuple Reduced Amino Acid Composition Type-11
(PseKRAAC_T11)

Description

There are 16 types of PseKRAAC function. In the functions, a (user-selected) grouping of the amino acids might be used to reduce the amino acid alphabet. Also, the functions have a type parameter. The parameter determines the protein sequence analyses which can be either gap or lambda-correlation. PseKRAAC_type11(PseKRAAC_T11) contains Grp 2-20.

Usage

PseKRAAC_T11(
    seqs,
    type = "gap",
    Grp = 5,
    GapOrLambdaValue = 2,
    k = 4,
    label = c()
)
Arguments

segs is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, segs could be a string vector. Each element of the vector is a peptide/protein sequence.

type This parameter has two valid value "lambda" and "gap". "lambda" calls lambda_model function and "gap" calls gap_model function.

Grp is a numeric value. It shows the id of an amino acid group. Please find the available groups in the detail section.

GapOrLambdaValue is an integer. If type is gap, this value shows number of gaps between two k-mers. If type is lambda, the value of GapOrLambdaValue shows the number of gaps between each two amino acids of k-mers.

k This parameter keeps the value of k in k-mer.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details


Value This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is (Grp)^k.

References


Examples

filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat1<-PseKRAAC_T11(seqs=filePrs,type="gap",Grp=4,GapOrLambdaValue=3,k=2)
mat2<-PseKRAAC_T11(seqs=filePrs,type="lambda",Grp=4,GapOrLambdaValue=3,k=2)

---

**PseKRAAC_T12**

*Pseudo K_tuple Reduced Amino Acid Composition Type-12 (PseKRAAC_T12)*

---

**Description**

There are 16 types of PseKRAAC function. In the functions, a (user-selected) grouping of the amino acids might be used to reduce the amino acid alphabet. Also, the functions have a type parameter. The parameter determines the protein sequence analyses which can be either gap or lambda-correlation. PseKRAAC_type12(PseKRAAC_T12) contains Grp 2-18,20.

**Usage**

```r
PseKRAAC_T12(
  seqs,
  type = "gap",
  Grp = 5,
  GapOrLambdaValue = 2,
  k = 4,
  label = c()
)
```

**Arguments**

- `seqs` is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- `type` This parameter has two valid value "lambda" and "gap". "lambda" calls lambda_model function and "gap" calls gap_model function.
- `Grp` is a numeric value. It shows the id of an amino acid group. Please find the available groups in the detail section.
- `GapOrLambdaValue` is an integer. If type is gap, this value shows number of gaps between two k-mers. If type is lambda, the value of GapOrLambdaValue shows the number of gaps between each two amino acids of k-mers.
- `k` This parameter keeps the value of k in k-mer.
- `label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
Details

Groups: 2=c(‘IVMLFWYC’, ’ARNDQEGHKPST’), 3=c(‘IVLMFWC’, ’YA’, ’RNDQEGHKPST’),
4=c(‘IVLMFW’, ’C’, ’YA’, ’RNDQEGHKPST’), 5=c(‘IVLMFW’, ’C’, ’YA’, ’G’, ’RNDQEGHKPST’),

Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and
the number of columns is (Grp)^k.

References


Examples

filePrs<-system.file("extdata/proteins.fasta",package=“ftrCOOL")

mat1<-PseKRAAC_T12(seqs=filePrs,type="gap",Grp=4,GapOrLambdaValue=3,k=2)

mat2<-PseKRAAC_T12(seqs=filePrs,type="lambda",Grp=4,GapOrLambdaValue=3,k=2)

PseKRAAC_T13

Description

There are 16 types of PseKRAAC function. In the functions, a (user-selected) grouping of the
amino acids might be used to reduce the amino acid alphabet. Also, the functions have a type
parameter. The parameter determines the protein sequence analyses which can be either gap or
lambda-correlation. PseKRAAC_type13(PseKRAAC_T13) contains Grp 4,12,17,20.
Usage

PseKRAAC_T13(
    seqs,
    type = "gap",
    Grp = 4,
    GapOrLambdaValue = 2,
    k = 4,
    label = c()
)

Arguments

seqs is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

type This parameter has two valid value "lambda" and "gap". "lambda" calls lambda_model function and "gap" calls gap_model function.

Grp is a numeric value. It shows the id of an amino acid group. Please find the available groups in the detail section.

GapOrLambdaValue is an integer. If type is gap, this value shows number of gaps between two k-mers. If type is lambda, the value of GapOrLambdaValue shows the number of gaps between each two amino acids of k-mers.

k This parameter keeps the value of k in k-mer.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details


Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is (Grp)^k.

References

Examples

```r
counter<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat1<-PseKRAAC_T13(seqs=counter,type="gap",Grp=17,GapOrLambdaValue=3,k=2)
mat2<-PseKRAAC_T13(seqs=counter,type="lambda",Grp=17,GapOrLambdaValue=3,k=2)
```

### PseKRAAC_T14

#### Description

There are 16 types of PseKRAAC function. In the functions, a (user-selected) grouping of the amino acids might be used to reduce the amino acid alphabet. Also, the functions have a type parameter. The parameter determines the protein sequence analyses which can be either gap or lambda-correlation. PseKRAAC_type14(PseKRAAC_T14) contains Grp 2-20.

#### Usage

```r
PseKRAAC_T14(
  seqs,
  type = "gap",
  Grp = 2,
  GapOrLambdaValue = 2,
  k = 4,
  label = c()
)
```

#### Arguments

- **seqs** is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- **type** This parameter has two valid value "lambda" and "gap". "lambda" calls lambda_model function and "gap" calls gap_model function.
- **Grp** is a numeric value. It shows the id of an amino acid group. Please find the available groups in the detail section.
- **GapOrLambdaValue** is an integer. If type is gap, this value shows number of gaps between two k-mers. If type is lambda, the value of GapOrLambdaValue shows the number of gaps between each two amino acids of k-mers.
- **k** This parameter keeps the value of k in k-mer.
- **label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
Details


Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is (Grp)^k.

References


Examples

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat1<-PseKRAAC_T14(seqs=filePrs,type="gap",Grp=4,GapOrLambdaValue=3,k=2)
mat2<-PseKRAAC_T14(seqs=filePrs,type="lambda",Grp=4,GapOrLambdaValue=3,k=2)
```

### Description

There are 16 types of PseKRAAC function. In the functions, a (user-selected) grouping of the amino acids might be used to reduce the amino acid alphabet. Also, the functions have a type parameter. The parameter determines the protein sequence analyses which can be either gap or lambda-correlation. PseKRAAC_type15(PseKRAAC_T15) contains Grp 2-16,20.
Usage

PseKRAAC_T15(
    seqs,
    type = "gap",
    Grp = 2,
    GapOrLambdaValue = 2,
    k = 4,
    label = c()
)

Arguments

seqs is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

type This parameter has two valid value "lambda" and "gap". "lambda" calls lambda_model function and "gap" calls gap_model function.

Grp is a numeric value. It shows the id of an amino acid group. Please find the available groups in the detail section.

GapOrLambdaValue is an integer. If type is gap, this value shows number of gaps between two k-mers. If type is lambda, the value of GapOrLambdaValue shows the number of gaps between each two amino acids of k-mers.

k This parameter keeps the value of k in k-mer.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details

Groups:

Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is (Grp)^k.
References


Examples

```
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat1<-PseKRAAC_T15(seqs=filePrs,type="gap",Grp=4,GapOrLambdaValue=3,k=2)
mat2<-PseKRAAC_T15(seqs=filePrs,type="lambda",Grp=4,GapOrLambdaValue=3,k=2)
```

<table>
<thead>
<tr>
<th>PseKRAAC_T16</th>
<th>Pseudo K_tuple Reduced Amino Acid Composition Type-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>(PseKRAAC_T16)</td>
<td></td>
</tr>
</tbody>
</table>

Description

There are 16 types of PseKRAAC function. In the functions, a (user-selected) grouping of the amino acids might be used to reduce the amino acid alphabet. Also, the functions have a type parameter. The parameter determines the protein sequence analyses which can be either gap or lambda-correlation. PseKRAAC_type16(PseKRAAC_T16) contains Grp 2-16,20.

Usage

```
PseKRAAC_T16(
    seqs,
    type = "gap",
    Grp = 2,
    GapOrLambdaValue = 2,
    k = 4,
    label = c()
)
```

Arguments

- **seqs**
  is a FASTA file with amino acid sequences. Each sequence starts with a ‘>’ character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

- **type**
  This parameter has two valid value "lambda" and "gap". "lambda" calls lambda_model function and "gap" calls gap_model function.

- **Grp**
  is a numeric value. It shows the id of an amino acid group. Please find the available groups in the detail section.
**GapOrLambdaValue**

is an integer. If type is gap, this value shows number of gaps between two k-mers. If type is lambda, the value of GapOrLambdaValue shows the number of gaps between each two amino acids of k-mers.

**k**

This parameter keeps the value of k in k-mer.

**label**

is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

---

**Details**

Groups:

- \(2 = c('IMVLFWY', 'G', 'PCAST', 'NHQEDRK')\)
- \(3 = c('IMVLFWY', 'GPCAST', 'NHQEDRK')\)
- \(4 = c('IMVLFWY', 'G', 'PCAST', 'NHQEDRK')\)
- \(5 = c('IMVL', 'FWY', 'G', 'PCAST', 'NHQEDRK')\)
- \(6 = c('IMVL', 'FWY', 'G', 'P', 'CAST', 'NHQEDRK')\)
- \(7 = c('IMVL', 'FWY', 'G', 'P', 'CAST', 'NHQED', 'RK')\)
- \(8 = c('IMVL', 'L', 'FWY', 'G', 'P', 'CAST', 'NHQED', 'RK')\)
- \(9 = c('IMVL', 'L', 'FWY', 'G', 'P', 'CAST', 'NHQED', 'RK')\)
- \(10 = c('IMVL', 'L', 'FWY', 'G', 'P', 'C', 'A', 'STNH', 'RKQ', 'D')\)
- \(12 = c('IMVL', 'L', 'FWY', 'G', 'P', 'C', 'A', 'STNH', 'RKQ', 'E', 'D')\)
- \(13 = c('IMVL', 'L', 'FWY', 'G', 'P', 'C', 'A', 'STNH', 'RKQ', 'E', 'D')\)
- \(14 = c('IMVL', 'L', 'FWY', 'G', 'P', 'C', 'A', 'STNH', 'RKQ', 'E', 'D')\)
- \(15 = c('IMVL', 'L', 'FWY', 'G', 'P', 'C', 'A', 'STNH', 'RKQ', 'E', 'D')\)
- \(16 = c('IMVL', 'L', 'FWY', 'G', 'P', 'C', 'A', 'STNH', 'RKQ', 'E', 'D')\)

**Value**

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is \((Grp)^k\).

**References**


**Examples**

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")

mat1<-PseKRAAC_T16(seqs=filePrs,type="gap",Grp=4,GapOrLambdaValue=3,k=2)
mat2<-PseKRAAC_T16(seqs=filePrs,type="lambda",Grp=4,GapOrLambdaValue=3,k=2)
```
PseKRAAC_T2 Pseudo K_tuple Reduced Amino Acid Composition Type-2 (PseKRAAC_T2)

Description

There are 16 types of PseKRAAC function. In the functions, a (user-selected) grouping of the
amino acids might be used to reduce the amino acid alphabet. Also, the functions have a type
parameter. The parameter determines the protein sequence analyses which can be either gap or
lambda-correlation. PseKRAAC_type2(PseKRAAC_T2) contains Grp 2-6,8,15,20.

Usage

PseKRAAC_T2(
  seqs,
  type = "gap",
  Grp = 2,
  GapOrLambdaValue = 2,
  k = 4,
  label = c()
)

Arguments

seqs is a FASTA file with amino acid sequences. Each sequence starts with a ’>’
character. Also, seqs could be a string vector. Each element of the vector is a
peptide/protein sequence.
type This parameter has two valid value "lambda" and "gap". "lambda" calls lambda_model
function and "gap" calls gap_model function.
Grp is a numeric value. It shows the id of an amino acid group. Please find the
available groups in the detail section.
GapOrLambdaValue is an integer. If type is gap, this value shows number of gaps between two k-
mers. If type is lambda, the value of GapOrLambdaValue shows the number of
gaps between each two amino acids of k-mers.
k This parameter keeps the value of k in k-mer.
label is an optional parameter. It is a vector whose length is equivalent to the number
of sequences. It shows the class of each entry (i.e., sequence).

Details

Groups:
2=c("LVIMCAGSTPFYW", "EDNQKRH"), 3=c("LVIMCAGSTP", "FYW", "EDNQKRH"), 4=c("LVIMC",
AGSTP", "FYW", "EDNQKRH"), 5=c("LVIMC", "AGSTP", "FYW", "EDNQ", "KRH"), 6=c("LVIM",
Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is $(\text{Grp})^k$.

References


Examples

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat1<-PseKRAAC_T2(seqs=filePrs,type="gap",Grp=4,GapOrLambdaValue=3,k=2)
mat2<-PseKRAAC_T2(seqs=filePrs,type="lambda",Grp=4,GapOrLambdaValue=3,k=2)
```

PseKRAAC_T3A

**Pseudo K_tuple Reduced Amino Acid Composition Type-3A**

(PseKRAAC_T3A)

Description

There are 16 types of PseKRAAC function. In the functions, a (user-selected) grouping of the amino acids might be used to reduce the amino acid alphabet. Also, the functions have a type parameter. The parameter determines the protein sequence analyses which can be either gap or lambda-correlation. PseKRAAC_type3 contain two type: type3A and type3B. `PseKRAAC_T3A` contains Grp 2-20.

Usage

```r
PseKRAAC_T3A(
  seqs,
  type = "gap",
  Grp = 2,
  GapOrLambdaValue = 2,
  k = 4,
  label = c()
)
```

Arguments

- **seqs**
  - is a FASTA file with amino acid sequences. Each sequence starts with a `>` character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

- **type**
  - This parameter has two valid value "lambda" and "gap". "lambda" calls lambda_model function and "gap" calls gap_model function.
Grp is a numeric value. It shows the id of an amino acid group. Please find the available groups in the detail section.

GapOrLambdaValue is an integer. If type is gap, this value shows number of gaps between two k-mers. If type is lambda, the value of GapOrLambdaValue shows the number of gaps between each two amino acids of k-mers.

k This parameter keeps the value of k in k-mer.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

### Details

Groups:

<table>
<thead>
<tr>
<th>Group</th>
<th>Amino Acid Sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grp2</td>
<td>c('AGSPDEQNHTKRMILFYV', 'W')</td>
</tr>
<tr>
<td>Grp3</td>
<td>c('AGSPDEQNHTKRMILFYV', 'W', 'C')</td>
</tr>
<tr>
<td>Grp4</td>
<td>c('AGSPDEQNHTKRMILFYV', 'W', 'YFL', 'C')</td>
</tr>
<tr>
<td>Grp5</td>
<td>c('AGSPDEQNHTKRMILFYV', 'W', 'YF', 'MIVL', 'C')</td>
</tr>
<tr>
<td>Grp6</td>
<td>c('AGSP', 'DEQNHTKRMILFYV')</td>
</tr>
<tr>
<td>Grp7</td>
<td>c('AGSP', 'DEQNH', 'TKRMILFYV', 'W', 'YF', 'L', 'C')</td>
</tr>
<tr>
<td>Grp8</td>
<td>c('AG', 'DEQNH', 'TKRMILFYV', 'HY', 'W', 'L', 'FP', 'CS')</td>
</tr>
<tr>
<td>Grp9</td>
<td>c('AG', 'P', 'DEQNH', 'TKRMILFYV', 'HY', 'W', 'L', 'FP', 'CS')</td>
</tr>
<tr>
<td>Grp10</td>
<td>c('AG', 'P', 'DEQNH', 'TKRMILFYV', 'HY', 'W', 'L', 'FP', 'CS')</td>
</tr>
<tr>
<td>Grp11</td>
<td>c('AG', 'P', 'DEQNH', 'TKRMILFYV', 'HY', 'W', 'L', 'FP', 'CS')</td>
</tr>
<tr>
<td>Grp12</td>
<td>c('AG', 'P', 'DEQNH', 'TKRMILFYV', 'HY', 'W', 'L', 'FP', 'CS')</td>
</tr>
<tr>
<td>Grp13</td>
<td>c('AG', 'P', 'DEQNH', 'TKRMILFYV', 'HY', 'W', 'L', 'FP', 'CS')</td>
</tr>
<tr>
<td>Grp14</td>
<td>c('AG', 'P', 'DEQNH', 'TKRMILFYV', 'HY', 'W', 'L', 'FP', 'CS')</td>
</tr>
<tr>
<td>Grp15</td>
<td>c('AG', 'P', 'DEQNH', 'TKRMILFYV', 'HY', 'W', 'L', 'FP', 'CS')</td>
</tr>
<tr>
<td>Grp16</td>
<td>c('AG', 'P', 'DEQNH', 'TKRMILFYV', 'HY', 'W', 'L', 'FP', 'CS')</td>
</tr>
<tr>
<td>Grp17</td>
<td>c('AG', 'P', 'DEQNH', 'TKRMILFYV', 'HY', 'W', 'L', 'FP', 'CS')</td>
</tr>
<tr>
<td>Grp18</td>
<td>c('AG', 'P', 'DEQNH', 'TKRMILFYV', 'HY', 'W', 'L', 'FP', 'CS')</td>
</tr>
<tr>
<td>Grp19</td>
<td>c('AG', 'P', 'DEQNH', 'TKRMILFYV', 'HY', 'W', 'L', 'FP', 'CS')</td>
</tr>
<tr>
<td>Grp20</td>
<td>c('AG', 'P', 'DEQNH', 'TKRMILFYV', 'HY', 'W', 'L', 'FP', 'CS')</td>
</tr>
</tbody>
</table>

Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is (Grp)^k.

### References


### Examples

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat1<-PseKRAAC_T3A(seqs=filePrs,type="gap",Grp=4,GapOrLambdaValue=3,k=2)
mat2<-PseKRAAC_T3A(seqs=filePrs,type="lambda",Grp=4,GapOrLambdaValue=3,k=2)
```
PseKRAAC_T3B

**Description**

There are 16 types of PseKRAAC function. In the functions, a (user-selected) grouping of the amino acids might be used to reduce the amino acid alphabet. Also, the functions have a type parameter. The parameter determines the protein sequence analyses which can be either gap or lambda-correlation. PseKRAAC_type3 contain two type: type3A and type3B. 'PseKRAAC_T3B' contains Grp 2-20.

**Usage**

```r
PseKRAAC_T3B(
  seqs,
  type = "gap",
  Grp = 2,
  GapOrLambdaValue = 2,
  k = 4,
  label = c()
)
```

**Arguments**

- **seqs**
  - is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

- **type**
  - This parameter has two valid value "lambda" and "gap". "lambda" calls lambda_model function and "gap" calls gap_model function.

- **Grp**
  - is a numeric value. It shows the id of an amino acid group. Please find the available groups in the detail section.

- **GapOrLambdaValue**
  - is an integer. If type is gap, this value shows number of gaps between two k-mers. If type is lambda, the value of GapOrLambdaValue shows the number of gaps between each two amino acids of k-mers.

- **k**
  - This parameter keeps the value of k in k-mer.

- **label**
  - is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

**Details**

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is \((Grp)^{k}\).

**References**


**Examples**

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")

mat1<-PseKRAAC_T3B(seqs=filePrs,type="gap",Grp=4,GapOrLambdaValue=3,k=2)

mat2<-PseKRAAC_T3B(seqs=filePrs,type="lambda",Grp=4,GapOrLambdaValue=3,k=2)
```

---

### PseKRAAC_T4

#### Pseudo K_tuple Reduced Amino Acid Composition Type-4

(PseKRAAC_T4)

**Description**

There are 16 types of PseKRAAC function. In the functions, a (user-selected) grouping of the amino acids might be used to reduce the amino acid alphabet. Also, the functions have a type parameter. The parameter determines the protein sequence analyses which can be either gap or lambda-correlation. PseKRAAC_type4(PseKRAAC_T4) contains Grp 5,8,9,11,13,20.

**Usage**

```r
PseKRAAC_T4(  
  seqs,  
  type = "gap",  
  Grp = 5,  
  GapOrLambdaValue = 2,
)```
Arguments

seqs is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

type This parameter has two valid value "lambda" and "gap". "lambda" calls lambda_model function and "gap" calls gap_model function.

Grp is a numeric value. It shows the id of an amino acid group. Please find the available groups in the detail section.

GapOrLambdaValue is an integer. If type is gap, this value shows number of gaps between two k-mers. If type is lambda, the value of GapOrLambdaValue shows the number of gaps between each two amino acids of k-mers.

k This parameter keeps the value of k in k-mer.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details


Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is $(\text{Grp})^k$.

References


Examples

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")

mat1<-PseKRAAC_T4(seqs=filePrs,type="gap",Grp=8,GapOrLambdaValue=3,k=2)

mat2<-PseKRAAC_T4(seqs=filePrs,type="lambda",Grp=8,GapOrLambdaValue=3,k=2)
```
Pseudo K_tuple Reduced Amino Acid Composition Type-5 (PseKRAAC_T5)

Description

There are 16 types of PseKRAAC function. In the functions, a (user-selected) grouping of the amino acids might be used to reduce the amino acid alphabet. Also, the functions have a type parameter. The parameter determines the protein sequence analyses which can be either gap or lambda-correlation. PseKRAAC_type5(PseKRAAC_T5) contains Grp 3, 4, 8, 10, 15, 20.

Usage

PseKRAAC_T5(
  seqs,
  type = "gap",
  Grp = 4,
  GapOrLambdaValue = 2,
  k = 4,
  label = c()
)

Arguments

seqs is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

type This parameter has two valid value "lambda" and "gap". "lambda" calls lambda_model function and "gap" calls gap_model function.

Grp is a numeric value. It shows the id of an amino acid group. Please find the available groups in the detail section.

GapOrLambdaValue is an integer. If type is gap, this value shows number of gaps between two k-mers. If type is lambda, the value of GapOrLambdaValue shows the number of gaps between each two amino acids of k-mers.

k This parameter keeps the value of k in k-mer.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details

Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is $(\text{Grp})^k$.

References


Examples

```r
filePrs <- system.file("extdata/proteins.fasta", package="ftrCOOL")
mat1 <- PseKRAAC_T5(seqs = filePrs, type = "gap", Grp = 4, GapOrLambdaValue = 3, k = 2)
mat2 <- PseKRAAC_T5(seqs = filePrs, type = "lambda", Grp = 4, GapOrLambdaValue = 3, k = 2)
```

Usage

```r
PseKRAAC_T6A(
  seqs,
  type = "gap",
  Grp = 5,
  GapOrLambdaValue = 2,
  k = 4,
  label = c()
)
```

Arguments

- **seqs** is a FASTA file with amino acid sequences. Each sequence starts with a ‘>’ character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- **type** This parameter has two valid value "lambda" and "gap". "lambda" calls lambda_model function and "gap" calls gap_model function.
Grp is a numeric value. It shows the id of an amino acid group. Please find the available groups in the detail section.

GapOrLambdaValue is an integer. If type is gap, this value shows number of gaps between two k-mers. If type is lambda, the value of GapOrLambdaValue shows the number of gaps between each two amino acids of k-mers.

k This parameter keeps the value of k in k-mer.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details


Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is (Grp)^k.

References


Examples

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat1<-PseKRAAC_T6A(seqs=filePrs,type="gap",Grp=4,GapOrLambdaValue=3,k=2)
mat2<-PseKRAAC_T6A(seqs=filePrs,type="lambda",Grp=4,GapOrLambdaValue=3,k=2)
```

<table>
<thead>
<tr>
<th>PseKRAAC_T6B</th>
<th>Pseudo K-tuple Reduced Amino Acid Composition Type-6B (PseKRAAC_T6B)</th>
</tr>
</thead>
</table>

Description

There are 16 types of PseKRAAC function. In the functions, a (user-selected) grouping of the amino acids might be used to reduce the amino acid alphabet. Also, the functions have a type parameter. The parameter determines the protein sequence analyses which can be either gap or lambda-correlation. PseKRAAC_type6 contain two type: type6A and type6B. 'PseKRAAC_T6B' contains Grp 5.
Usage

PseKRAAC_T6B(
  seqs,
  type = "gap",
  Grp = 5,
  GapOrLambdaValue = 2,
  k = 4,
  label = c()
)

Arguments

seqs is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

type This parameter has two valid value "lambda" and "gap". "lambda" calls lambda_model function and "gap" calls gap_model function.

Grp is a numeric value. It shows the id of an amino acid group. Please find the available groups in the detail section.

GapOrLambdaValue is an integer. If type is gap, this value shows number of gaps between two k-mers. If type is lambda, the value of GapOrLambdaValue shows the number of gaps between each two amino acids of k-mers.

k This parameter keeps the value of k in k-mer.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details

Groups: 5=c('AEHKQRST', 'CFILMVWY', 'DN', 'G', 'P')

Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is (Grp)^k.

References


Examples

filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat1<-PseKRAAC_T6B(seqs=filePrs,type="gap",Grp=5,GapOrLambdaValue=3,k=2)
mat2<-PseKRAAC_T6B(seqs=filePrs,type="lambda",Grp=5,GapOrLambdaValue=3,k=2)
Description

There are 16 types of PseKRAAC function. In the functions, a (user-selected) grouping of the amino acids might be used to reduce the amino acid alphabet. Also, the functions have a type parameter. The parameter determines the protein sequence analyses which can be either gap or lambda-correlation. PseKRAAC_type7(PseKRAAC_T7) contains Grp 2-20.

Usage

```r
PseKRAAC_T7(
  seqs,
  type = "gap",
  Grp = 5,
  GapOrLambdaValue = 2,
  k = 4,
  label = c()
)
```

Arguments

- **seqs**: is a FASTA file with amino acid sequences. Each sequence starts with a ‘>’ character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- **type**: This parameter has two valid value "lambda" and "gap". "lambda" calls lambda_model function and "gap" calls gap_model function.
- **Grp**: is a numeric value. It shows the id of an amino acid group. Please find the available groups in the detail section.
- **GapOrLambdaValue**: is an integer. If type is gap, this value shows number of gaps between two k-mers. If type is lambda, the value of GapOrLambdaValue shows the number of gaps between each two amino acids of k-mers.
- **k**: This parameter keeps the value of k in k-mer.
- **label**: is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Details


Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is (Grp)^k.

References


Examples

```r
filePrs<-system.file("extdata/proteins.fasta", package="ftrCOOL")
mat1<-PseKRAAC_T7(seqs=filePrs, type="gap", Grp=4, GapOrLambdaValue=3, k=2)
mat2<-PseKRAAC_T7(seqs=filePrs, type="lambda", Grp=4, GapOrLambdaValue=3, k=2)
```

---

**PseKRAAC_T8**

**Pseudo K_tuple Reduced Amino Acid Composition Type-8**

*(PseKRAAC_T8)*

**Description**

There are 16 types of PseKRAAC function. In the functions, a (user-selected) grouping of the amino acids might be used to reduce the amino acid alphabet. Also, the functions have a type parameter. The parameter determines the protein sequence analyses which can be either gap or lambda-correlation. PseKRAAC_type8(PseKRAAC_T8) contains Grp 2-20.

**Usage**

```r
PseKRAAC_T8(
  seqs,
  type = "gap",
  Grp = 5,
  GapOrLambdaValue = 2,
)```
\[ k = 4, \\ label = c() \]

### Arguments

**seqs** is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

**type** This parameter has two valid value "lambda" and "gap". "lambda" calls lambda_model function and "gap" calls gap_model function.

**Grp** is a numeric value. It shows the id of an amino acid group. Please find the available groups in the detail section.

**GapOrLambdaValue** is an integer. If type is gap, this value shows number of gaps between two k-mers. If type is lambda, the value of GapOrLambdaValue shows the number of gaps between each two amino acids of k-mers.

**k** This parameter keeps the value of k in k-mer.

**label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

### Details

Groups:
- Grp2=c('ADEGKNPQRST', 'CFHILMVWY')
- Grp3=c('ADEGNPST', 'CHKQRW', 'FILMvy')
- Grp4=c('AGNPST', 'CHWY', 'DEKQR', 'FILMV'), Grp5=c('AGPST', 'CFWY', 'DEN', 'HKQR', 'ILMV'), Grp6=c('APST', 'CW', 'DEGN', 'FHY', 'ILMV', 'KQR'), Grp7=c('AGST', 'CW', 'DEN', 'FY', 'HP', 'ILMV', 'KQR'), Grp8=c('AST', 'CG', 'DEN', 'FY', 'HP', 'ILV', 'KQR', 'MW'),

### Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is \((Grp)^k\).

### References

Examples

```r
filePrs <- system.file("extdata/proteins.fasta", package="ftrCOOL")
mat1 <- PseKRAAC_T8(seqs=filePrs, type="gap", Grp=4, GapOrLambdaValue=3, k=2)
mat2 <- PseKRAAC_T8(seqs=filePrs, type="lambda", Grp=4, GapOrLambdaValue=3, k=2)
```

### Description

There are 16 types of PseKRAAC function. In the functions, a (user-selected) grouping of the amino acids might be used to reduce the amino acid alphabet. Also, the functions have a type parameter. The parameter determines the protein sequence analyses which can be either gap or lambda-correlation. `PseKRAAC_type9(PseKRAAC_T9)` contains Grp 2-20.

### Usage

```r
PseKRAAC_T9(
  seqs,
  type = "gap",
  Grp = 5,
  GapOrLambdaValue = 2,
  k = 4,
  label = c()
)
```

### Arguments

- **seqs**: is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- **type**: This parameter has two valid value "lambda" and "gap". "lambda" calls lambda_model function and "gap" calls gap_model function.
- **Grp**: is a numeric value. It shows the id of an amino acid group. Please find the available groups in the detail section.
- **GapOrLambdaValue**: is an integer. If type is gap, this value shows number of gaps between two k-mers. If type is lambda, the value of GapOrLambdaValue shows the number of gaps between each two amino acids of k-mers.
- **k**: This parameter keeps the value of k in k-mer.
- **label**: is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
Details


Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is (Grp)^k.

References


Examples

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat1<-PseKRAAC_T9(seqs=filePrs,type="gap",Grp=4,GapOrLambdaValue=3,k=2)
mat2<-PseKRAAC_T9(seqs=filePrs,type="lambda",Grp=4,GapOrLambdaValue=3,k=2)
```

PSSM

Position-Specific Scoring Matrix (PSSM)

Description

This function receives as input PSSM matrices (which are created by PSI-BLAST software) and converts them into feature vectors.

Usage

```r
PSSM(dirPath, outFormat = "mat", outputFileDist = "")
```
Arguments

FormsModule  Path of the directory which contains all output files of PSI-BLAST. Each file belongs to a sequence.

outFormat  It can take two values: 'mat' (which stands for matrix) and 'txt'. The default value is 'mat'.

outputFileDist  It shows the path and name of the 'txt' output file.

Value

The output depends on the outFormat parameter which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same length such that the number of columns is (sequence length)*(20) and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

```
dir = tempdir()
ad<-paste0(dir,"/pssm.txt")
PSSMdir<-system.file("testForder",package="ftrCOOL")
PSSMdir<-paste0(PSSMdir,"/PSSMdir/")
mat<-PSSM(PSSMdir,outFormat="txt",outputFileDist=ad)

unlink("dir", recursive = TRUE)
```

PSTNPds  Position-Specific Trinucleotide Propensity based on double-strand (PSTNPds)

Description

This function works like PSTNPss_DNA except that it considers T as A and G as C. So it converts Ts in the sequence to A and Gs to C. Then, it works with 2 alphabets A and C. For more details refer to PSTNPss_DNA.

Usage

```
PSTNPds(seqs, pos, neg, label = c())
```
Arguments

seqs is a FASTA file containing nucleotide sequences. The sequences start with '>'. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

pos is a FASTA file containing nucleotide sequences. Each sequence starts with '>'. Also, the value of this parameter can be a string vector. The sequences are positive sequences in the training model.

neg is a FASTA file containing nucleotide sequences. Each sequence starts with '>'. Also, the value of this parameter can be a string vector. The sequences are negative sequences in the training model.

label is an optional parameter. It is a vector whose length is equal to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

It returns a feature matrix. The number of columns is equal to the length of sequences minus two and the number of rows is equal to the number of sequences.

Note

The length of the sequences in positive and negative data sets and the input sets should be equal.

References


Examples

```r
ptmSeqsADR<-system.file("extdata/",package="ftrCOOL")
posSeqs<-fa.read(file=paste0(ptmSeqsADR,"/posData.txt"),alphabet="dna")
negSeqs<-fa.read(file=paste0(ptmSeqsADR,"/negData.txt"),alphabet="dna")
seqs<-fa.read(file=paste0(ptmSeqsADR,"/testData.txt"),alphabet="dna")
PSTNPds(seqs=seqs,pos=posSeqs[1],neg=negSeqs[1])
```
### Description

The inputs to this function are positive and negative data sets and a set of sequences. The output of the function is a matrix of feature vectors. The number of rows of the output matrix is equal to the number of sequences. The feature vector for an input sequence with length L is \([u(1), u(2), \ldots, u(L-2)]\). For each input sequence, \(u(1)\) is calculated by subtracting the frequency of sequences (which start with the same trinucleotides as the input sequence) in the positive set with those starting with the same trinucleotide in the negative set. We compute \(u(i)\) like \(u(1)\) with the exception that instead of the first trinucleotide, the \(i\)th trinucleotide is considered.

### Usage

```r
PSTNPss_DNA(seqs, pos, neg, label = c())
```

### Arguments

- **seqs**: is a FASTA file containing nucleotide sequences. The sequences start with ‘\>'. Also, `seqs` could be a string vector. Each element of the vector is a nucleotide sequence.
- **pos**: is a fasta file containing nucleotide sequences. Each sequence starts with ‘\>'. Also, the value of this parameter can be a string vector. The sequences are positive sequences in the training model.
- **neg**: is a fasta file containing nucleotide sequences. Each sequence starts with ‘\>'. Also, the value of this parameter can be a string vector.
- **label**: is an optional parameter. It is a vector whose length is equal to the number of sequences. It shows the class of each entry (i.e., sequence).

### Value

It returns a feature matrix. The number of columns is equal to the length of sequences minus two and the number of rows is equal to the number of sequences.

### Note

The length of the sequences in positive and negative data sets and the input sets should be equal.

### Examples

```r
ptmSeqsADR<-system.file("extdata/",package="ftrCOOL")
posSeqs<-fa.read(file=paste0(ptmSeqsADR,"/posDNA.txt"),alphabet="dna")
```
negSeqs<-fa.read(file=paste0(ptmSeqsADR,"/negDNA.txt"),alphabet="dna")
seqs<-fa.read(file=paste0(ptmSeqsADR,"/DNA_testing.txt"),alphabet="dna")

mat=PSTNPss_DNA(seqs=seqs,pos=posSeqs,neg=negSeqs)

---

**PSTNPss_RNA**

*Position-Specific Tri-ribonucleotide Propensity based on single-strand RNA (PSTNPss_RNA)*

**Description**

The inputs to this function are positive and negative data sets and a set of sequences. The output of the function is a matrix of feature vectors. The number of rows of the output matrix is equal to the number of sequences. The feature vector for an input sequence with length L is \([u(1),u(2),...,u(L-2)]\). For each input sequence, \(u(1)\) is calculated by subtracting the frequency of sequences (which start with the same tri-ribonucleotides as the input sequence) in the positive set with those starting with the same tri-ribonucleotide in the negative set. We compute \(u(i)\) like \(u(1)\) with the exception that instead of the first tri-ribonucleotide, the ith tri-ribonucleotide is considered.

**Usage**

PSTNPss_RNA(seqs, pos, neg, label = c())

**Arguments**

- **seqs** is a FASTA file containing ribonucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.
- **pos** is a fasta file containing ribonucleotide sequences. Each sequence starts with ‘>’. Also, the value of this parameter can be a string vector. The sequences are positive sequences in the training model.
- **neg** is a fasta file containing ribonucleotide sequences. Each sequence starts with ‘>’. Also, the value of this parameter can be a string vector.
- **label** is an optional parameter. It is a vector whose length is equal to the number of sequences. It shows the class of each entry (i.e., sequence).

**Value**

It returns a feature matrix. The number of columns is equal to the length of sequences minus two and the number of rows is equal to the number of sequences.

**Note**

The length of the sequences in positive and negative data sets and the input sets should be equal.
Examples

```r
ptmSeqsADR <- system.file("extdata/", package="ftrCOOL")

posSeqs <- fa.read(file=paste0(ptmSeqsADR,"/pos2RNA.txt"), alphabet="rna")
negSeqs <- fa.read(file=paste0(ptmSeqsADR,"/neg2RNA.txt"), alphabet="rna")
seqs <- fa.read(file=paste0(ptmSeqsADR,"/testSeq2RNA.txt"), alphabet="rna")

PSTNPss_RNA(seqs=seqs, pos=posSeqs, neg=negSeqs)
```
Value

It returns a feature matrix which the number of rows equals to the number of sequences and the number of columns is \((nlag*2)\). For each distance \(d\), there are two values. One value for Granthman and another one for Schneider distance.

Note

For \(d\) between 21 to \(nlag\), the function calculates tau values for \((d-20)\) to \((nlag-20)\).

Examples

```r
filePrs<-system.file("extdata/proteins.fasta",package="ftrCOOL")
mat<-QSOrder(seqs=filePrs,nlag=25)
```

---

**readASAdir**

*Read Directory of Accessible Solvent accessibility predicted files (readASAdir)*

Description

This function reads a directory that contains the output files of SPINE-X. It gets the directory path as the input and returns a list of vectors. Each vector includes the ASA predicted value for amino acids of the sequence.

Usage

```r
readASAdir(dirPath)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dirPath</td>
<td>path of the directory which contains all the output files of SPINE-X. Each file belongs to a sequence.</td>
</tr>
</tbody>
</table>

Value

A list of vectors with all the predicted ASA value for each amino acid. The length of the list is the number of files(sequences) and the length of each vector is (length of sequence(i))

Examples

```r
PredASAdir<-system.file("testForder",package="ftrCOOL")
PredASAdir<-paste0(PredASAdir,"/ASAdir/")
PredVectASA<-readASAdir(PredASAdir)
```
**readDisDir**

*Read disorder predicted Directory (readDisDir)*

**Description**

This function reads a directory that contains the output VSL2 files. It gets the directory path as the input and returns a list of vectors. Each vector includes the disorder/order type for the amino acids of the sequence.

**Usage**

```r
readDisDir(dirPath)
```

**Arguments**

- `dirPath`  
  the path of a directory which contains all the VSL2 output files.

**Value**

a list of vectors with all the predicted disorder/order type for each amino acid. The length of the list is equal to the number of files(sequences) and the length of each vector is the length of the sequence(i).

**Examples**

```r
PredDisdir<-system.file("testForder",package="ftrCOOL")
PredDisdir<-paste0(PredDisdir,"/Disdir/")
listPredVect<-readDisDir(PredDisdir)
```

---

**readPSSMdir**

*Read PSSM Directory (readPSSMdir)*

**Description**

This function reads a directory that contains the output psi-blast. It gets the directory path as the input and returns a list of vectors. Each vector includes the type for the amino acids of the sequence.

**Usage**

```r
readPSSMdir(dirPath)
```

**Arguments**

- `dirPath`  
  the path of a directory which contains all the VSL2 output files.
The function reads a directory that contains the output files of PSIPRED. It gets the directory path as the input and returns a list of vectors. Each vector contains the secondary structure of the amino acids in a peptide/protein sequence.

**Usage**

```r
readss2Dir(dirPath)
```

**Arguments**

- `dirPath`: The path of the directory which contains all predss2 files. Each file belongs to a sequence.

**Value**

returns a list of vectors with all the predicted secondary structure for each amino acid. The length of the list is the number of files(sequences) and the length of each vector is (length sequence(i))

**Examples**

```r
PredSS2dir<-system.file("testFolder",package="ftrCOOL")
PredSS2dir<-paste0(PredSS2dir,"/ss2Dir/")
listPredVect<-readss2Dir(PredSS2dir)
```
**readTorsionDir**  
*Read Directory of Torsion predicted files (readTorsionDir)*

**Description**

This function reads a directory that contains the output files of SPINE-X. It gets the directory path as the input and returns a list of vectors. Each vector includes the phi and psi angle of the amino acids of the sequence.

**Usage**

```r
readTorsionDir(dirPath)
```

**Arguments**

- `dirPath`  
  The path of the directory which contains all output files of SPINE-X. Each file belongs to a sequence.

**Value**

returns a list of vectors with all the predicted phi and psi angles for each amino acid. The length of the list is the number of files(sequences) and the length of each vector is `(2(\text{phi-psi})*\text{length sequence}(i))`.

**Examples**

```r
PredTorsioNdir<-system.file("testForder",package="ftrCOOL")
PredTorsioNdir<-paste0(PredTorsioNdir,"/TorsioNdir/")
PredVectASA<-readTorsionDir(PredTorsioNdir)
```

**revComp**  
*reverseCompelement (revComp)*

**Description**

This function returns the reverse complement of a dna sequence.

**Usage**

```r
revComp(seq, outputType = "str")
```
Arguments

**seq**

is a DNA sequence.

**outputType**

this parameter can take two values: 'char' or 'str'. If outputType is 'str', the reverse complement sequence of the input sequence is returned as a string. Otherwise, a vector of characters which represent the reverse complement is returned. Default value is 'str'.

Value

The reverse complement of the input sequence.

Examples

```r
ptmSeqsADR <- system.file("extdata/", package="ftrCOOL")
ptmSeqsVect <- as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[2])
Seq <- ptmSeqsVect[1]
revCompSeq <- revComp(seq = Seq, outputType = "char")
```

SAAC

**Splitted Amino Acid Composition (SAAC)**

Description

This function splits the input sequence into three parts. The first part is N-terminal and the third part is C-terminal and middle part contains all amino acids between these two part. N-terminal will be determined by the first numNterm amino acid in the sequences and C-terminal is determined by numCterm of the last amino acids in the sequence. Users should enter numNterm and numCterm parameters. Their default value is 25. The function calculates kAAComposition for each of the three parts.

Usage

```r
SAAC(seqs, k = 1, numNterm = 5, numCterm = 5, normalized = TRUE, label = c())
```

Arguments

**seqs**

is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.

**k**

shows which type of amino acid composition applies to the parts. For example, the amino acid composition is applied when k=1 and when k=2, the dipeptide Composition is applied.

**numNterm**

shows how many amino acids should be considered for N-terminal.

**numCterm**

shows how many amino acids should be considered for C-terminal.
SGAAC

is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

It returns a feature matrix. The number of rows is equal to the number of sequences. The number of columns is \(3\times(20^k)\).

Examples

```r
filePrs <- system.file("extdata/proteins.fasta", package="ftrCOOL")
mat <- SGAAC(seqs = filePrs, k = 1, numNterm = 15, numCterm = 15)
```

Description

In this function, amino acids are first grouped into a user-defined category. Later, the splitted amino Acid composition is computed. Please note that this function differs from SAAC which works on individual amino acids.

Usage

```r
SGAAC(
  seqs,
  k = 1,
  numNterm = 25,
  numCterm = 25,
  Grp = "locFus",
  normalized = TRUE,
  label = c()
)
```

Arguments

- **seqs** is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- **k** shows which type of amino acid composition applies to the parts. For example, the amino acid composition is applied when \(k=1\) and when \(k=2\), the dipeptide Composition is applied.
- **numNterm** shows how many amino acids should be considered for N-terminal.
SOCNumber

**numCterm**
- shows how many amino acids should be considered for C-terminal.

**Grp**
- is a list of vectors containing amino acids. Each vector represents a category. Users can define a customized amino acid grouping, provided that the sum of all amino acids is 20 and there is no repeated amino acid in the groups. Also, users can choose 'cTriad'(conjointTriad), 'locFus', or 'aromatic'. Each option provides specific information about the type of an amino acid grouping.

**normalized**
- is a logical parameter. When it is FALSE, the return value of the function does not change. Otherwise, the return value is normalized using the length of the sequence.

**label**
- is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

**Value**
- It returns a feature matrix. The number of rows is equal to the number of sequences. The number of columns is $3 \times (\text{number of groups})^k$.

**Examples**
```r
filePrs <- system.file("extdata/proteins.fasta", package="ftrCOOL")
mat <- SGAAC(seqs = filePrs, k = 1, numNterm = 15, numCterm = 15, Grp = "aromatic")
```

**SOCNumber**

**Sequence Order Coupling Number (SOCNumber)**

**Description**
- This function uses dissimilarity matrices Grantham and Schneider to compute the dissimilarity between amino acid pairs. The distance between amino acid pairs is determined by $d$ which varies between 1 to $nlag$. For each $d$, it computes the sum of the dissimilarities of all amino acid pairs. The sum shows the value of tau for a value $d$. The feature vector contains the values of taus for both matrices. Thus, the length of the feature vector is equal to $nlag \times 2$.

**Usage**

```r
SOCNumber(seqs, nlag = 30, label = c())
```

**Arguments**
- **seqs**
  - is a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- **nlag**
  - is a numeric value which shows the maximum distance between two amino acids. Distances can be 1, 2, ..., or nlag. Default is 30.
- **label**
  - is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
Value

It returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is \((nlag*2)\). For each distance \(d\), there are two values. One value for Granthman and another one for Schneider distance.

Note

When \(d=1\), the pairs of amino acids have no gap and when \(d=2\), there is one gap between the amino acid pairs in the sequence. It will repeat likewise for other values of \(d\).

Examples

```r
filePrs <- system.file("extdata/proteins.fasta", package="ftrCOOL")
mat <- SOCNumber(seqs=filePrs, nlag=25)
```

SSEB

Secondary Structure Elements Binary (SSEB)

Description

This function works based on the output of PSIPRED which predicts the secondary structure of the amino acids in a sequence. The output of the PSIPRED is a tab-delimited file which contains the secondary structure in the third column. SSEB gives a binary number (i.e., ‘001’=’H’, ‘010’=E’, ‘100’=’C’) for each amino acid.

Usage

```r
SSEB(dirPath, binaryType = "numBin", outFormat = "mat", outputFileDist = "")
```

Arguments

- `dirPath` : Path of the directory which contains all output files of PSIPRED. Each file belongs to a sequence.
- `binaryType` : It can take any of the following values: (‘strBin’, ‘logicBin’, ‘numBin’). ‘strBin’ (String binary): each structure is represented by a string containing 3 characters (0-1). Helix = ”001″, Extended = ”010″, coil = ”100″. ‘logicBin’ (logical value): Each structure is represented by a vector containing 3 logical entries. Helix = c(FALSE,FALSE,TRUE), Extended = c(FALSE,TRUE,FALSE), Coil = c(TRUE,FALSE,FALSE). ‘numBin’ (numeric bin): Each structure is represented by a numeric (i.e., integer) vector containing 3 numerals. Helix = c(0,0,1), Extended = c(0,1,0), coil = c(1,0,0).
- `outFormat` : It can take two values: ‘mat’ (which stands for matrix) and ‘txt’. The default value is ‘mat’.
- `outputFileDist` : It shows the path and name of the ‘txt’ output file.
This function converts each amino acid to a 3-bit value, such that 2 bits are 0 and 1 bit is 1. The position of 1 shows the type of the secondary structure of the amino acids in the protein/peptide. In this function, '001' is used to show Helix structure, '010' to show Extended structure and '100' to show coil structure.

Value

The output is different depending on the outFormat parameter ('mat' or 'txt'). If outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and if binaryType is 'strBin', the number of columns is the length of the sequences. Otherwise, it is equal to (length of the sequences)*3. If outFormat is 'txt', all binary values will be written to a tab-delimited file. Each line in the file shows the binary format of a sequence.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in the outFormat parameter for sequences with different lengths. Warning: If the outFormat is set to 'mat' for sequences with different lengths, it returns an error. It is noteworthy that 'txt' format is not usable for machine learning purposes.

Examples

```r
dir = tempdir()
ad<-paste0(dir,"/SSEB.txt")
Predss2dir<-system.file("testOrder",package="ftrCOOL")
Predss2dir<-paste0(Predss2dir,"/ss2Dir/"
mat<-SSEB(Predss2dir,binaryType="numBin",outFormat="txt",outputFileDist=ad)
unlink("dir", recursive = TRUE)
```
Arguments
dirPath Path of the directory which contains all output files of PSIPRED. Each file belongs to a sequence.

Value
It returns a feature matrix which the number of rows is the number of sequences and the number of columns is 3. The first column shows the number of amino acids which participate in the coil structure. The second column shows the number of amino acids in the extended structure and the last column shows the number of amino acids in the helix structure.

Examples

```r
Predss2dir <- system.file("testForder", package="ftrCOOL")
Predss2dir <- paste0(Predss2dir, "/ss2Dir/")
mat <- SSEC(Predss2dir)
```

---

**SSES**

**Secondary Structure Elements Simple (SSES)**

Description
This function works based on the output of PSIPRED which predicts the secondary structure of the amino acids in a sequence. The output of the PSIPRED is a tab-delimited file which contains the secondary structure in the third column. The function represents amino acids in the helix structure by 'H', amino acids in the extended structure by 'E', and amino acids in the coil structure by 'C'.

Usage
```
SSES(dirPath, outFormat = "mat", outputFileDist = "")
```

Arguments
dirPath Path of the directory which contains all output files of PSIPRED. Each file belongs to a sequence.
outFormat It can take two values: 'mat' (which stands for matrix) and 'txt'. The default value is 'mat'.
outputFileDist It shows the path and name of the 'txt' output file.

Value
The output depends on the outFormat which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same lengths such that the number of columns is equal to the length of the sequences and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.
**Note**

This function is provided for the sequences with the same lengths. However, the users can use 'txt' option in the outFormat parameter for sequences with different lengths. Warning: If the outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when the output format is 'txt', the label information is not displayed in the text file. It is noteworthy that, 'txt' format is not usable for machine learning purposes.

**Examples**

```r
dir = tempdir()
ad<-paste0(dir,"/simpleSSE.txt")
Predss2dir<-system.file("testFolder",package="ftrCOOL")
Predss2dir<-paste0(Predss2dir,"/ss2Dir/")
mat<-SSES(Predss2dir,outFormat="txt",outputFileDist=ad)
unlink("dir", recursive = TRUE)
```

---

**Description**

The inputs to this function are phi and psi angles of each amino acid in the sequence. We use the output of SPINE-X software to obtain the angles. Further, the TA function replaces each amino acid of the sequence with a vector. The vector contain two elements: The phi and psi angles.

**Usage**

`TorsionAngle(dirPath, outFormat = "mat", outputFileDist = "")`

**Arguments**

- `dirPath` Path of the directory which contains all output files of SPINE-X. Each file belongs to a sequence.
- `outFormat` It can take two values: 'mat' (which stands for matrix) and 'txt'. The default value is 'mat'.
- `outputFileDist` It shows the path and name of the 'txt' output file.

**Value**

The output is different depending on the outFormat parameter ('mat' or 'txt'). If the outFormat is set to 'mat', it returns a feature matrix for sequences with the same lengths. The number of rows is equal to the number of sequences and the number of columns is (length of the sequence)*2. If the outFormat is set to 'txt', all binary values will be written in a 'txt' file. Each row belongs to a sequence.
Note

This function is provided for sequences with the same lengths. Users can use ‘txt’ option in outFormat parameter for sequences with different lengths. Warning: If the outFormat is set to ‘mat’ for sequences with different lengths, it returns an error. It is noteworthy that ‘txt’ format is not usable for machine learning purposes.

Examples

dir = tempdir()
ad<-paste0(dir,"/ta.txt")
PredTorsioNdir<-system.file("testForder",package="ftrCOOL")
PredTorsioNdir<-paste0(PredTorsioNdir,"/TorsioNdir/")
mat<-TorsionAngle(PredTorsioNdir,outFormat="txt",outputFileDist=ad)
unlink("dir", recursive = TRUE)

TPCP_DNA

Trinucleotide physicochemical properties (TPCP_DNA)

Description

This function replaces trinucleotides in a sequence with their physicochemical properties which is multiplied by normalized frequency of that tri-nucleotide.

Usage

TPCP_DNA(
    seqs,
    selectedIdx = c("Dnase I", "Bendability (DNAse)")
    threshold = 1,
    label = c(),
    outFormat = "mat",
    outputFileDist = ""
)

Arguments

seqs is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

selectedIdx TPCP_DNA function works based on physicochemical properties. Users, select the properties by their ids or indexes in TRI_DNA index file. The default values of the vector are the ids in "Dnase I", "Bendability (DNAse)".

threshold is a number between 0 to 1. In selectedIdx, indices with a correlation higher than the threshold will be deleted. The default value is 1.
label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

outFormat (output format) can take two values: 'mat' (matrix) and 'txt'. The default value is 'mat'.

outputFileDist shows the path and name of the 'txt' output file.

Details

There are 12 physicochemical indexes in the trinucleotide database.

Value

The output depends on the outFormat parameter which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same length such that the number of columns is (sequence length-2)*(number of selected trinucleotide properties) and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes if sequences have different sizes. Otherwise 'txt' format is also usable for machine learning purposes.

Examples

```r
fileLNC<-system.file("extdata/Athaliana1.fa",package="ftrCOOL")
vect<-TPCP_DNA(seqs = fileLNC,threshold=1,outFormat="mat")
```

---

TriNUCindex_DNA  Tri Nucleotide Index (TriNucIndex)

Description

This function replaces trinucleotides in a sequence with their physicochemical properties in the trinucleotide index file.

Usage

```r
TriNUCindex_DNA(
  seqs,
  selectedNucIdx = c("Dnase I", "Bendability (DNAse)"),
  threshold = 1,
  label = c(),
)```
TriNUCIndex_DNA

```r
outFormat = "mat",
outputFileDist = ""
)
```

Arguments

- **seqs**: is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
- **selectedNucIdx**: TriNucIndex function works based on physicochemical properties. Users, select the properties by their ids or indexes in TRI_DNA index file. The default values of the vector are the ids in "Dnase I", "Bendability (DNase)".
- **threshold**: is a number between 0 to 1. In selectedNucIdx, indices with a correlation higher than the threshold will be deleted. The default value is 1.
- **label**: is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
- **outFormat**: (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.
- **outputFileDist**: shows the path and name of the 'txt' output file.

Details

There are 12 physicochemical indexes in the trinucleotide database.

Value

The output depends on the outFormat parameter which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same length such that the number of columns is (sequence length-2)*(number of selected trinucleotide properties) and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.

Note

This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat parameter for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes.

Examples

```r
fileLNC<-system.file("extdata/Athaliana1.fa",package="ftrCOOL")
vect<-TriNUCIndex_DNA(seqs = fileLNC,threshold=1,outFormat="mat")
```
Description

These group of functions (Zcurve (9, 12, 36, 48, 144)_bit) function calculates the Z-curves. Z-curves are based on frequencies of nucleotides, di-nucleotides, or tri-nucleotides and their positions on the sequences. For more information about the methods please refer to reference part.

Usage

`Zcurve12bit_DNA(seqs, ORF = FALSE, reverseORF = TRUE, label = c())`

Arguments

- **seqs**: is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
- **ORF**: (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).
- **reverseORF**: is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).
- **label**: is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 12.

References


Examples

```r
fileLNC<system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
mat<-Zcurve12bit_DNA(seqs=fileLNC,ORF=TRUE,reverseORF=FALSE)```
Zcurve12bit_RNA

---

**Description**

These group of functions (Zcurve (9, 12, 36, 48, 144)_bit) function calculates the Z-curves. Z-curves are based on frequencies of ribonucleotides, di-ribonucleotides, or tri-ribonucleotides and their positions on the sequences. For more information about the methods please refer to reference part.

**Usage**

Zcurve12bit_RNA(seqs, ORF = FALSE, reverseORF = TRUE, label = c())

**Arguments**

- **seqs**
  - is a FASTA file containing ribonucleotide sequences. The sequences start with '>'. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.
- **ORF**
  - (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).
- **reverseORF**
  - is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).
- **label**
  - is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

**Value**

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 12.

**References**


**Examples**

```r
fileLNC<-system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
mat<-Zcurve12bit_RNA(seqs=fileLNC,ORF=TRUE,reverseORF=FALSE)
```
**Description**

These group of functions (Zcurve (9, 12, 36, 48, 144)bit) function calculates the Z-curves. Z-curves are based on frequencies of nucleotides, di-nucleotides, or tri-nucleotides and their positions on the sequences. For more information about the methods please refer to reference part.

**Usage**

```
Zcurve144bit_DNA(seqs, ORF = FALSE, reverseORF = TRUE, label = c())
```

**Arguments**

- `seqs` is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
- `ORF` (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).
- `reverseORF` is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).
- `label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

**Value**

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 144.

**References**


**Examples**

```r
fileLNC <- system.file("extdata/Athaliana_LNCRNA.fa", package="ftrCOOL")
mat <- Zcurve144bit_DNA(seqs=fileLNC, ORF=TRUE, reverseORF=FALSE)
```
Description

These group of functions (Zcurve (9, 12, 36, 48, 144)_bit) function calculates the Z-curves. Z-curves are based on frequencies of ribonucleotides, di-ribonucleotides, or tri-ribonucleotides and their positions on the sequences. For more information about the methods please refer to reference part.

Usage

Zcurve144bit_RNA(seqs, ORF = FALSE, reverseORF = TRUE, label = c())

Arguments

- **seqs** is a FASTA file containing ribonucleotide sequences. The sequences start with '>'. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.
- **ORF** (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).
- **reverseORF** is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).
- **label** is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 144.

References


Examples

fileLNC<-system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
mat<-Zcurve144bit_RNA(seqs=fileLNC,ORF=TRUE,reverseORF=FALSE)
Zcurve36bit_DNA

Description

These group of functions (Zcurve (9, 12, 36, 48, 144)_bit) function calculates the Z-curves. Z-curves are based on frequencies of nucleotides, di-nucleotides, or tri-nucleotides and their positions on the sequences. For more information about the methods please refer to reference part.

Usage

Zcurve36bit_DNA(seqs, ORF = FALSE, reverseORF = TRUE, label = c())

Arguments

seqs is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

ORF (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

reverseORF is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 36.

References


Examples

fileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
mat<-Zcurve36bit_DNA(seqs=fileLNC,ORF=TRUE,reverseORF=FALSE)
**Description**

These group of functions (Zcurve (9, 12, 36, 48, 144)_bit) function calculates the Z-curves. Z-curves are based on frequencies of ribonucleotides, di-ribonucleotides, or tri-ribonucleotides and their positions on the sequences. For more information about the methods please refer to reference part.

**Usage**

Zcurve36bit_RNA(seqs, ORF = FALSE, reverseORF = TRUE, label = c())

**Arguments**

- `seqs` is a FASTA file containing ribonucleotide sequences. The sequences start with '>'. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.
- `ORF` (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).
- `reverseORF` is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).
- `label` is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

**Value**

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 36.

**References**


**Examples**

```r
generic<--system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
mat<-Zcurve36bit_RNA(seqs=feneric,ORF=TRUE,reverseORF=FALSE)
```
Zcurve48bit_DNA

Description

These group of functions (Zcurve (9, 12, 36, 48, 144)_bit) function calculates the Z-curves. Z-curves are based on frequencies of nucleotides, di-nucleotides, or tri-nucleotides and their positions on the sequences. For more information about the methods please refer to reference part.

Usage

Zcurve48bit_DNA(seqs, ORF = FALSE, reverseORF = TRUE, label = c())

Arguments

- **seqs**
  is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.

- **ORF**
  (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

- **reverseORF**
  is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

- **label**
  is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 48.

References


Examples

```r
fileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
mat<-Zcurve48bit_DNA(seqs=fileLNC,ORF=TRUE,reverseORF=FALSE)
```
Zcurve48bit_RNA

Z_curve_48bit_RNA (Zcurve48bit_RNA)

Description
These group of functions (Zcurve (9, 12, 36, 48, 144)_bit) function calculates the Z-curves. Z-curves are based on frequencies of ribo ribonucleotides, di-ribonucleotides, or tri-ribonucleotides and their positions on the sequences. For more information about the methods please refer to reference part.

Usage
Zcurve48bit_RNA(seqs, ORF = FALSE, reverseORF = TRUE, label = c())

Arguments
- seqs is a FASTA file containing ribonucleotide sequences. The sequences start with '>'. Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.
- ORF (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).
- reverseORF is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).
- label is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value
This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 48.

References

Examples
fileLNC<-system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
mat<-Zcurve48bit_RNA(seqs=fileLNC,ORF=TRUE,reverseORF=FALSE)
Zcurve9bit_DNA: Description

These group of functions (Zcurve (9, 12, 36, 48, 144)_bit) function calculates the Z-curves. Z-curves are based on frequencies of nucleotides, di-nucleotides, or tri-nucleotides and their positions on the sequences. For more information about the methods please refer to reference part.

Usage

Zcurve9bit_DNA(seqs, ORF = FALSE, reverseORF = TRUE, label = c())

Arguments

- seqs: is a FASTA file containing nucleotide sequences. The sequences start with ‘>’. Also, seqs could be a string vector. Each element of the vector is a nucleotide sequence.
- ORF: (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).
- reverseORF: is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).
- label: is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 9.

References


Examples

```r
fileLNC<-system.file("extdata/Athaliana_LNCRNA.fa",package="ftrCOOL")
mat<-Zcurve9bit_DNA(seqs=fileLNC,ORF=TRUE,reverseORF=FALSE)
```
Description

These group of functions (Zcurve (9, 12, 36, 48, 144)_bit) function calculates the Z-curves. Z-curves are based on frequencies of ribo ribonucleotides, di-ribonucleotides, or tri-ribonucleotides and their positions on the sequences. For more information about the methods please refer to reference part.

Usage

Zcurve9bit_RNA(seqs, ORF = FALSE, reverseORF = TRUE, label = c())

Arguments

seqs  is a FASTA file containing ribonucleotide sequences. The sequences start with '>' . Also, seqs could be a string vector. Each element of the vector is a ribonucleotide sequence.

ORF  (Open Reading Frame) is a logical parameter. If it is set to true, ORF region of each sequence is considered instead of the original sequence (i.e., 3-frame).

reverseORF  is a logical parameter. It is enabled only if ORF is true. If reverseORF is true, ORF region will be searched in the sequence and also in the reverse complement of the sequence (i.e., 6-frame).

label  is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).

Value

This function returns a feature matrix. The number of rows is equal to the number of sequences and the number of columns is 9.

References


Examples

fileLNC<-system.file("extdata/Carica_papaya101RNA.txt",package="ftrCOOL")
mat<-Zcurve9bit_RNA(seqs=fileLNC,ORF=TRUE,reverseORF=FALSE)
**Z-SCALE (zSCALE)**

**Description**
This function converts the amino acids of a sequence to five physicochemical descriptor variables which were developed by Sandberg et al. in 1998. The Z-SCALE function can be applied to encode peptides of equal length.

**Usage**
```r
zSCALE(seqs, label = c(), outFormat = "mat", outputFileDist = "")
```

**Arguments**
- `seqs`: a FASTA file with amino acid sequences. Each sequence starts with a '>' character. Also, seqs could be a string vector. Each element of the vector is a peptide/protein sequence.
- `label`: is an optional parameter. It is a vector whose length is equivalent to the number of sequences. It shows the class of each entry (i.e., sequence).
- `outFormat` (output format) can take two values: 'mat'(matrix) and 'txt'. The default value is 'mat'.
- `outputFileDist`: shows the path and name of the 'txt' output file.

**Value**
The output depends on the outFormat parameter which can be either 'mat' or 'txt'. If outFormat is 'mat', the function returns a feature matrix for sequences with the same length such that the number of columns is (sequence length)*(5) and the number of rows is equal to the number of sequences. If the outFormat is 'txt', the output is written to a tab-delimited file.

**Note**
This function is provided for sequences with the same lengths. Users can use 'txt' option in outFormat parameter for sequences with different lengths. Warning: If outFormat is set to 'mat' for sequences with different lengths, it returns an error. Also, when output format is 'txt', label information is not shown in the text file. It is noteworthy that 'txt' format is not usable for machine learning purposes.

**Examples**
```r
ptmSeqsADR<-system.file("extdata/",package="ftrCOOL")
ptmSeqsVect<-as.vector(read.csv(paste0(ptmSeqsADR,"/ptmVect101AA.csv"))[,2])
mat<-zSCALE(seqs = ptmSeqsVect,outFormat="mat")
```
Index

AA2Binary, 5
AAindex, 7
AAKpartComposition, 8, 85
AAutoCor, 9
AESNN3, 11
alphabetCheck, 12
ANF_DNA, 13
ANF_RNA, 14
APAAC, 15
APkNUCdi_DNA, 16
APkNUCdi_RNA, 17
APkNUCTri_DNA, 19
ASA, 20
ASDC, 21
ASDC_DNA, 22
ASDC_RNA, 23
AutoCorDiNUC_DNA, 24
AutoCorDiNUC_RNA, 25
AutoCorTriNUC_DNA, 27
binary_3bit_T1, 28
binary_3bit_T2, 30
binary_3bit_T3, 31
binary_3bit_T4, 32
binary_3bit_T5, 34
binary_3bit_T6, 35
binary_3bit_T7, 36
binary_5bit_T1, 38
binary_5bit_T2, 39
binary_6bit, 41
BLOSUM62, 42

CkSAApair, 43, 44
CkSGAAPair, 44
CkSNUCpair_DNA, 46
CkSNUCpair_RNA, 47
codonAdaptionIndex, 48
CodonFraction, 49
CodonUsage_DNA, 50
CodonUsage_RNA, 51

conjointTriad, 51
conjointTriadKS, 52
CTD, 53, 54–56
CTDC, 54, 54
CTDD, 54, 55
CTDT, 54, 56

DDE, 57, 87
DiNUC2Binary_DNA, 58
DiNUC2Binary_RNA, 59
DiNUCindex_DNA, 60
DiNUCindex_RNA, 62
DisorderB, 63
DisorderC, 64
DisorderS, 65
DistancePair, 66
DPCP_DNA, 67
DPCP_RNA, 68

EAAComposition, 69, 72
EffectiveNumberCodon, 71
EGAAComposition, 72
EIIP, 73
ENUComposition_DNA, 75
ENUComposition_RNA, 76
ExpectedValKmerNUC_DNA, 77
ExpectedValKmerNUC_RNA, 78
ExpectedValueAA, 80, 80
ExpectedValueGAA, 80
ExpectedValueGKmerAA, 81
ExpectedValueKmerAA, 82, 83

fa.read, 84
fickettScore, 85

G_Ccontent_DNA, 88
G_Ccontent_RNA, 89
GAAComposition, 85
GrpDDE, 87

kAAComposition, 90, 91, 180
kGAACComposition, 91
KNN_DNA, 95
KNN_RNA, 96
KNNPeptide, 92, 93, 95, 96
KNNProtein, 93
KNUComposition_DNA, 97
KNUComposition_RNA, 98
LocalPoSpKAAF, 99
LocalPoSpKNUCF_DNA, 101
LocalPoSpKNUCF_RNA, 102
maxORF, 103
maxORF_RNA, 106
maxORF_length_DNA, 104
maxORF_length_RNA, 105
Mismatch_DNA, 107
Mismatch_RNA, 108
MMI_DNA, 109
MMI_RNA, 110
nameKmer, 111
NCP_DNA, 111
NCP_RNA, 113
needleman, 115
nonStandardSeq, 115
NUC2Binary_DNA, 116
NUC2Binary_RNA, 118
NUCKpartComposition_DNA, 119
NUCKpartComposition_RNA, 120
OPF_10bit, 121
OPF_7bit_T1, 122
OPF_7bit_T2, 124
OPF_7bit_T3, 125
PCPseDNC, 126
PS2_DNA, 128
PS2_RNA, 129
PS3_DNA, 131
PS3_RNA, 132
PS4_DNA, 134
PS4_RNA, 135
PSEAAC, 137
PseEIIP, 138
PSEkNUCd1_DNA, 126, 139
PSEkNUCd1_RNA, 140
PSEkNUCtri_DNA, 142
PseKRAAC_T1, 143
PseKRAAC_T10, 145
PseKRAAC_T11, 146
PseKRAAC_T12, 148
PseKRAAC_T13, 149
PseKRAAC_T14, 151
PseKRAAC_T15, 152
PseKRAAC_T16, 154
PseKRAAC_T2, 156
PseKRAAC_T3A, 157
PseKRAAC_T3B, 159
PseKRAAC_T4, 160
PseKRAAC_T5, 162
PseKRAAC_T6A, 163
PseKRAAC_T6B, 164
PseKRAAC_T7, 166
PseKRAAC_T8, 167
PseKRAAC_T9, 169
PSSM, 170
PSTNPds, 171
PSTNPss_DNA, 171, 173
PSTNPss_RNA, 174
QSOrder, 175
readASAdir, 176
readDisDir, 177
readPSSMdir, 177
readss2Dir, 178
readTorsionDir, 179
revComp, 179
SAAC, 180, 181
SGAAC, 181
SOCNumber, 175, 182
SSEB, 183
SSEC, 184
SSES, 185
TorsionAngle, 186
TPCP_DNA, 187
TriNUCindex_DNA, 188
Zcurve12bit_DNA, 190
Zcurve12bit_RNA, 191
Zcurve144bit_DNA, 192
Zcurve144bit_RNA, 193
Zcurve36bit_DNA, 194
Zcurve36bit_RNA, 195
Zcurve48bit_DNA, 196
Zcurve48bit_RNA, 197
Zcurve9bit_DNA, 198
Zcurve9bit_RNA, 199
zSCALE, 200