Package 'BaMORC'

January 3, 2019

Type Package

Title Bayesian Model Optimized Reference Correction Method for Assigned and Unassigned Protein NMR Spectra

Version 1.0.1

Depends R (>= 3.1.0)

Date 2019-1-2

Description Provides reference correction for protein NMR spectra. Bayesian Model Optimized Reference Correction (BaMORC) is utilizing Bayesian probabilistic framework to perform protein NMR referencing correction, currently for alpha and beta carbon-13 chemical shifts, without any resonance assignment and/or three-dimensional protein structure. For more detailed explanation, please refer to the paper ``Automatic 13C Chemical Shift Reference Correction for Unassigned Protein NMR Spectra" <https://rdcu.be/4ly5> (Journal of Biomolecular NMR, Aug 2018)" <doi:10.1007/s10858-018-0202-5>.

URL https://github.com/MoseleyBioinformaticsLab/BaMORC

BugReports https://github.com/MoseleyBioinformaticsLab/BaMORC/issues

License BSD_3_clause + file LICENSE

Encoding UTF-8

LazyData true

Imports data.table, tidyr, DEoptim, httr, docopt, stringr, jsonlite, readr, devtools, RBMRB, BMRBr

RoxygenNote 6.1.1

Suggests knitr, rmarkdown, formatR

VignetteBuilder knitr

NeedsCompilation no

Author Xi Chen [aut, cre] (<https://orcid.org/0000-0001-7094-6748>), Andrey Smelter [aut] (<https://orcid.org/0000-0003-3056-9225>), Hunter Moseley [aut] (<https://orcid.org/0000-0003-3995-5368>)

Maintainer Xi Chen <billchenxi@gmail.com>

Repository CRAN

Date/Publication 2019-01-02 23:40:03 UTC

R topics documented:

`AA57OLMatrix` *Overlapping Matrix for 57 amino acid and secondary structure combinations.*

Description

A dataset containing the pre-calculated overlapping matrix 57 amino acid and secondary structure combinations:

Usage

```
AA57OLMatrix
```

Format

A matrix with 57 by 57 elements.

Details

- column names or row names.
  - First letter. amino acid typings in single-letter format, not include glycine ("A", "C", "D", "E", "F", "H", "I", "K", "L", "M", "N", "P", "Q", "R", "S", "T", "V", "W", "Y")
  - second letter. secondary structure in single-letter format ("B", "H", "C")

`AA57OLWeights` *Overlapping weights for 57 amino acid and secondary structure combinations.*

Description

A vector containing the pre-calculated overlapping weights 57 amino acid and secondary structure combinations:

Usage

```
AA57OLWeights
```

Format

A vector of 57 elements.

Details

- names.
  - First letter. amino acid typings in single-letter format, not include glycine ("A", "C", "D", "E", "F", "H", "I", "K", "L", "M", "N", "P", "Q", "R", "S", "T", "V", "W", "Y")
  - second letter. secondary structure in single-letter format ("B", "H", "C")

aaCodes1Letter19 *Single-letter amino acid naming convention.*

Description

A vector containing amino acid names with all letters capitalized, (no glycine), they are: "A", "C", "D", "E", "F", "H", "I", "K", "L", "M", "N", "P", "Q", "R", "S", "T", "V", "W", "Y".

Usage

```
aaCodes1Letter19
```

Format

A vector of 19 elements.

aaCodes1Letter20 *Single-letter amino acid naming convention.*

Description

A vector containing amino acid names with all letters capitalized, (no glycine, but includes the oxidized cystine), they are: "A", "B", "C", "D", "E", "F", "H", "I", "K", "L", "M", "N", "P", "Q", "R", "S", "T", "V", "W", "Y".

Usage

```
aaCodes1Letter20
```

Format

A vector of 20 elements.

aaCodes3Letter1stCap *Three-letter amino acid naming convention with first letter capitalized.*

Description

A vector containing amino acid names with first letter capitalized (no glycine), they are: "Ala", "Cys", "Asp", "Glu", "Phe", "His", "Ile", "Lys", "Leu", "Met", "Asn", "Pro", "Gln", "Arg", "Ser", "Thr", "Val", "Trp", "Tyr".

Usage

```
aaCodes3Letter1stCap
```

Format

A vector of 19 elements.

aaCodes3LetterAllCap.v *Three-letter amino acid naming convention with all letters capitalized.*

Description

A vector containing amino acid names with all letters capitalized (no glycine), they are: "ALA", "CYS", "ASP", "GLU", "PHE", "HIS", "ILE", "LYS", "LEU", "MET", "ASN", "PRO", "GLN", "ARG", "SER", "THR", "VAL", "TRP", "TYR".

Usage

```
aaCodes3LetterAllCap.v
```

Format

A vector of 19 elements.

aaFreq — *Pre-defined amino acid frequency data.*

Description

Sample data for testing `calculate_mse()` function.

- AA_SS. amino acid typing and secondary structure naming combination (single-letter convention)
- Freq. relative cumulated frequency

Usage

```
aaFreq
```

Format

A data frame with 57 rows and 2 columns.

AvgCov.t — *Average covariance across three secondary structures for all amino acid typings (including oxidized cystine).*

Description

A dataset containing average covariance across three secondary structure for all amino acid typings (including oxidized cystine):

- AA. amino acid typing names
- AvgCov. average covariance value across three secondary structures

Usage

```
AvgCov.t
```

Format

A data frame with 20 rows and 2 columns.

bamorc *Calculates the referencing correction value.*

Description

The core package function that calculates the carbon-13 reference correction using an input protein sequence with associated secondary structure information along with a table of alpha and beta carbon chemical shift pairs. The output of this function is the correction value that should be added to the input carbon chemical shifts.

Usage

```
bamorc(sequence, secondary_structure = NULL, chemical_shifts_input,
  from = -5, to = 5)
```

Arguments

sequence	string of sequence
secondary_structure	string of secondary structure (optional)
chemical_shifts_input	table of alpha and beta carbon chemical shift pairs in data.frame
from	the lower bound of the optimization search window
to	the upper bound of the optimization search window

Value

Carbon-13 reference correction value that should be applied (added) to the input carbon chemical shifts data.

Examples

```
sequence <- paste(RefDB_data$carbonDat[[1]]$AA,collapse = "")
secondary_structure <- paste(RefDB_data$carbonDat[[1]]$SS,collapse = "")
chemical_shifts_input <- RefDB_data$carbonDat[[1]][,c(4,5)]
from=-5
to=5

## Not run: bamorc(sequence, secondary_structure, chemical_shifts_input, from=-5, to=5)

# Expected output
# [1] 0.0142443


sequence <- paste(BaMORC::RefDB_data$carbonDat[[1]]$AA,collapse = "")
chemical_shifts_input <- BaMORC::RefDB_data$carbonDat[[1]][,c(4,5)]
from=-5
to=5
```

```
## Not run: bamorc(sequence=sequence, chemical_shifts_input=chemical_shifts_input, from=-5, to=5)
# Expected output
# [1] 0.009805279
```

`calculate_aa_prob` *Calculates an amino acid typing probability.*

Description

Function returns the probability (density) for a certain type of amino acid based on a chi-squared statistics with 2 degrees of freedom.

Usage

```
calculate_aa_prob(chi_squared_stat, df = 2)
```

Arguments

`chi_squared_stat`	a single or a vector of chi-squared statistics
`df`	degrees of freedom, default is 2

Value

Input can be a single value or a vector of values, the output will be probability density for each value.

Examples

```
# Find density for a chi square parameter with 3 degrees of freedom
calculate_aa_prob(0.314, df=3)
# Find density for a list of (chi square statistics) with 2 degrees of freedom
calculate_aa_prob(c(0.05, 0.1, 0.5), 2)
```

`calculate_chi_squared_stat`
Calculates a chi squared statistic(s).

Description

`calculate_chi_squared_stat` Given a pair of C_alpha and C_beta chemical shifts, this function will return a list of calculated chi squared statistics based on the combination of amino acid typings and secondary structures.

Usage

```
calculate_chi_squared_stat(cacb_pair)
```

Arguments

cacb_pair	A pair of carbon chemical shifts c(Ca, Cb)

Value

A list of chi-squared statistics basing the combination of amino acid typings and secondary structures.

Examples

```
calculate_chi_squared_stat(c(54,45))
```

---

calculate_mse *Calculates mean squared error*

Description

This function will return a mean squared error between estimated density of amino acid typing and secondary structure combination based on the given dataset and reference correction values for the alpha and beta carbons. The estimated amino acid typing density is based on the BaMORC method.

Usage

```
calculate_mse(step_ca, step_cb, dat_cacb, aa_Freq)
```

Arguments

step_ca	Potential correction value for alpha carbon.
step_cb	Potential correction value for beta carbon.
dat_cacb	Chemical shift data frame of alpha and beta carbons.
aa_Freq	Actual amino acid typing and secondary structure frequency calculated basing on provided protein sequence.

Value

Mean squared error.

Examples

```
# chemicalShifts and aaFreq are predefined sample variables for demo purpose.

calculate_mse(step_ca=1, step_cb=1, dat_cacb=chemicalShifts[, c(3,4)], aa_Freq=aaFreq)
```

calculate_rcf *Calculates the relative cumulative frequency for amino acid and secondary structure.*

Description

This function calculates the relative cumulative frequency of amino acid and secondary structure combination.

Usage

```
calculate_rcf(sequence, secondary_structure)
```

Arguments

sequence	String of protein sequence with one letter convention
secondary_structure	String of protein secondary structure with single letter convention

Value

Relative cumulative frequency.

Examples

```
sequence = paste(RefDB_data$carbonDat[[1]]$AA, collapse = "")
secondary_structure = paste(RefDB_data$carbonDat[[1]]$SS, collapse = "")
relativeCumulativeFrequency = calculate_rcf(sequence, secondary_structure)
```

CAMuTable *Mean chemical shift values for alpha carbon .*

Description

A dataset containing mean chemical shift values for alpha carbon for each amino acid across three secondary structure for all amino acid typings (including oxidized cystine and glycine):

- Residue. amino acid typing name (single-letter convention)
- C. coil
- H. helix
- B. beta strand
- A. average of three secondary structures.

Usage

```
CAMuTable
```

Format

A data frame with 21 rows and 5 columns.

`CarbonCov.t` *Covariance values of chemical shifts of alpha and beta carbons.*

Description

A dataset containing covariance values of chemical shifts of alpha and beta carbons for each amino acid across three secondary structure for all amino acid typings (including oxidized cystine but not glycine):

- AA.SS. amino acid typing and secondary structure combination (single-letter convention)
- Value. covariance value

Usage

```
CarbonCov.t
```

Format

A data frame with 79 rows and 2 columns.

`CASdTable` *Standard deviation of chemical shift values for alpha carbon .*

Description

A dataset containing Standard deviation of chemical shift values for alpha carbon for each amino acid across three secondary structure for all amino acid typings (including oxidized cystine and glycine):

- Residue. amino acid typing name (single-letter convention)
- C. coil
- H. helix
- B. beta strand
- A. average of three secondary structures.

Usage

```
CASdTable
```

Format

A data frame with 21 rows and 5 columns.

CBMuTable *Mean chemical shift values for beta carbon .*

Description

A dataset containing mean chemical shift values for beta carbon for each amino acid across three secondary structure for all amino acid typings (including oxidized cystine and glycine):

- Residue. amino acid typing name (single-letter convention)
- C. coil
- H. helix
- B. beta strand
- A. average of three secondary structures.

Usage

```
CBMuTable
```

Format

A data frame with 21 rows and 5 columns.

CBSdTable *Standard deviation of chemical shift values for beta carbon .*

Description

A dataset containing Standard deviation of chemical shift values for beta carbon for each amino acid across three secondary structure for all amino acid typings (including oxidized cystine and glycine):

- Residue. amino acid typing (single-letter convention)
- C. coil
- H. helix
- B. beta strand
- A. average of three secondary structures.

Usage

```
CBSdTable
```

Format

A data frame with 21 rows and 5 columns.

chemicalShifts *Pre-defined sample chemical shifts data.*

Description

Sample data for testing `calculate_mse()` function.

- AA. amino acid typing (single-letter convention)
- SS. secondary structure (single-letter convention)
- CA. chemical shift value of alpha carbon
- CB. chemical shift value of beta carbon

Usage

```
chemicalShifts
```

Format

A data frame with 55 rows and 4 columns.

cname *All the amino acids and secondary structures combinations for easy access.*

Description

A dataset containing all the amino acids and secondary structures combinations: A-H", "A-B", "A-C", "R-H", "R-B", "R-C", "N-H", "N-B", "N-C", "D-H", "D-B", "D-C", "B-H", "B-B", "B-C", "C-H", "C-B", "C-C", "Q-H", "Q-B", "Q-C", "E-H", "E-B", "E-C", "H-H", "H-B", "H-C", "I-H", "I-B", "I-C", "L-H", "L-B", "L-C", "K-H", "K-B", "K-C", "M-H", "M-B", "M-C", "F-H", "F-B", "F-C", "P-H", "P-B", "P-C", "S-H", "S-B", "S-C", "T-H", "T-B", "T-C", "Y-H", "Y-B", "Y-C", "W-H", "W-B", "W-C", "V-H", "V-B", "V-C".

Usage

```
cname
```

Format

A vector with 60 elements.

ID — *RefDB ID included in the BaMORC package.*

Description

A vector containing all the RefDB ID included in the BaMORC package.

Usage

```
ID
```

Format

A vector of 1557 elements.

inverseMatrices — *inverseMatrices.*

Description

A data set containing all the pre-calculated inverse matrices and the access functions.

$getInvMatrix(name) get the inverse matrix basing on the name, which is the amino acid and secondary structure combination (single-letter naming convention)

$getNames() get all the names of the inverse matrices

$Matrices the data set of all the pre-calculated inverse matrices

Usage

```
inverseMatrices
```

Format

An object of inverse matrices with their accessing functions.

jpred_fetcher *Using JPred Mass-submission scheduler program to submit protein sequence and return secondary structure results.*

Description

Using JPred Mass-submission scheduler program to submit protein sequence and return secondary structure results.

Usage

```
jpred_fetcher(protein_sequence)
```

Arguments

protein_sequence
: protein sequence

Value

protein secondary structure information

Examples

```
protein_sequence <- "MQVWPIEGIKKFETLSYLPPLTVEDLLKQI"
## Not run: secondary_structure <- jpred_fetcher(protein_sequence)
```

read_db_file `read_db_file()` *reads in data from existing database that included in the BaMORC package. This database was extracted from RefDB database.*

Description

`read_db_file()` reads in data from existing database that included in the BaMORC package. This database was extracted from RefDB database.

Usage

```
read_db_file(id)
```

Arguments

id
: BMRB or RefDB entry ID.

Value

Protein sequence, secondary structure information and chemical shifts dataframe.

Examples

```
id = 4022
head(read_db_file(id))
```

`read_nmrstar_file` *Extracts data from BMRB STAR 3.0 file.* `read_nmrstar_file()` *parses BMRB STAR 3.0 file. It will extract sequence information and chemical shifts for both alpha and beta carbons.*

Description

Extracts data from BMRB STAR 3.0 file. `read_nmrstar_file()` parses BMRB STAR 3.0 file. It will extract sequence information and chemical shifts for both alpha and beta carbons.

Usage

```
read_nmrstar_file(file_path)
```

Arguments

`file_path`	File path where input chemical shifts file is located

Value

Protein sequence and chemical shifts dataframe.

Examples

```
## Download a BMRB file
library(BMRBr)
## Not run: bmrb_download(id_list = "4020", output_dir = "./", verbose = F)

## Read in BMRB file and procec
file_path = "bmr4020.str"
## Not run: head(read_nmrstar_file(file_path)

## Delete downloaded BMRB file
## Not run: unlink("./bmr4020.str")
```

read_raw_file	*Extracts data from a protein NMR experimental peak list.* read_raw_file() *function reads in a user provided protein NMR experimental peak list. It currently supports file format in csv, txt with deliminator of comma, whitespace or semicolon. Note: please don't leave space between sequence and chemical shifts data, otherwise it will report error.*

Description

Extracts data from a protein NMR experimental peak list. read_raw_file() function reads in a user provided protein NMR experimental peak list. It currently supports file format in csv, txt with deliminator of comma, whitespace or semicolon. Note: please don't leave space between sequence and chemical shifts data, otherwise it will report error.

Usage

```
read_raw_file(file_path, delim = "comma", assigned = FALSE)
```

Arguments

file_path	File path where input chemical shifts file is located
delim	Delimiter for parsing file
assigned	Flag tell whether the input chemical shifts file is already assigned or not

Value

A list contains protein sequence and chemical shift table.

Examples

```
input_type = "ws"
sample_file_path = system.file("extdata", "sample_input_ws.txt", package = "BaMORC")
head(read_raw_file(file_path=sample_file_path, delim="ws"))

input_type = "csv"
sample_file_path = system.file("extdata", "sample_input.csv", package = "BaMORC")
head(read_raw_file(file_path=sample_file_path, delim="comma"))
unlink("sample_input.csv")

input_type = "sc"
sample_file_path = system.file("extdata", "sample_input_sc.txt", package = "BaMORC")
head(read_raw_file(file_path=sample_file_path, delim="semicolon"))
unlink("sample_input_sc.txt")
```

RefDB.StatCA *Statistics of chemical shifts values of alpha carbons from RefDB.*

Description

A dataset containing statistics of chemical shifts of alpha carbons for each amino acid from RefDB data:

- Residue. amino acid typing (single-letter convention)
- Coil.mu. mean value of alpha carbon in coil
- Coil.sd. mean value of alpha carbon in coil
- Helix.mu. mean value of alpha carbon in helix
- Helix.sd. standard deviation value of alpha carbon in helix
- Beta.mu. mean value of alpha carbon in coil
- Beta.sd. standard deviation value of alpha carbon in coil
- Avg.mu. average of mean values of alpha carbon across three secondary structures
- Avg.sd. average of standard deviation values of alpha carbon across three secondary structures

Usage

```
RefDB.StatCA
```

Format

A data frame with 21 rows and 9 columns.

RefDB.StatCB *Statistics of chemical shifts values of beta carbons from RefDB.*

Description

A dataset containing statistics of chemical shifts of beta carbons for each amino acid from RefDB data:

- Residue. amino acid typing (single-letter convention)
- Coil.mu. mean value of beta carbon in coil
- Coil.sd. mean value of beta carbon in coil
- Helix.mu. mean value of beta carbon in helix
- Helix.sd. standard deviation value of beta carbon in helix
- Beta.mu. mean value of beta carbon in coil
- Beta.sd. standard deviation value of beta carbon in coil
- Avg.mu. average of mean values of beta carbon across three secondary structures
- Avg.sd. average of standard deviation values of beta carbon across three secondary structures

Usage

```
RefDB.StatCB
```

Format

A data frame with 21 rows and 9 columns.

`RefDB_data` *RefDB object*

Description

Collection of data from Re-referenced Protein Chemical shift Database (RefDB) and their access functions.

Usage

```
RefDB_data
```

Format

A class contains 1557 13C datasets and access functions.

Details

$AminoAcid data set of 19 amino acid single-letter naming convention (excluding glycine)

$carbonDat data set (list) of all 1557 RefDB carbon 13 raw data pre-processed in data frame format

$carbonDat_narm data set (list) of all 1557 RefDB carbon 13 raw data pre-processed in data frame format with NA removed

$carbonDat_rmGU data set (list) of all 1557 RefDB carbon 13 raw data pre-processed in data frame format with NA, glycine and undetermined secondary structure elements removed

$DBID data set of all 1557 RefDB IDs

$getData(index = NA, ID = NA, type = "raw") function that return single dataset, (index or ID, only one is allowed)

- index. the index of the dataset
- ID. the RefDB ID number of the dataset
- type what kind of data will be fetched. "raw": returns raw data; "datatable": return data in data frame format; "rmNA": return data with NA removed; "rmGU: return data with NA, glycine and undetermined secondary structure elements removed

$getFreq(index = NA, ID = NA) function that return amino acid relative cumulative frequency for each data set (index or ID, only one is allowed)

$getSecStr(index = NA, ID = NA) function that return secondary structure information for each data set (index or ID, only one is allowed)

$getSequence(index = NA, ID = NA) function that return protein sequence information for each data set (index or ID, only one is allowed)

$rawData the data set contains all the raw data from RefDB

`unassigned_bamorc` *Calculates the referencing correction value for unassigned protein NMR peaklists.* `unassigned_bamorc()` *will analyze unassigned protein NMR spectra, first groups the peaklist via SSC, then estimates the secondary structure via JPred, finally using BaMORC core function to calculate the reference correction value.*

Description

Calculates the referencing correction value for unassigned protein NMR peaklists. `unassigned_bamorc()` will analyze unassigned protein NMR spectra, first groups the peaklist via SSC, then estimates the secondary structure via JPred, finally using BaMORC core function to calculate the reference correction value.

Usage

```
unassigned_bamorc(peakList_file_loc, sequence,
  secondary_structure = NULL, from = -5, to = 5,
  ssc = "moseleybioinformaticslab/ssc",
  para = "--plformat=sparky --stype=HNcoCACB --dims=H,N,CA/CB --rdims=H,N")
```

Arguments

`peakList_file_loc`	NMR HNCACB file location
`sequence`	sequence string of protein of interest
`secondary_structure`	string of secondary structure if available
`from`	the lower bound of the optimization
`to`	the upper bound of the optimization
`ssc`	location of ssc docker image
`para`	parameter input for ssc function (no need to change)

Value

Reference correction value.

Examples

```
sequence = "RPAFCLEPPYAGPGKARIIRYFYNAAAGAAQAFVYGGVRAKRNNFASAADALAACAAA"
peakList_file_loc = system.file("extdata", "bpti_HNcoCACB.txt", package = "BaMORC")
## Not run: unassigned_bamorc(peakList_file_loc, sequence, secondary_structure=NULL,
from=-5, to=5, ssc="moseleybioinformaticslab/ssc",
para="--plformat=sparky --stype=HNcoCACB --dims=H,N,CA/CB --rdims=H,N")
## End(Not run)
# Expected result should be around (due to randomness): 0.0007890328
```

Index