Package ‘peptoolkit’

July 13, 2023

**Type** Package

**Title** A Toolkit for Using Peptide Sequences in Machine Learning

**Version** 0.0.1

**Description** This toolkit is designed for manipulation and analysis of peptides. It provides functionalities to assist researchers in peptide engineering and proteomics. Users can manipulate peptides by adding amino acids at every position, count occurrences of each amino acid at each position, and transform amino acid counts based on probabilities. The package offers functionalities to select the best versus the worst peptides and analyze these peptides, which includes counting specific residues, reducing peptide sequences, extracting features through One Hot Encoding (OHE), and utilizing Quantitative Structure-Activity Relationship (QSAR) properties (based in the package 'Peptides' by Osorio et al. (2015) <doi:10.32614/RJ-2015-001>). This package is intended for both researchers and bioinformatics enthusiasts working on peptide-based projects, especially for their use with machine learning.

**Depends** R (>= 4.3.0)

**License** GPL (>= 3)

**Encoding** UTF-8

**Imports** Peptides, stats, dplyr, stringr, caret

**RoxygenNote** 7.2.3

**URL** https://github.com/jrcodina/peptoolkit

**BugReports** https://github.com/jrcodina/peptoolkit/issues

**NeedsCompilation** no

**Author** Josep-Ramon Codina [aut, cre] (<https://orcid.org/0000-0003-4391-450X>)

**Maintainer** Josep-Ramon Codina <jrc356@miami.edu>

**Repository** CRAN

**Date/Publication** 2023-07-13 15:30:06 UTC

R topics documented:

  appearance_to_binary ........................................... 2
  count_aa ........................................................... 3
appearance_to_binary

Transform Amino Acid Appearance Probability into -1, 0, or 1

Description

This function transforms the counts of amino acids to a -1, 0, 1 matrix based on a probability of appearance of each peptide in each position.

Usage

appearance_to_binary(x, threshold = 1.65, group = "Best", percentage = 0.05)

Arguments

x A data frame containing peptide sequences.
threshold The probability threshold to determine the transformation.
group A character string indicating which part of the data to consider. Either 'Best' or 'Worst'.
percentage The percentage of the data to consider, if group is specified.

Value

A matrix with the same dimensions as the input where each cell has been transformed to -1, 0, or 1 based on the probability threshold.

Examples

# Generate a mock data frame
peptide_data <- data.frame(Sequence = c("ACGT", "TGCA", "GATC", "CGAT"))

# Apply the function to the mock data
appearance_to_binary(peptide_data, group = "Best", percentage = 0.5)
### count_aa

**Count Amino Acids**

**Description**

This function counts the occurrence of each of the 20 amino acids at each of the first `n` positions across a vector of peptide sequences.

**Usage**

```r
count_aa(peptides, n = 4)
```

**Arguments**

- `peptides`: A character vector of peptide sequences.
- `n`: The number of initial positions to consider in each peptide sequence.

**Value**

A data frame with `n` rows and 20 columns where each row represents a position in the peptide sequence and each column represents an amino acid. Each cell in the data frame contains the count of a particular amino acid at a particular position.

**Examples**

```r
count_aa(c("ACDF", "BCDE", "ABCD"), n = 2)
```

### extract_features_OHE

**Extract One-Hot Encoded (OHE) Features from Peptide Sequences**

**Description**

This function takes a data frame or a vector of peptide sequences and generates a one-hot encoded data frame representing each amino acid in the sequences. It can also include additional data (such as docking information), if provided.

**Usage**

```r
extract_features_OHE(df, sequence_col = "Sequence", docking_col = NULL)
```

**Arguments**

- `df`: A data frame or a vector of peptide sequences.
- `sequence_col`: A string representing the name of the column containing the peptide sequences.
- `docking_col`: A string representing the name of the column containing the docking information.
extract_features_QSAR

Value

A data frame containing one-hot encoded peptide sequences and, if provided, docking information.

Examples

```r
# Load required library caret
library(caret)
# Generate a mock data frame of peptide sequences
df <- data.frame(Sequence = c("AVILG", "VILGA", "ILGAV", "LGAVI"), X = c(1,1,2,3))
# Apply the function to the mock data
extract_features_OHE(df)
```

---

**extract_features_QSAR**  
*Extract QSAR Features from Peptide Sequences*

**Description**

This function extracts various Quantitative Structure-Activity Relationship (QSAR) features from peptide sequences. The extraction is based on a variety of amino acid properties and functions from the "Peptides" package (https://github.com/dosorio/Peptides/).

**Usage**

```r
extract_features_QSAR(
  n,
  pH = 7.4,
  custom.list = FALSE,
  PeList = NULL,
  rem.cys = FALSE,
  rem.met = FALSE,
  rem.sali = FALSE,
  norm = FALSE
)
```

**Arguments**

- `n`: The length of the peptide sequences. Must be more than 2.
- `pH`: The pH used for calculating charge (default is 7.4).
- `custom.list`: A boolean indicating if a custom peptide list is provided (default is FALSE).
- `PeList`: The custom list of peptides (required if custom.list is TRUE).
- `rem.cys`: A boolean indicating if sequences with Cys should be removed (default is FALSE).
- `rem.met`: A boolean indicating if sequences with Met should be removed (default is FALSE).
- `rem.sali`: A boolean indicating if sequences with 2 or more small aliphatic amino acids should be removed (default is FALSE).
- `norm`: A boolean indicating if the data should be normalized (default is FALSE).
filter_residues  

Filter Peptides by Residue Counts

Description
This function counts the number of specified residues in each peptide sequence and filters out the ones with more than the specified limit. It’s defaults is for filtering out small alliphatic residues.

Usage

```r
filter_residues(
  df,
  sequence_col = "Sequence",
  residues = c("A", "V", "I", "L", "G"),
  max_residues = 2
)
```

Arguments

- `df`  
  A data frame containing peptide sequences.
- `sequence_col`  
  The name of the column that contains the sequences.
- `residues`  
  A character vector of residues to count.
- `max_residues`  
  The maximum number of allowed residues.

Value

A filtered data frame.

Examples

```r
# Generate a mock data frame
peptide_data <- data.frame(Sequence = c("AVILG", "VILGA", "ILGAV", "LGAVI"))
# Apply the function to the mock data
filter_residues(peptide_data, residues = c("A", "V", "I", "L", "G"), max_residues = 2)
```
reduce_sequences

increment

Increment Peptide Sequences

Description
This function generates new peptide sequences by adding each of the 20 amino acids to each position of the input peptide or peptides.

Usage
increment(peptides, num_added = 1)

Arguments
- peptides: A character vector of peptide sequences.
- num_added: The number of amino acids to be added to each position of the peptide.

Value
A character vector of new peptide sequences.

Examples
increment(c("AC", "DE"))
increment("ACDE", num_added = 2)

reduce_sequences

Reduce Peptide Sequences by One Residue

Description
This function takes a vector of peptide sequences and generates all possible sequences by removing one amino acid residue at a time. It can also associate each sequence with an ID, if provided.

Usage
reduce_sequences(peptides, id = NULL)

Arguments
- peptides: A character vector of peptide sequences.
- id: A character vector of IDs that correspond to the peptides.

Value
A list of data frames, each containing all possible sequences resulting from removing one amino acid from the original sequence.
select_best_vs_worst

Examples

# Generate a mock vector of peptide sequences
peptides <- c("AVILG", "VILGA", "ILGAV", "LGAVI")
# Apply the function to the mock data
reduce_sequences(peptides)

select_best_vs_worst  Select Best vs Worst Peptides

Description

This function identifies the peptides from the function *appearance_to_binary* that are 1 in one group and 0 or -1 in another group, and expands the grid to all possible combinations.

Usage

select_best_vs_worst(appearance_best, appearance_worst)

Arguments

appearance_best
A matrix with transformed counts for the 'best' group.

appearance_worst
A matrix with transformed counts for the 'worst' group.

Value

A data frame with combinations of 'best' peptides.

Examples

# Generate some mock data
appearance_best <- matrix(c(1, -1, 0, 1, -1), nrow = 5, ncol = 4)
appearance_worst <- matrix(c(-1, 1, 0, -1, 1), nrow = 5, ncol = 4)
# Call the function
select_best_vs_worst(appearance_best, appearance_worst)
Index

appearance_to_binary, 2
count_aa, 3
extract_features_OHE, 3
extract_features_QSAR, 4
filter_residues, 5
increment, 6
reduce_sequences, 6
select_best_vs_worst, 7