protr: Protein Sequence Feature Extraction with R

Xiao Nan  
Central South University

Dongsheng Cao  
Central South University

Qingsong Xu  
Central South University

Yizeng Liang  
Central South University

Abstract

The protr package aims for protein sequence feature extraction, which could be easily applied in Bioinformatics and Chemogenomics research. The descriptors listed in this package include Amino Acid Composition (Amino Acid Composition/Dipeptide Composition/Tripeptide Composition), Autocorrelation (Normalized Moreau-Broto Autocorrelation/Moran Autocorrelation/Geary Autocorrelation), CTD (Composition/Transition/Distribution), Conjoint Traid, Quasi-Sequence Order (Sequence Order Coupling Number/Quasi-sequence Order Descriptors), and Pseudo Amino Acid Composition (Pseudo Amino Acid Composition/Amphiphilic Pseudo Amino Acid Composition). Total 14 descriptors in 6 categories. The package is collaboratively developed by Computational Biology and Drug Design Group, Central South University.

Keywords: R, protr, protein sequence, amino acid, feature extraction.

1. Introduction

The protr package (Xiao et al. 2012) presented in this article implemented most of the state-of-the-art protein sequence feature extraction methods with R. Several self-explanatory examples have been included to illustrate the benefits of using protr. Many more examples are available within the package.

The protr package is available from the Comprehensive R Archive Network (CRAN) now, visit http://CRAN.R-project.org/package=protr for more details. This vignette corresponds to protr version 0.1-0 and was typeset on 2012-11-19.

Generally, each type of the descriptors(features) could be extracted with a function named extractX() in the protr package, where X stands for the abbreviation of the descriptor name. The descriptors and the function names implemented are listed below:

- Amino Acid Composition
  - extractAAC() - Amino Acid Composition
  - extractDC() - Dipeptide Composition
  - extractTC() - Tripeptide Composition

- Autocorrelation
In the next section, we’ll introduce these descriptors and the function usages in this order.

## 2. Protein Sequence Descriptors in protr

A protein or peptide sequence with $N$ amino acid residues could be generally represented as $\{R_1, R_2, \ldots, R_n\}$, where $R_i$ represents the residue at the $i$-th position in the sequence. The labels $i$ and $j$ are used to index amino acid position in a sequence, and $r$, $s$, $t$ are used to represent the amino acid type. The computed features are roughly divided into 4 groups according to their known applications described in the literature.

A protein sequence could be divided equally into segments and the methods, described as follows for the global sequence, could be applied to each segment.

### 2.1. Amino Acid Composition (AAC)

The Amino Acid Composition (AAC) is the fraction of each amino acid type within a protein. The fractions of all 20 natural amino acids are calculated as:

$$f(r) = \frac{N_r}{N} \quad r = 1, 2, \ldots, 20.$$  

where $N_r$ is the number of the amino acid type $r$ and $N$ is the length of the sequence.

As was described above, we could use the function `extractAAC()` to extract the features from protein sequences:
Xiao Nan, Dongsheng Cao

> require(protr)
> x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
> extractAAC(x)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>R</th>
<th>N</th>
<th>D</th>
<th>C</th>
<th>E</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.06405694</td>
<td>0.07117438</td>
<td>0.03914591</td>
<td>0.05160142</td>
<td>0.06761566</td>
<td>0.04804270</td>
<td>0.04804270</td>
</tr>
<tr>
<td>G</td>
<td>0.08185053</td>
<td>0.03024911</td>
<td>0.03558719</td>
<td>0.07651246</td>
<td>0.03914591</td>
<td>0.01245552</td>
<td>0.03202847</td>
</tr>
<tr>
<td>P</td>
<td>0.05338078</td>
<td>0.08896797</td>
<td>0.04448399</td>
<td>0.02313167</td>
<td>0.04270463</td>
<td>0.04982206</td>
<td></td>
</tr>
</tbody>
</table>

Here with the function `readFASTA()` we loaded a single protein sequence (P00750, Tissue-type plasminogen activator) from a FASTA format file. Then extracted the AAC descriptors with `extractAAC()`. The result returned is a named vector, whose elements are tagged with the name of each amino acid.

2.2. Dipeptide Composition (DC)

The Dipeptide Composition (DC) gives 400 features, defined as:

\[ f(r, s) = \frac{N_{rs}}{N - 1} \quad r, s = 1, 2, \ldots, 20. \]

where \( N_{rs} \) is the number of dipeptide represented by amino acid type \( r \) and type \( s \). Similar to `extractAAC()`, here we use `extractDC()` to compute the features:

> dc = extractDC(x)
> head(dc, n = 30L)

<table>
<thead>
<tr>
<th></th>
<th>AA</th>
<th>RA</th>
<th>NA</th>
<th>DA</th>
<th>CA</th>
<th>EA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.003565062</td>
<td>0.003565062</td>
<td>0.000000000</td>
<td>0.007130125</td>
<td>0.003565062</td>
<td>0.003565062</td>
</tr>
<tr>
<td>QA</td>
<td>0.007130125</td>
<td>0.007130125</td>
<td>0.001782531</td>
<td>0.003565062</td>
<td>0.001782531</td>
<td>0.001782531</td>
</tr>
<tr>
<td>MA</td>
<td>0.000000000</td>
<td>0.005347594</td>
<td>0.003565062</td>
<td>0.007130125</td>
<td>0.003565062</td>
<td>0.000000000</td>
</tr>
<tr>
<td>YA</td>
<td>0.000000000</td>
<td>0.000000000</td>
<td>0.003565062</td>
<td>0.007130125</td>
<td>0.005347594</td>
<td>0.001782531</td>
</tr>
<tr>
<td>CR</td>
<td>0.005347594</td>
<td>0.005347594</td>
<td>0.000000000</td>
<td>0.007130125</td>
<td>0.001782531</td>
<td>0.003565062</td>
</tr>
</tbody>
</table>

Here we only showed the first 30 elements of the result vector and omitted the rest of the result. The element names of the returned vector are self-explanatory as before.

2.3. Tripeptide Composition (TC)

The Tripeptide Composition (TC) gives 8000 features, defined as:

\[ f(r, s, t) = \frac{N_{rst}}{N - 2} \quad r, s, t = 1, 2, \ldots, 20 \]
where \( N_{rst} \) is the number of tripeptides represented by amino acid type \( r \), \( s \) and \( t \). With function \( \text{extractTC()} \), we could easily obtain the length-8000 descriptor, to save some space, here we also omitted the tedious outputs:

\[
> \text{tc = extractTC(x)} \\
> \text{head(tc, n = 36L)}
\]

\[
\begin{array}{cccccccc}
\text{AAA} & \text{RAA} & \text{NAA} & \text{DAA} & \text{CAA} & \text{EAA} \\
0.000000000 & 0.000000000 & 0.000000000 & 0.000000000 & 0.000000000 & 0.000000000 \\
\text{QAA} & \text{GAA} & \text{HAA} & \text{IAA} & \text{LAA} & \text{KAA} \\
0.001785714 & 0.000000000 & 0.000000000 & 0.000000000 & 0.000000000 & 0.000000000 \\
\text{MAA} & \text{FAA} & \text{PAA} & \text{SAA} & \text{TAA} & \text{WAA} \\
0.000000000 & 0.000000000 & 0.000000000 & 0.000000000 & 0.000000000 & 0.000000000 \\
\text{YAA} & \text{VAA} & \text{ARA} & \text{RRA} & \text{NRA} & \text{DRA} \\
0.000000000 & 0.000000000 & 0.000000000 & 0.000000000 & 0.000000000 & 0.000000000 \\
\text{CRA} & \text{ERA} & \text{QRA} & \text{GRA} & \text{HRA} & \text{IRA} \\
0.000000000 & 0.000000000 & 0.000000000 & 0.000000000 & 0.000000000 & 0.000000000 \\
\text{LRA} & \text{KRA} & \text{MRA} & \text{FRA} & \text{PRA} & \text{SRA} \\
0.000000000 & 0.000000000 & 0.000000000 & 0.000000000 & 0.000000000 & 0.000000000
\end{array}
\]

### 2.4. Autocorrelation Descriptors

Autocorrelation descriptors are defined based on the distribution of amino acid properties along the sequence. The amino acid properties used here are various types of amino acids index (Retrieved from AAindex Database: [http://www.genome.jp/dbget/aaindex.html](http://www.genome.jp/dbget/aaindex.html), see Kawashima et al. (1999), Kawashima and Kanehisa (2000), and Kawashima et al. (2008), see Figure 1 for an illustrated example). Three types of autocorrelation descriptors are defined here and described below.

All the amino acid indices are centralized and standardized before the calculation, i.e.

\[
P_r = \frac{P_r - \bar{P}}{\sigma}
\]

where \( \bar{P} \) is the average of the property of the 20 amino acids:

\[
\bar{P} = \frac{\sum_{r=1}^{20} P_r}{20} \quad \text{and} \quad \sigma = \sqrt{\frac{1}{2} \sum_{r=1}^{20} (P_r - \bar{P})^2}
\]

**Normalized Moreau-Broto Autocorrelation Descriptors**

Moreau-Broto autocorrelation descriptors application to protein sequences could be defined as:

\[
AC(d) = \sum_{i=1}^{N-d} P_i P_{i+d} \quad d = 1, 2, \ldots, \text{nlag}
\]
Database: AAindex
Entry: ANDN920101
LinkDB: ANDN920101

H ANDN920101
D alpha-CH chemical shifts (Andersen et al., 1992)
R LIT: 1810048 PMID: 1575719
A Andersen, N.H., Cao, B. and Chen, C.
T Peptide/protein structure analysis using the chemical shift index method: upfield alpha-CH values reveal dynamic helices and aL sites
C BUNA790102 0.949

<table>
<thead>
<tr>
<th></th>
<th>A/L</th>
<th>R/K</th>
<th>N/M</th>
<th>D/F</th>
<th>C/P</th>
<th>Q/S</th>
<th>E/T</th>
<th>G/W</th>
<th>H/Y</th>
<th>I/V</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.35</td>
<td>4.38</td>
<td>4.75</td>
<td>4.76</td>
<td>4.65</td>
<td>4.37</td>
<td>4.29</td>
<td>3.97</td>
<td>4.63</td>
<td>3.95</td>
</tr>
<tr>
<td>2</td>
<td>4.17</td>
<td>4.36</td>
<td>4.52</td>
<td>4.66</td>
<td>4.44</td>
<td>4.35</td>
<td>4.70</td>
<td>4.60</td>
<td>3.95</td>
<td></td>
</tr>
</tbody>
</table>

DBGET integrated database retrieval system

Figure 1: An illustrated example in the AAIndex database

where $d$ is called the lag of the autocorrelation and $P_i$ and $P_{i+d}$ are the properties of the amino acids at position $i$ and $i + d$, respectively. nlag is the maximum value of the lag.

The normalized Moreau-Broto autocorrelation descriptors are defined as:

$$ATS(d) = \frac{AC(d)}{N - d} \quad d = 1, 2, \ldots, \text{nlag}$$

The corresponding function for this descriptor is `extractMoreauBroto()`. A typical call could be:

```r
> moreau = extractMoreauBroto(x)
> head(moreau, n = 36L)
CIDH920105.lag1 CIDH920105.lag2 CIDH920105.lag3 CIDH920105.lag4
0.081573213 -0.016064817 -0.015982990 -0.025739038
CIDH920105.lag5 CIDH920105.lag6 CIDH920105.lag7 CIDH920105.lag8
0.079058632 -0.042771564 -0.036320847 0.024087298
CIDH920105.lag9 CIDH920105.lag10 CIDH920105.lag11 CIDH920105.lag12
-0.005273958 0.052274763 0.082170073 0.005419919
CIDH920105.lag13 CIDH920105.lag14 CIDH920105.lag15 CIDH920105.lag16
0.083292042 0.004810584 0.001872446 -0.001531495
CIDH920105.lag17 CIDH920105.lag18 CIDH920105.lag19 CIDH920105.lag20
-0.011917230 0.071161551 0.033473197 0.026882737
CIDH920105.lag21 CIDH920105.lag22 CIDH920105.lag23 CIDH920105.lag24
0.073075402 0.115272790 0.041517897 -0.027025993
CIDH920105.lag25 CIDH920105.lag26 CIDH920105.lag27 CIDH920105.lag28
0.033477388 -0.003245255 0.078117010 -0.028177304
CIDH920105.lag29 CIDH920105.lag30 BHR880101.lag1 BHR880101.lag2
0.046695832 0.020584423 0.052740185 0.030804784
BHR880101.lag3 BHR880101.lag4 BHR880101.lag5 BHR880101.lag6
0.037170476 -0.058993771 0.070641780 -0.089192490
```

The 8 default properties used here are:
protr: Protein Sequence Feature Extraction with R

- AccNo. CIDH920105 — Normalized Average Hydrophobicity Scales
- AccNo. BHAR880101 — Average Flexibility Indices
- AccNo. CHAM820101 — Polarizability Parameter
- AccNo. CHAM820102 — Free Energy of Solution in Water, kcal/mole
- AccNo. CHOC760101 — Residue Accessible Surface Area in Tripeptide
- AccNo. BIGC670101 — Residue Volume
- AccNo. CHAM810101 — Steric Parameter
- AccNo. DAYM780201 — Relative Mutability

Users could change the property names of AAindex database with the argument `props`. The AAindex data shipped with `protr` could be loaded by `data(AAindex)`, which has the detailed information of each property. With the argument `customprops` and `nlag`, users could specify their own properties and lag value to calculate with. For illustration, we could use:

```r
> # Define 3 custom properties
> myprops = data.frame(AccNo = c("MyProp1", "MyProp2", "MyProp3"),
+   A = c(0.62, -0.5, 15), R = c(-2.53, 3, 101),
+   N = c(-0.78, 0.2, 58), D = c(-0.9, 3, 59),
+   C = c(0.29, -1, 47), E = c(-0.74, 3, 73),
+   Q = c(-0.85, 0.2, 72), G = c(0.48, 0, 1),
+   H = c(-0.4, -0.5, 82), I = c(1.38, -1.8, 57),
+   L = c(1.06, -1.8, 57), K = c(-1.5, 3, 73),
+   M = c(0.64, -1.3, 75), F = c(1.19, -2.5, 91),
+   P = c(0.12, 0, 42), S = c(-0.18, 0.3, 31),
+   T = c(-0.05, -0.4, 45), W = c(0.81, -3.4, 130),
+   Y = c(0.26, -2.3, 107), V = c(1.08, -1.5, 43))
> # Use 4 properties in the AAindex database, and 3 customized properties
> moreau2 = extractMoreauBroto(x, customprops = myprops,
+   props = c("CIDH920105", "BHAR880101",
+     "CHAM820101", "CHAM820102",
+     "MyProp1", "MyProp2", "MyProp3"))
> head(moreau2, n = 36L)

CIDH920105.lag1  CIDH920105.lag2  CIDH920105.lag3  CIDH920105.lag4
  0.081573213   -0.016064817  -0.015982990   -0.025739038
CIDH920105.lag5  CIDH920105.lag6  CIDH920105.lag7  CIDH920105.lag8
  0.079058632   -0.042771564  -0.036320847   -0.024087298
CIDH920105.lag9  CIDH920105.lag10 CIDH920105.lag11 CIDH920105.lag12
 -0.005273958   0.052274763   0.082170073   0.005419919
CIDH920105.lag13 CIDH920105.lag14 CIDH920105.lag15 CIDH920105.lag16
  0.083292042   0.004810584   0.001872446  -0.001531495
CIDH920105.lag17 CIDH920105.lag18 CIDH920105.lag19 CIDH920105.lag20
```
Xiao Nan, Dongsheng Cao

-0.011917230 0.071161551 0.033473197 0.026882737
CIDH920105.lag1 CIDH920105.lag2 CIDH920105.lag3 CIDH920105.lag4
0.073075402 0.115272790 0.041517897 -0.027025993
CIDH920105.lag5 CIDH920105.lag6 CIDH920105.lag7 CIDH920105.lag8
0.03477388 -0.003245255 0.078117010 -0.028177304
CIDH920105.lag9 CIDH920105.lag10 CIDH920105.lag11 CIDH920105.lag12
0.046695832 0.020584423 0.052740185 0.030804784
CIDH920105.lag13 CIDH920105.lag14 CIDH920105.lag15 CIDH920105.lag16

About the standard input format of props and customprops, see ?extractMoreauBroto for details.

**Moran Autocorrelation Descriptors**

Moran autocorrelation descriptors application to protein sequence may be defined as:

\[
I(d) = \frac{1}{N-d} \sum_{i=1}^{N-d} (P_i - \bar{P})(P_{i+d} - \bar{P}) \cdot \frac{1}{N} \sum_{i=1}^{N} (P_i - \bar{P})^2 \quad d = 1, 2, \ldots, 30
\]

where \(d\) and \(P_i\) and \(P_{i+d}\) are defined in the same way as in the first place, and \(\bar{P}\) is the considered property \(P\) along the sequence, i.e.,

\[
\bar{P} = \frac{1}{N} \sum_{i=1}^{N} P_i
\]

\(d, P, P_i\) and \(P_{i+d}\), nlag have the same meaning as above.

With extractMoran(), which has exactly the same arguments with extractMoreauBroto(), we could compute the Moran autocorrelation descriptors (only output the first 36 elements of the result):

```r
> # Use the 3 custom properties defined before
> # and 4 properties in the AAindex database
> moran = extractMoran(x, customprops = myprops,
+                      props = c('CIDH920105', 'BHAR880101',
+                           'CHAM820101', 'CHAM820102',
+                           'MyProp1', 'MyProp2', 'MyProp3'))
> head(moran, n = 36L)

CIDH920105.lag1 CIDH920105.lag2 CIDH920105.lag3 CIDH920105.lag4
0.062895724 -0.044827681 -0.045065117 -0.055955678
CIDH920105.lag5 CIDH920105.lag6 CIDH920105.lag7 CIDH920105.lag8
0.060586372 -0.074128412 -0.067308852 -0.001293384
CIDH920105.lag9 CIDH920105.lag10 CIDH920105.lag11 CIDH920105.lag12
-0.033747588 0.029392193 0.061789800 -0.023368437
CIDH920105.lag13 CIDH920105.lag14 CIDH920105.lag15 CIDH920105.lag16
```
**Geary Autocorrelation Descriptors**

Geary autocorrelation descriptors for protein sequence could be defined as:

\[
C(d) = \frac{1}{2(N-d)} \sum_{i=1}^{N-d} (P_i - P_{i+d})^2 - \frac{1}{N-1} \sum_{i=1}^{N} (P_i - \bar{P})^2
\]

\[
d = 1, 2, \ldots, 30
\]

where \(d, P_i, P_{i+d}\) and \(n\text{lag}\) have the same meaning as above.

For each amino acid index, there will be \(3 \times n\text{lag}\) autocorrelation descriptors. The usage of `extractGeary()` is exactly the same with `extractMoreauBroto()` and `extractMoran()`:

```r
> # Use the 3 custom properties defined before
> # and 4 properties in the AAindex database
> geary = extractGeary(x, customprops = myprops,
+     props = c('CIDH920105', 'BHAR880101',
+               'CHAM820101', 'CHAM820102',
+               'MyProp1', 'MyProp2', 'MyProp3'))
> head(geary, n = 36L)

            CIDH920105.lag1 CIDH920105.lag2 CIDH920105.lag3 CIDH920105.lag4
0.9361830  1.0442920  1.0452843  1.0563467
CIDH920105.lag5 CIDH920105.lag6 CIDH920105.lag7 CIDH920105.lag8
0.9406031  1.0765517  1.0675786  0.9991363
CIDH920105.lag9 CIDH920105.lag10 CIDH920105.lag11 CIDH920105.lag12
1.0316555  0.9684585  0.9353130  1.0201990
CIDH920105.lag13 CIDH920105.lag14 CIDH920105.lag15 CIDH920105.lag16
0.9340933  1.0207373  1.0251486  1.0290464
CIDH920105.lag17 CIDH920105.lag18 CIDH920105.lag19 CIDH920105.lag20
1.0414375  0.9494403  0.9905987  0.9987183
CIDH920105.lag21 CIDH920105.lag22 CIDH920105.lag23 CIDH920105.lag24
0.9472542  0.9010009  0.9828848  1.0574098
CIDH920105.lag25 CIDH920105.lag26 CIDH920105.lag27 CIDH920105.lag28
0.9897955  1.0290018  0.9400066  1.0584150
```
2.5. Composition / Transition / Distribution

These descriptors are developed and described by Dubchak et al. (1995) and Dubchak et al. (1999).

| Sequence | M | T | E | I | T | A | S | M | V | K | E | L | R | E | A | T | G | T | G | A |
| Transformation | 1 | 5 | 10 | 15 | 20 | 3 | 2 | 1 | 3 | 2 | 2 | 2 | 3 | 3 | 1 | 1 | 3 | 1 | 1 | 2 | 2 | 2 | 2 | 2 |
| Index for 1 |Transformation | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Index for 2 | | | | | | | | | | | | | | | | | | | | | | | | | |
| Index for 3 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1/2 Transitions | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1/3 Transitions | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2/3 Transitions | | | | | | | | | | | | | | | | | | | | | | | | | |

Figure 2: The sequence of a hypothetic protein indicating the construction of composition, transition and distribution descriptors of a protein. Sequence index indicates the position of an amino acid in the sequence. The index for each type of amino acids in the sequence (‘1’, ‘2’ or ‘3’) indicates the position of the first, second, third, ... of that type of amino acid. 1/2 transition indicates the position of ‘12’ or ‘21’ pairs in the sequence (1/3 and 2/3 are defined in the same way.).

Step 1: Sequence Encoding

The amino acids are divided in three classes according to its attribute and each amino acid is encoded by one of the indices 1, 2, 3 according to which class it belonged. The attributes used here include hydrophobicity, normalized van der Waals volume polarity, and polarizability, as in the references. The corresponding division is in the table 1.

For example, for a given sequence “MTEITAAMVKELRESTGAGA”, it will be encoded as “3213222331131122222” according to its hydrophobicity division.

Step 2: Compute Composition, Transition and Distribution Descriptors

Three descriptors, Composition (C), Transition (T), and Distribution (D) were calculated for a given attribute as follows.

Composition

It is the global percent for each encoded class in the sequence. In the above example using hydrophobicity division, the numbers for encoded classes “1”, “2”, “3” are 5, 10, 5 respectively, so the compositions for them are 5/20 = 25%, 10/20 = 10%, and 5/20 = 25% respectively, where 20 is the length of the protein sequence. Composition can be defined as
Table 1: Amino acid attributes and the division of the amino acids into three groups for each attribute

<table>
<thead>
<tr>
<th></th>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hydrophobicity</td>
<td>Polar</td>
<td>Neutral</td>
<td>Hydrophobicity</td>
</tr>
<tr>
<td>Normalized van der</td>
<td>0.2-7.8</td>
<td>2.95-4.0</td>
<td>4.03-8.08</td>
</tr>
<tr>
<td>Polarity</td>
<td>4.9-6.2</td>
<td>8.0-9.2</td>
<td>10.4-13.0</td>
</tr>
<tr>
<td>Polarisability</td>
<td>0-1.08</td>
<td>0.128-0.186</td>
<td>0.219-0.409</td>
</tr>
<tr>
<td>Charge</td>
<td>Positive</td>
<td>Neutral</td>
<td>Negative</td>
</tr>
<tr>
<td>Secondary Structure</td>
<td>Helix</td>
<td>Strand</td>
<td>Coil</td>
</tr>
<tr>
<td>Solvent Accessibility</td>
<td>Buried</td>
<td>Exposed</td>
<td>Intermediate</td>
</tr>
</tbody>
</table>

\[ C_r = \frac{n_r}{n} \quad r = 1, 2, 3 \]

where \( n_r \) is the number of amino acid type \( r \) in the encoded sequence and \( N \) is the length of the sequence. An example for `extractCTDC()` could be:

\[ > \text{extractCTDC(x)} \]

```
0.29715302 0.40569395 0.29715302 0.45195730 0.29715302 0.2508968 0.33985765
0.33274021 0.32740214 0.33096085 0.41814947 0.26088968 0.11032028 0.79903559
0.09964413 0.38967972 0.29537367 0.31494662 0.43060498 0.29715302 0.27224199
```

The result shows the elements whose names are `PropertyNumber.GroupNumber` in the returned vector.

**Transition**

A transition from class 1 to 2 is the percent frequency with which 1 is followed by 2 or 2 is followed by 1 in the encoded sequence. Transition descriptor can be calculated as

\[ T_{rs} = \frac{n_{rs} + n_{sr}}{N - 1} \quad rs = \{12, 13, 23\} \]

where \( n_{rs}, n_{sr} \) is the numbers of dipeptide encoded as “rs” and “sr” respectively in the sequence and \( N \) is the length of the sequence. An example for `extractCTDT()` could be:

\[ > \text{extractCTDT(x)} \]
> extractCTDT(x)

```
 0.27094474  0.16042781  0.23551159  0.26737968  0.22638146  0.1712299
 0.21033868  0.20499109  0.23707665  0.27272727  0.15151515  0.24598930
 0.18181818  0.02139037  0.15686275  0.27272727  0.22816399  0.15864528
prop7.Tr1221  prop7.Tr1331  prop7.Tr2332
 0.25133690  0.21568627  0.1803565
```

**Distribution**

The “distribution” descriptor describes the distribution of each attribute in the sequence. There are five “distribution” descriptors for each attribute and they are the position percents in the whole sequence for the first residue, 25% residues, 50% residues, 75% residues and 100% residues, respectively, for a specified encoded class. For example, there are 10 residues encoded as “2” in the above example, the positions for the first residue “2”, the 2th residue “2” (25%*10=2), the 5th “2” residue (50%*10=5), the 7th “2” (75%*10=7) and the 10th residue “2” (100%*10) in the encoded sequence are 2, 5, 15, 17, 20 respectively, so the distribution descriptors for “2” are: 10.0 (2/20*100), 25.0 (5/20*100), 75.0 (15/20*100), 85.0 (17/20*100), 100.0 (20/20*100), respectively.

Finally, an example for `extractCTDD()` could be:

> extractCTDD(x)

```
 0.3558719  23.1316726  50.1779359  73.8434164
 99.8220641  0.5338078  27.4021352  47.3309609
 75.2669039  100.0000000  0.1779359  19.5729537
 51.7793594  75.6227758  99.6441281  0.3558719
 25.6227758  48.0427046  75.4448399  100.0000000
 1.4234875  23.3096085  54.4483986  76.3345196
 99.4661922  0.1779359  22.7758007  48.9323843
 69.5729537  99.8220641  0.1779359  20.9964413
 50.8896797  74.5551601  99.6441281  0.5338078
```
### Conjoint Triad Descriptors

Conjoint triad descriptors are proposed by Shen et al. (2007). These conjoint triad features abstracts the features of protein pairs based on the classification of amino acids. In this approach, each protein sequence is represented by a vector space consisting of features of amino acids. To reduce the dimensions of vector space, the 20 amino acids were clustered into several classes according to their dipoles and volumes of the side chains. The conjoint triad features are calculated as follows:

#### Step 1: Classification of Amino Acids

Electrostatic and hydrophobic interactions dominate protein-protein interactions. These two
kinds of interactions may be reflected by the dipoles and volumes of the side chains of amino acids, respectively. Accordingly, these two parameters were calculated, respectively, by using the density-functional theory method B3LYP/6-31G and molecular modeling approach. Based on the dipoles and volumes of the side chains, the 20 amino acids could be clustered into seven classes (See Table 2). Amino acids within the same class likely involve synonymous mutations because of their similar characteristics.

Table 2: Classification of amino acids based on dipoles and volumes of the side chains

<table>
<thead>
<tr>
<th>No.</th>
<th>Dipole Scale</th>
<th>Volume Scale</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>−</td>
<td>−</td>
<td>Ala, Gly, Val</td>
</tr>
<tr>
<td>2</td>
<td>−</td>
<td>+</td>
<td>Ile, Leu, Phe, Pro</td>
</tr>
<tr>
<td>3</td>
<td>+</td>
<td>+</td>
<td>Tyr, Met, Thr, Ser</td>
</tr>
<tr>
<td>4</td>
<td>++</td>
<td>+</td>
<td>His, Asn, Gln, Tpr</td>
</tr>
<tr>
<td>5</td>
<td>+ ++</td>
<td>+</td>
<td>Arg, Lys</td>
</tr>
<tr>
<td>6</td>
<td>+′ +′ +′</td>
<td>+</td>
<td>Asp, Glu</td>
</tr>
<tr>
<td>7</td>
<td>+′ ′ +′ ′ ′</td>
<td></td>
<td>Cys</td>
</tr>
</tbody>
</table>

Step 2: Conjoint Triad Calculation

The conjoint triad descriptors considered the properties of one amino acid and its vicinal amino acids and regarded any three continuous amino acids as a unit. Thus, the triads can be differentiated according to the classes of amino acids, i.e., triads composed by three amino acids belonging to the same classes, such as ART and VKS, could be treated identically. To conveniently represent a protein, we first use a binary space \((V, F)\) to represent a protein sequence. Here, \(V\) is the vector space of the sequence features, and each feature \(v_i\) represents a sort of triad type; \(F\) is the frequency vector corresponding to \(V\), and the value of the \(i\)-th dimension of \(F(f_i)\) is the frequency of type \(v_i\) appearing in the protein sequence. For the amino acids that have been categorized into seven classes, the size of \(V\) should be \(7 \times 7 \times 7\); thus \(i = 1, 2, \ldots, 343\). The detailed description for \((V, F)\) is illustrated in Figure 3.

Clearly, each protein correlates to the length (number of amino acids) of protein. In general, a long protein would have a large value of \(f_i\), which complicates the comparison between two heterogeneous proteins. Thus, we defined a new parameter, \(d_i\), by normalizing \(f_i\) with the following equation:

\[
d_i = \frac{f_i - \min\{f_1, f_2, \ldots, f_{343}\}}{\max\{f_1, f_2, \ldots, f_{343}\}}
\]

The numerical value of \(d_i\) of each protein ranges from 0 to 1, which thereby enables the comparison between proteins. Accordingly, we obtain another vector space (designated \(D\)) consisting of \(d_i\) to represent protein.

To compute conjoint triads of protein sequences, we could simply use:

1Dipole Scale (Debye): −, Dipole < 1.0; +, 1.0 < Dipole < 2.0; ++, 2.0 < Dipole < 3.0; + + +, Dipole > 3.0; +′ +′ +′, Dipole > 3.0 with opposite orientation.
2Volume Scale (Å³): −, Volume < 50; +, Volume > 50.
3Cys is separated from class 3 because of its ability to form disulfide bonds.
Protein Sequence Feature Extraction with $R = f(A, G, V)$, $G = f(I, L, F, P)$, $Y = f(Y, M, T, S)$, $H = f(H, N, Q, W)$, $K = f(R, K)$, $D = f(D, E)$, $C = v_1, v_4, v_7, v_{338}, v_{341}, v_2, v_5, v_8, v_{339}, v_{342}, v_3, v_6, v_9, v_{340}, v_{341}, v_{342}, v_{343}$

Figure 3: Schematic diagram for constructing the vector space $(V, F)$ of protein sequence. $V$ is the vector space of the sequence features; each feature ($v_i$) represents a triad composed of three consecutive amino acids; $F$ is the frequency vector corresponding to $V$, and the value of the $i$-th dimension of $F(f_i)$ is the frequency that $v_i$ triad appeared in the protein sequence.
> ctriad = extractCTriad(x)
> head(ctriad, n = 65L)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>0.1</th>
<th>0.3</th>
<th>0.6</th>
<th>0.2</th>
<th>0.4</th>
<th>0.0</th>
<th>0.3</th>
<th>1.0</th>
<th>0.6</th>
<th>0.5</th>
<th>0.0</th>
<th>0.2</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>VS111</td>
<td>VS211</td>
<td>VS311</td>
<td>VS411</td>
<td>VS511</td>
<td>VS611</td>
<td>VS711</td>
<td>VS121</td>
<td>VS221</td>
<td>VS321</td>
<td>0.2</td>
<td>0.4</td>
<td>0.5</td>
<td>0.2</td>
<td>0.3</td>
<td>0.3</td>
<td>0.1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.2</td>
<td>0.2</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>VS721</td>
<td>VS131</td>
<td>VS231</td>
<td>VS331</td>
<td>VS431</td>
<td>VS531</td>
<td>VS631</td>
<td>VS731</td>
<td>VS141</td>
<td>VS241</td>
<td>0.0</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.3</td>
<td>0.3</td>
<td>0.1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.2</td>
<td>0.2</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>VS441</td>
<td>VS541</td>
<td>VS641</td>
<td>VS741</td>
<td>VS151</td>
<td>VS251</td>
<td>VS351</td>
<td>VS451</td>
<td>VS551</td>
<td>VS651</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.5</td>
<td>0.1</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.1</td>
<td>0.4</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>VS561</td>
<td>VS661</td>
<td>VS761</td>
<td>VS171</td>
<td>VS271</td>
<td>VS371</td>
<td>VS471</td>
<td>VS571</td>
<td>VS671</td>
<td>VS771</td>
<td>0.2</td>
<td>0.0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.3</td>
<td>0.1</td>
<td>0.0</td>
<td>0.1</td>
<td>0.0</td>
<td>0.1</td>
<td>0.8</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>VS412</td>
<td>VS512</td>
<td>VS612</td>
<td>VS712</td>
<td>VS122</td>
<td>VS222</td>
<td>VS322</td>
<td>VS422</td>
<td>VS522</td>
<td>VS622</td>
<td>0.6</td>
<td>0.1</td>
<td>0.5</td>
<td>0.2</td>
<td>0.8</td>
<td>0.5</td>
<td>0.2</td>
<td>0.3</td>
<td>0.2</td>
<td>0.0</td>
<td>0.2</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

by which we only outputted the first 65 of total 343 dimension to save space.

### 2.7. Quasi-sequence-order Descriptors

The quasi-sequence-order descriptors are proposed by Chou (2000). They are derived from the distance matrix between the 20 amino acids.

#### Sequence-order-coupling Number

The $d$-th rank sequence-order-coupling number is defined as:

$$
\tau_d = \sum_{i=1}^{N-d} (d_{i,i+d})^2 \quad d = 1, 2, \ldots, \text{maxlag}
$$

where $d_{i,i+d}$ is the distance between the two amino acids at position $i$ and $i + d$.

**Note:** maxlag is the maximum lag and the length of the protein must be not less than maxlag.

The function `extractSOCN(x)` is used for computing the sequence-order-coupling numbers:

> extractSOCN(x)

<table>
<thead>
<tr>
<th>Schneider.lag1</th>
<th>Schneider.lag2</th>
<th>Schneider.lag3</th>
<th>Schneider.lag4</th>
<th>Schneider.lag5</th>
</tr>
</thead>
<tbody>
<tr>
<td>204.2036</td>
<td>199.8708</td>
<td>206.8102</td>
<td>197.4828</td>
<td>193.3366</td>
</tr>
<tr>
<td>Schneider.lag6</td>
<td>Schneider.lag7</td>
<td>Schneider.lag8</td>
<td>Schneider.lag9</td>
<td>Schneider.lag10</td>
</tr>
<tr>
<td>208.1936</td>
<td>195.5476</td>
<td>200.9789</td>
<td>196.7110</td>
<td>193.9931</td>
</tr>
<tr>
<td>Schneider.lag11</td>
<td>Schneider.lag12</td>
<td>Schneider.lag13</td>
<td>Schneider.lag14</td>
<td>Schneider.lag15</td>
</tr>
<tr>
<td>199.7031</td>
<td>204.9389</td>
<td>187.0140</td>
<td>198.4702</td>
<td>205.4526</td>
</tr>
<tr>
<td>Schneider.lag16</td>
<td>Schneider.lag17</td>
<td>Schneider.lag18</td>
<td>Schneider.lag19</td>
<td>Schneider.lag20</td>
</tr>
<tr>
<td>193.1274</td>
<td>187.3529</td>
<td>190.4949</td>
<td>202.8853</td>
<td>198.5299</td>
</tr>
<tr>
<td>Schneider.lag21</td>
<td>Schneider.lag22</td>
<td>Schneider.lag23</td>
<td>Schneider.lag24</td>
<td>Schneider.lag25</td>
</tr>
<tr>
<td>191.1013</td>
<td>185.0074</td>
<td>189.9857</td>
<td>202.7113</td>
<td>201.6267</td>
</tr>
<tr>
<td>Schneider.lag26</td>
<td>Schneider.lag27</td>
<td>Schneider.lag28</td>
<td>Schneider.lag29</td>
<td>Schneider.lag30</td>
</tr>
</tbody>
</table>

Protein Sequence Feature Extraction with R

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>194.5770</td>
<td>185.9939</td>
<td>204.1297</td>
<td>191.1629</td>
<td>183.9073</td>
</tr>
<tr>
<td>Grantham.lag1</td>
<td>Grantham.lag2</td>
<td>Grantham.lag3</td>
<td>Grantham.lag4</td>
<td>Grantham.lag5</td>
</tr>
<tr>
<td>6674686.0000</td>
<td>6761609.0000</td>
<td>7138892.0000</td>
<td>6748261.0000</td>
<td>6291229.0000</td>
</tr>
<tr>
<td>Grantham.lag6</td>
<td>Grantham.lag7</td>
<td>Grantham.lag8</td>
<td>Grantham.lag9</td>
<td>Grantham.lag10</td>
</tr>
<tr>
<td>6839853.0000</td>
<td>6594164.0000</td>
<td>6556148.0000</td>
<td>6620183.0000</td>
<td>6770614.0000</td>
</tr>
<tr>
<td>Grantham.lag11</td>
<td>Grantham.lag12</td>
<td>Grantham.lag13</td>
<td>Grantham.lag14</td>
<td>Grantham.lag15</td>
</tr>
<tr>
<td>6495689.0000</td>
<td>6865537.0000</td>
<td>6297267.0000</td>
<td>6498247.0000</td>
<td>6615566.0000</td>
</tr>
<tr>
<td>Grantham.lag16</td>
<td>Grantham.lag17</td>
<td>Grantham.lag18</td>
<td>Grantham.lag19</td>
<td>Grantham.lag20</td>
</tr>
<tr>
<td>6572680.0000</td>
<td>6569081.0000</td>
<td>6173947.0000</td>
<td>6570829.0000</td>
<td>6471308.0000</td>
</tr>
<tr>
<td>Grantham.lag21</td>
<td>Grantham.lag22</td>
<td>Grantham.lag23</td>
<td>Grantham.lag24</td>
<td>Grantham.lag25</td>
</tr>
<tr>
<td>6461649.0000</td>
<td>5939432.0000</td>
<td>6532121.0000</td>
<td>6652472.0000</td>
<td>6480660.0000</td>
</tr>
<tr>
<td>Grantham.lag26</td>
<td>Grantham.lag27</td>
<td>Grantham.lag28</td>
<td>Grantham.lag29</td>
<td>Grantham.lag30</td>
</tr>
<tr>
<td>6382281.0000</td>
<td>6276521.0000</td>
<td>6537634.0000</td>
<td>6442991.0000</td>
<td>6350157.0000</td>
</tr>
</tbody>
</table>

Users could also specify the maximum lag value with the nlag argument.

Note: In addition to Schneider-Wrede physicochemical distance matrix (Schneider and Wrede 1994) used by Kuo-Chen Chou, another chemical distance matrix by Grantham (1974) is also used here. So the feature dimension will be nlag * 2. The quasi-sequence-order descriptors described next also utilized the two matrices.

Quasi-sequence-order Descriptors

For each amino acid type, a quasi-sequence-order descriptor can be defined as:

\[ X_r = \frac{f_r}{\sum_{r=1}^{20} f_r + w \sum_{d=1}^{\text{maxlag}} \tau_d} \quad r = 1, 2, \ldots, 20 \]

where \( f_r \) is the normalized occurrence for amino acid type \( i \) and \( w \) is a weighting factor (\( w = 0.1 \)). These are the first 20 quasi-sequence-order descriptors. The other 30 quasi-sequence-order are defined as:

\[ X_d = \frac{w \tau_{d-20}}{\sum_{r=1}^{20} f_r + w \sum_{d=1}^{\text{maxlag}} \tau_d} \quad d = 21, 22, \ldots, 20 + \text{maxlag} \]

An minimal example for extractQSO() could be:

```r
> extractQSO(x)

Schneider.Xr.A Schneider.Xr.R Schneider.Xr.N Schneider.Xr.D Schneider.Xr.C
  6.096218e-02   6.773576e-02   3.725467e-02   4.910846e-02   6.434897e-02
Schneider.Xr.E Schneider.Xr.Q Schneider.Xr.G Schneider.Xr.H Schneider.Xr.I
  4.572164e-02   4.572164e-02   7.789612e-02   2.878770e-02   3.386788e-02
Schneider.Xr.L Schneider.Xr.K Schneider.Xr.M Schneider.Xr.F Schneider.Xr.P
  7.281594e-02   3.725467e-02   1.853767e-02   3.048456e-02   5.080182e-02
Schneider.Xr.S Schneider.Xr.T Schneider.Xr.W Schneider.Xr.Y Schneider.Xr.V
  8.466970e-02   4.233485e-02   2.201412e-02   4.064145e-02   4.741503e-02
```
Figure 4: A schematic drawing to show (a) the 1st-rank, (b) the 2nd-rank, and (c) the 3rd-rank sequence-order-coupling mode along a protein sequence. (a) Reflects the coupling mode between all the most contiguous residues, (b) that between all the 2nd most contiguous residues, and (c) that between all the 3rd most contiguous residues. This figure is from Chou (2000).
where users could also specify the maximum lag with argument \texttt{nlag} and the weighting factor with argument \texttt{w}.

\subsection{2.8. Pseudo-Amino Acid Composition (PAAC)}

This group of descriptors are proposed in Chou (2001). PAAC descriptors (\url{http://www.csbio.sjtu.edu.cn/bioinf/PseAAC/type1.htm}) are also called the type 1 pseudo-amino acid composition. Let $H_1^o(i)$, $H_2^o(i)$, $M^o(i)$ ($i = 1, 2, 3, \ldots, 20$) be the original hydrophobicity values, the original hydrophilicity values and the original side chain masses of the 20 natural amino acids, respectively. They are converted to following qualities by a standard conversion:

$$H_1(i) = \frac{H_1^o(i) - \frac{1}{20} \sum_{i=1}^{20} H_1^o(i)}{\sqrt{\sum_{i=1}^{20} [H_1^o(i) - \frac{1}{20} \sum_{i=1}^{20} H_1^o(i)]^2}}$$

$H_2^o(i)$ and $M^o(i)$ are normalized as $H_2(i)$ and $M(i)$ in the same way.

![Figure 5](http://example.com/example.png)

Figure 5: A schematic drawing to show (a) the first-tier, (b) the second-tier, and (c) the third-tier sequence order correlation mode along a protein sequence. Panel (a) reflects the correlation mode between all the most contiguous residues, panel (b) that between all the second-most contiguous residues, and panel (c) that between all the third-most contiguous residues. This figure is from Chou (2001).
Then, a correlation function could be defined as

$$\Theta(R_i, R_j) = \frac{1}{3} \left\{ [H_1(R_i) - H_1(R_j)]^2 + [H_2(R_i) - H_2(R_j)]^2 + [M(R_i) - M(R_j)]^2 \right\}$$

This correlation function is actually an averaged value for the three amino acid properties: hydrophobicity value, hydrophilicity value and side chain mass. Therefore we can extend this definition of correlation function for one amino acid property or for a set of n amino acid properties.

For one amino acid property, the correlation can be defined as:

$$\Theta(R_i, R_j) = [H_1(R_i) - H_1(R_j)]^2$$

where \(H(R_i)\) is the amino acid property of amino acid \(R_i\) after standardization.

For a set of n amino acid properties, it can be defined as: where \(H_k(R_i)\) is the k-th property in the amino acid property set for amino acid \(R_i\).

$$\Theta(R_i, R_j) = \frac{1}{n} \sum_{k=1}^{n} [H_k(R_i) - H_k(R_j)]^2$$

where \(H_k(R_i)\) is the k-th property in the amino acid property set for amino acid \(R_i\).

A set of descriptors called sequence order-correlated factors are defined as:

$$\theta_1 = \frac{1}{N-1} \sum_{i=1}^{N-1} \Theta(R_i, R_{i+1})$$

$$\theta_2 = \frac{1}{N-2} \sum_{i=1}^{N-2} \Theta(R_i, R_{i+2})$$

$$\theta_3 = \frac{1}{N-3} \sum_{i=1}^{N-3} \Theta(R_i, R_{i+3})$$

$$\ldots$$

$$\theta_\lambda = \frac{1}{N-\lambda} \sum_{i=1}^{N-\lambda} \Theta(R_i, R_{i+\lambda})$$

\(\lambda (\lambda < L)\) is a parameter to be chosen. Let \(f_i\) be the normalized occurrence frequency of the 20 amino acids in the protein sequence, a set of \(20 + \lambda\) descriptors called the pseudo-amino acid composition for a protein sequence can be defined as:

$$X_c = \frac{f_c}{\sum_{r=1}^{20} f_r + w \sum_{j=1}^{\lambda} \theta_j} \quad (1 < c < 20)$$

$$X_c = \frac{w \theta_{c-20}}{\sum_{r=1}^{20} f_r + w \sum_{j=1}^{\lambda} \theta_j} \quad (21 < c < 20 + \lambda)$$
where $w$ is the weighting factor for the sequence-order effect and is set as $w = 0.05$ in protr as suggested by Kuo-Chen Chou.

With `extractPAAC()`, we could compute the PAAC descriptors:

```r
> extractPAAC(x)

9.07025432 10.07806035 5.54293319 7.30659376 9.57415734
6.80269074 6.80269074 11.58976941 4.28317565 5.03903018
10.83391488 5.54293319 1.76366056 4.53512716 7.5584527
12.59757544 6.29878772 3.27536961 6.04683621 7.05464225
Xc2.lambda.1  Xc2.lambda.2  Xc2.lambda.3  Xc2.lambda.4  Xc2.lambda.5
0.02514092  0.02500357  0.02527773  0.02553159  0.02445265
Xc2.lambda.6  Xc2.lambda.7  Xc2.lambda.8  Xc2.lambda.9  Xc2.lambda.10
0.02561910  0.02486131  0.02506656  0.02553952  0.02437663
Xc2.lambda.11  Xc2.lambda.12  Xc2.lambda.13  Xc2.lambda.14  Xc2.lambda.15
0.02491262  0.02533803  0.02351915  0.02479912  0.02548431
Xc2.lambda.16  Xc2.lambda.17  Xc2.lambda.18  Xc2.lambda.19  Xc2.lambda.20
0.02478210  0.02513770  0.02457224  0.02543046  0.02530089
Xc2.lambda.21  Xc2.lambda.22  Xc2.lambda.23  Xc2.lambda.24  Xc2.lambda.25
0.02476967  0.02342389  0.02431684  0.02610300  0.02626722
Xc2.lambda.26  Xc2.lambda.27  Xc2.lambda.28  Xc2.lambda.29  Xc2.lambda.30
0.02457082  0.02343049  0.02588823  0.02490463  0.02451951
```

The `extractPAAC()` function also provides the `props` and `customprops` arguments, which is similar to the functions for Moreau-Broto/Moran/Geary autocorrelation descriptors. For minor differences, see `?extractPAAC`. Users could specify the lambda parameter and the weighting factor with arguments `lambda` and `w`.

**Note:** In the work of Kuo-Chen Chou, the definition for “normalized occurrence frequency” was not given. In this work, we define it as the occurrence frequency of amino acid in the sequence normalized to 100% and hence our calculated values are not the same as values by them.

### 2.9. Amphiphilic Pseudo-Amino Acid Composition (APAAC)

Amphiphilic Pseudo-Amino Acid Composition (APAAC, [http://www.csbio.sjtu.edu.cn/bioinf/PseAAC/type2.htm](http://www.csbio.sjtu.edu.cn/bioinf/PseAAC/type2.htm)) was proposed in Chou (2001). APAAC is also recognized as the type 2 pseudo-amino acid composition. The definitions of these qualities are similar to the PAAC descriptors. From $H_1(i)$ and $H_2(j)$ defined before, the hydrophobicity and hydrophilicity correlation functions are defined respectively as:

$$H_{ij}^1 = H_1(i)H_1(j)$$

$$H_{ij}^2 = H_2(i)H_2(j)$$
From these qualities, sequence order factors can be defined as:

\[
\tau_1 = \frac{1}{N - 1} \sum_{i=1}^{N-1} H_{i,i+1}^1
\]
\[
\tau_2 = \frac{1}{N - 1} \sum_{i=1}^{N-1} H_{i,i+1}^2
\]
\[
\tau_3 = \frac{1}{N - 2} \sum_{i=1}^{N-2} H_{i,i+2}^1
\]
\[
\tau_4 = \frac{1}{N - 2} \sum_{i=1}^{N-2} H_{i,i+2}^2
\]
\[\ldots\]
\[
\tau_{2\lambda-1} = \frac{1}{N - \lambda} \sum_{i=1}^{N-\lambda} H_{i,i+\lambda}^1
\]
\[
\tau_{2\lambda} = \frac{1}{N - \lambda} \sum_{i=1}^{N-\lambda} H_{i,i+\lambda}^2
\]

Then a set of descriptors called *Amphiphilic Pseudo-Amino Acid Composition (APAAC)* are defined as:

\[
P_c = \frac{f_c}{\sum_{r=1}^{20} f_r + w \sum_{j=1}^{2\lambda} \tau_j} \quad (1 < c < 20)
\]
\[
P_c = \frac{w \tau_u}{\sum_{r=1}^{20} f_r + w \sum_{j=1}^{2\lambda} \tau_j} \quad (21 < u < 20 + 2\lambda)
\]

where \( w \) is the weighting factor and is taken as \( w = 0.5 \) in *protr* as in the work of Chou KC.

A minimal example for *extracAPAAC()* is:

\[
\begin{align*}
\text{Pc1.A} & & \text{Pc1.R} & & \text{Pc1.N} \\
3.537412e+01 & & 3.930458e+01 & & 2.161752e+01 \\
\text{Pc1.D} & & \text{Pc1.C} & & \text{Pc1.E} \\
2.849582e+01 & & 3.733935e+01 & & 2.653059e+01 \\
\text{Pc1.Q} & & \text{Pc1.G} & & \text{Pc1.H} \\
2.653059e+01 & & 4.520027e+01 & & 1.670445e+01 \\
\text{Pc1.I} & & \text{Pc1.L} & & \text{Pc1.K} \\
1.965229e+01 & & 4.225242e+01 & & 2.161752e+01 \\
\text{Pc1.M} & & \text{Pc1.F} & & \text{Pc1.P} \\
6.878302e+00 & & 1.768706e+01 & & 2.947844e+01 \\
\text{Pc1.S} & & \text{Pc1.T} & & \text{Pc1.W}
\end{align*}
\]
Figure 6: A schematic diagram to show (a1/a2) the first-rank, (b1/b2) the second-rank and (c1/c2) the third-rank sequence-order-coupling mode along a protein sequence through a hydrophobicity/hydrophilicity correlation function, where $H_{i,j}^{1}$ and $H_{i,j}^{2}$ are given by Equation (3). Panel (a1/a2) reflects the coupling mode between all the most contiguous residues, panel (b1/b2) that between all the second-most contiguous residues and panel (c1/c2) that between all the third-most contiguous residues. This figure is from Chou (2005).
This function has the same arguments as `extractPAAC()`.
3. Miscellaneous Tools

In this section, we will briefly introduce some useful tools provided by the protr package.

3.1. Retrieve Protein Sequences from UniProt

This function getUniProt() gets protein sequences from uniprot.org by protein ID(s). The input ID is a character vector specifying the protein ID(s). The returned sequences are stored in a list:

```r
> ids = c('P00750', 'P00751', 'P00752')
> prots = getUniProt(ids)
> print(prots)
[[1]]
[1] "MDAMKRGLCCVLGLCAVFSQEQIEAIHRFRGARSQYQICRDEKTQMIYQQHQLRWPVLRNSRNVEYCNW
SRAQCHSVPVKSCEPRCFNQGTCQYALFSDFVQCPEFGAKCCEIDTRATCQDIQGQSYRGTWSTAESGAECT
NWNSSALQKYPSQRRPDAI1RLGLGHNYCNPRDRDSKFPWIVCFKAGYFFSECPSTACSEGSNDQYENGASAYRG
HSLTSEGASCLFWNSMLIGKVYTAQPNPSAQALGLKHNYCRNPDGAKPWCHVLKRRDLTWDVYEDCPSTCCQLRQ
YSQPFQRGTLFAIDASHPQAAIFAKHHSPFRFQGGLQILSSCWILSAHCHFCQRFPFHPHVTILGRTRVVP
GEEEEKFVEEYKIVHKEFDDTDYNDIALLQLSDKSECCSQSVTVLTPPADLQLPDWTCEPSEGYKCHEALSP
FYSERLKEAHRVLVPSRRCSTQHLLNRTVDNMLCDGKSGPQANLHDACQGSGPLCVLNDGRMTLVGIISWG
LGCGQKDVPGVYKTVMQYDWDNRMP"
[[2]]
[1] "MGSLPQCLMPFILGLSGLGVTITTPWSLARPQGSCLSLEGVEIKGGSFRLLQEGFQALEYVCPGFGYPVPVQ
TRTCRSSTGSLSTLQTVNHKACRAIHCPRHPDFENEGYWPRTYPNYSDIEIFHCDGYTLSGANSRTQVNG
RWSQTAICDNGGNCSNPQPIPTKRGPSQRYLDSVYHCRCGRTLQGSRTRQECGGSWSGTQPSQDSFMYTD
PQEVEAEFLSLTETIEGVDADCHGHPGEQQKRVKIVALPSGSMNMYLVLDGSIDSAGSNFTGAKKCLKLINIEKVASY
GVKPRGTVLTVATPVMKVEADSSNAWTVKQNEINTEDHLKSTNATQLAQSVMSSWPVDPPEGWNR
RHIILMSTDGHLHSGPDITIVEDLLLYICDRKKNPREYLDVYFGVGLPNVQINALASDQKEHIFGBKVD
MENLEDVYQMDigesQSLSCGMWHRHKTDYHQPWAKISVRIPSKSGHSMEGAVSEFYVTLAACHFTVDDKE
HSLIKLGVGEKRLDEIEVLFPHYNIGNKQEGAIPEFYDYDWALIPKLKNKLYQTRICLPCTEGTTRARLLPP
TTTCQQQKEELPAPQDIKFALVFSEEEKKTKLTRKEVYIKNQDGKCSSRDAQYPGDKVKEISVTPRFLCTGVS
YPADPNCTCRDSGGPLVHKASRFQIFQGVSISVGVDCKVQKRMRSKQVPAMHDFINFLQLFWLKEKLQDELLGFL"
[[3]]
[1] "APPQISRIIIGRECEKHSPWHPQVAAHYSSFSCQGAVLBNPKWLTAHCHKNFRGYYWVLRGHNLFENNTAQF
FGVTADFPFHPGFNLSLKXHTKADKGDYSDHMMLRLRQLQSPAKITDAKVLKELPTEQELPGSTCEASWGSGIEPGB
FEPFPDEIQCVQTLTQNTFCABAHPBKVTEMLCAGLYPGKDKTCMDSGGPLINCGMWQGITSWHTFCGSANGPKS
IYTKLIFYLDWINDTENP"
```

3.2. Read FASTA format files

The readFASTA() function provides a convenient way to read protein sequences stored in FASTA format files. See ?readFASTA for details. The returned sequences are stored in a named list, whose components are named with the protein sequences’ names.
3.3. Check Protein Sequences

The `protcheck()` function checks if the protein sequence’s amino acid types are in the 20 default types, which returns a `TRUE` if all the amino acids in the sequence belongs to the 20 default types:

```r
> x = readFASTA(system.file('protseq/P00750.fasta', package = 'protr'))[[1]]
> # A real sequence
> protcheck(x)

[1] TRUE

> # An artificial sequence
> protcheck(paste(x, 'Z', sep = ''))

[1] FALSE
```

3.4. Protein Sequence Segmentation

The `protseg()` function extracts the segmentations from the protein sequence. Users could specify a sequence `x`, and a character `aa`, one of the 20 amino acid types, and a positive integer `k`, which controls the window size (half of the window).

This function returns a named list, each component contains one of the segmentations (a character string), names of the list components are the positions of the specified amino acid in the sequence. See the example below:

```r
> protseg(x, aa = 'M', k = 5)

$`48`
[1] "DEKTQMIYQQH"

$`242`
[1] "LPWNSMILIGK"

$`490`
[1] "TVTDNMLCAGD"

$`525`
[1] "LNDGRMTLVGI"
```
4. Summary

The descriptors’ information has been listed in table 3.

Table 3: List of protr computed features for protein sequences

<table>
<thead>
<tr>
<th>Feature Group</th>
<th>Feature Name</th>
<th>Feature Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amino Acid Composition</td>
<td>Amino Acid Composition</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Dipeptide Composition</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td>Tripeptide Composition</td>
<td>8000</td>
</tr>
<tr>
<td>Autocorrelation</td>
<td>Normalized Moreau-Broto Autocorrelation</td>
<td>240(^1)</td>
</tr>
<tr>
<td></td>
<td>Moran Autocorrelation</td>
<td>240(^1)</td>
</tr>
<tr>
<td></td>
<td>Geary Autocorrelation</td>
<td>240(^1)</td>
</tr>
<tr>
<td>CTD</td>
<td>Composition</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Transition</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Distribution</td>
<td>105</td>
</tr>
<tr>
<td>Conjoint Triad</td>
<td>Conjoint Triad</td>
<td>343</td>
</tr>
<tr>
<td>Quasi-sequence-order Descriptors</td>
<td>Sequence-order-coupling Number</td>
<td>60(^2)</td>
</tr>
<tr>
<td></td>
<td>Quasi-sequence-order Descriptors</td>
<td>100(^2)</td>
</tr>
<tr>
<td>Pseudo-Amino Acid Composition</td>
<td>Pseudo-Amino Acid Composition</td>
<td>50(^3)</td>
</tr>
<tr>
<td></td>
<td>Amphiphilic Pseudo-Amino Acid Composition</td>
<td>80(^4)</td>
</tr>
</tbody>
</table>

In this article, we discussed the functionality of the protr package, which is trying to offer a comprehensive and neat choice for protein sequence feature extraction.

Acknowledgments

We’d like to thank the Computational Biology and Drug Design Group (http://cbdd.csu.edu.cn/index/) of Central South University for supporting the development of this R package. Further suggestions and comments are welcomed.

References


\(^1\)The number depends on the choice of the number of properties of amino acids and the choice of the maximum values of the lag. The default is use 8 types of properties and lag = 30.

\(^2\)The number depends on the maximum value of lag. By default lag = 30. And two distance matrices were used, so the feature dimension is 30 × 2 = 60 and (20 + 30) × 2 = 100.

\(^3\)The number depends on the choice of the number of the set of amino acid properties and the choice of the \(\lambda\) value. The default is use 3 types of properties proposed by Kuo-Chen Chou and \(\lambda = 30\).

\(^4\)The number depends on the choice of the \(\lambda\) value. The default is that \(\lambda = 30\).


Affiliation:
Xiao Nan  
School of Math Science & Statistics  
Central South University  
Changsha, Hunan, P. R. China  
E-mail: road2stat@gmail.com  
URL: http://www.road2stat.com/

Dongsheng Cao  
School of Chemistry & Chemical Engineering  
Central South University  
Changsha, Hunan, P. R. China  
E-mail: oriental-cds@163.com  
URL: http://cbdd.csu.edu.cn/

Qingsong Xu  
School of Math Science & Statistics  
Central South University  
Changsha, Hunan, P. R. China  
E-mail: dasongxu@gmail.com

Yizeng Liang  
School of Chemistry & Chemical Engineering  
Central South University  
Changsha, Hunan, P. R. China  
E-mail: yizeng_liang@263.net