Package ‘procoil’

February 2, 2026

Type Package

Title Prediction of Oligomerization of Coiled Coil Proteins

Version 2.39.0

Date 2025-09-19

Depends R (>= 3.3.0), kebabs

Imports methods, stats, graphics, S4Vectors, Biostrings, utils

Suggests knitr

Maintainer Ulrich Bodenhofer <ulrich@bodenhofer.com>

Description The package allows for predicting whether a coiled coil sequence (amino acid sequence plus heptad register) is more likely to form a dimer or more likely to form a trimer. Additionally to the prediction itself, a prediction profile is computed which allows for determining the strengths to which the individual residues are indicative for either class. Prediction profiles can also be visualized as curves or heatmaps.

License GPL (>= 2)

Collate AllClasses.R access-methods.R show-methods.R plot-methods.R predict-methods.R weights-methods.R profile-methods.R readCCModel.R writeCCModel.R fitted-methods.R heatmap-methods.R

URL https://github.com/UBod/procoil

VignetteBuilder knitr

LazyLoad yes

LazyData yes

biocViews Proteomics, Classification, SupportVectorMachine

git_url https://git.bioconductor.org/packages/procoil

git_branch devel

git_last_commit c3f2a65

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2026-02-01

Author Ulrich Bodenhofer [aut, cre]

Contents

`procoil-package` *Prediction of Oligomerization of Coiled Coil Proteins*

Description

The package allows for predicting whether a coiled coil sequence (amino acid sequence plus heptad register) is more likely to form a dimer or more likely to form a trimer. Additionally to the prediction itself, a prediction profile is computed which allows for determining the strengths to which the individual residues are indicative for either class. Prediction profiles can also be visualized as curves or heatmaps.

Details

The package defines two S4 classes, `CCModel` and `CCProfile`. The former's purpose is to represent a coiled coil prediction model. The default model `PrOCoilModel` is pre-loaded when the package is loaded. An alternative model `PrOCoilModelBA` is also available. Other models can be loaded with the function `readCCModel`. The `predict` function is used to predict the oligomerization of one or more coiled coil sequences (which consist of a amino acid sequences and heptad registers aligned to them). The result is stored in a `CCProfile` object. The resulting prediction profile can be visualized with `plot`.

Author(s)

Ulrich Bodenhofer

References

`https://github.com/UBod/procoil/`

Mahrenholz, C.C., Abfalter, I.G., Bodenhofer, U., Volkmer, R., and Hochreiter, S. (2011) Complex networks govern coiled coil oligomerization - predicting and profiling by means of a machine learning approach. Mol. Cell. Proteomics 10(5):M110.004994. DOI: doi:10.1074/mcp.M110.004994.

Examples

```
## display summary of default model
PrOCoilModel

## predict oligomerization of GCN4 wildtype
GCN4wt <- predict(PrOCoilModel,
                  "MKQLEDKVEELLSKNYHLENEVARLKKLV",
                  "abcdefgabcdefgabcdefgabcdefga")

## display result
GCN4wt

## plot profile
plot(GCN4wt)

## predict oligomerization of unknown sequence (Marcoil example)
MarcoilEx <- predict(PrOCoilModel,
    "MGECDQLLVFMITSRVLVLSTLIIMDSRQVYLENLRQFAENLRQNIENVHSFLENLRADLENLRQKFPGKWYSAMPGRHG",
    "-------------------------------abcdefgabcdefgabcdefgabcdefgabcdefg--------------")

## display result
MarcoilEx

## plot profile
plot(MarcoilEx)
```

`CCModel-class` *Class "CCModel"*

Description

S4 class representing a coiled coil prediction model

Objects from the Class

In principle, objects of this class can be created by calls of the form `new("CCModel")`, although it is probably never necessary to create such an object from scratch - and not advised either. The default model is stored in the object `PrOCoilModel`. An alternative model, `PrOCoilModelBA`, that is optimized for balanced accuracy is available too (see below). Custom models can be loaded from files using the function `readCCModel`.

Discriminant function of model

Given a new coiled coil sequence x and a model, the discriminant function of the model is given as

$$f(x) = b + \sum_{p \in P} N(p, x) \cdot w(p),$$

where b is a constant offset, $N(p, x)$ denotes the number of occurrences of pattern p in sequence x, and $w(p)$ is the weight assigned to pattern p. P is the set of all patterns contained in the model. In the models used in the **procoil** package, the weights are computed from a support vector machine. Models can include kernel normalization or not. The formula above refers to the variant without kernel normalization. If kernel normalization is employed, the weights are computed in a different way and the discriminant function changes to

$$f(x) = b + \frac{\sum_{p \in P} N(p, x) \cdot w(p)}{R(x)},$$

where $R(x)$ is a normalization value depending on the sample x. It is defined as follows:

$$R(x) = \sqrt{\sum_{p \in P} N(p, x)^2}$$

The **procoil** package does not consider arbitrary patterns, but only very specific ones: pairs of amino acids at fixed register positions with no more than a maximum number m of residues in between. Internally, these patterns are represented as strings with an amino acid letter on the first position, then a certain number of wildcards (between 0 and m as noted above), then the second amino acid letter, and an aligned sequence with the same number of wildcards and letters 'a'-'g' denoting the heptad register position on the first and last amino acid, e.g. "N..La..d". This pattern matches a coiled coil sequence if the sequence has an 'N' (Asparagine) at an 'a' position and a 'L' (Leucine) at the next 'd' position. For instance, the GCN4 wildtype has one occurrence of this pattern:

```
MKQLEDKVEELLSKNYHLENEVARLKKLV
abcdefgabcdefgabcdefgabcdefga
              N..L
              a  d
```

Slots

`b`**:** Object of class `numeric` the value b as described above

`m`**:** Object of class `integer` the value m as described above

`scaling`**:** Object of class `logical` indicating whether the model should employ kernel normalization

`weights`**:** Object of class `matrix` storing all pattern weights; the matrix in this slot is actually consisting of only one row that contains the weights. The patterns are stored in column names of the matrix and encoded in the format described above

Methods

predict `signature(object = "CCModel")`: see `predict`

show `signature(object = "CCModel")`: displays the most important information stored in the `CCModel` object `object`, such as, kernel parameters and a summary of weights.

weights `signature(object="CCModel")`: returns the weights stored in `object` as a named numeric vector.

Default model `PrOCoilModel`

The **procoil** package provides a default coiled coil prediction model, `PrOCoilModel`. The model was created with the **kebabs** package *[Palme et al., 2015]* using the coiled coil kernel with $m = 5$, $C = 2$, and kernel normalization on the BLAST-augmented data set. It is optimized for standard (unbalanced) accuracy, i.e. it tries to minimize the probability of misclassifications. Since dimers are more frequent in the data set, it slightly favors dimers for unknown sequences.

Note that this is not the original model as described in *[Mahrenholz et al., 2011]*. The models have been re-trained for version 2.0.0 of the package using a newer snapshot of PDB and newer methods. The original models are still available for download and can still be used if the user wishes to. For detailed instructions, see the package vignette.

Alternative model `PrOCoilModelBA`

As mentioned above, the default model `PrOCoilModel` slightly favors dimers. This may be undesirable for some applications. For such cases, an alternative model `PrOCoilModelBA` is available that is optimized for balanced accuracy, i.e. it tries not to favor the larger class - dimers -, but may therefore prefer trimers in borderline cases. The overall misclassification probability is slightly higher for this model than for the default model `PrOCoilModel`.

The model `PrOCoilModelBA` was created with PSVM *[Hochreiter and Obermayer, 2006]* using the coiled coil kernel with $m = 8$, $C = 8$, $\varepsilon = 0.8$, class balancing, and kernel normalization on the PDB data set (i.e. without BLAST augmentation).

The same applies as for `PrOCoilModel`: this model has been re-trained for package version 2.0.0. For detailed instructions how to use the original models, see the package vignette.

Author(s)

Ulrich Bodenhofer

References

`https://github.com/UBod/procoil/`

Mahrenholz, C.C., Abfalter, I.G., Bodenhofer, U., Volkmer, R., and Hochreiter, S. (2011) Complex networks govern coiled coil oligomerization - predicting and profiling by means of a machine learning approach. Mol. Cell. Proteomics 10(5):M110.004994. DOI: doi:10.1074/mcp.M110.004994.

Palme, J., Hochreiter, S., and Bodenhofer, U. (2015) KeBABS: an R package for kernel-based analysis of biological sequences. Bioinformatics 31(15):2574-2576. DOI: doi:10.1093/bioinformatics/btv176.

Hochreiter, S., and Obermayer, K. (2006) Support vector machines for dyadic data. Neural Computation 18:1472-1510. DOI: doi:10.1162/neco.2006.18.6.1472.

See Also

`predict-methods`

Examples

```
showClass("CCModel")

## show summary of default model (optimized for accuracy)
PrOCoilModel

## show weight of pattern "N..La..d"
weights(PrOCoilModel)["N..La..d"]

## show the 10 patterns that are most indicative for trimers
## (as the weights are sorted in descending order in PrOCoilModel)
weights(PrOCoilModel)[1:10]

## predict oligomerization of GCN4 wildtype
GCN4wt <- predict(PrOCoilModel,
                  "MKQLEDKVEELLSKNYHLENEVARLKKLV",
                  "abcdefgabcdefgabcdefgabcdefga")

## show summary of alternative model (optimized for balanced accuracy)
PrOCoilModelBA

## show weight of pattern "N..La..d"
weights(PrOCoilModelBA)["N..La..d"]

## show the 10 patterns that are most indicative for trimers
## (as the weights are sorted in descending order in PrOCoilModelBA)
weights(PrOCoilModelBA)[1:10]
```

CCModel-FileOps *Reading and writing of coiled coil prediction model from/to files*

Description

Functions for reading a coiled coil prediction models from a file into a CCModel object and writing a CCModel object to a file.

Usage

```
readCCModel(file)
writeCCModel(object, file)
```

Arguments

file	the name of the file from which readCCModel should read the model / the name of the file to which writeCCModel should write the model
object	the CCModel object that writeCCModel should write to a file

Details

The **procoil** package comes with two ready-made models for oligomerization prediction, `PrOCoilModel` and `PrOCoilModelBA`. In case the user wants to define custom models or wishes to use previous versions of the prediction models, the functions `readCCModel` and `writeCCModel` can be used to read/write models from/to plain text files that can be viewed and also modified.

`writeCCModel` writes models in the following format:

```
_b,-1.07262284445085
_m,5
_scaling,1
L...Vd...a,1.63626232200227
R....Eg....e,1.5382098040217
R.Ec.e,1.29025032360792
E..Ve..a,1.22837780239385
...
```

Correspondingly, `readCModel` expects the file to conform to the above format. See `CCModel` for an overview of model parameters and an explanation of patterns and weights.

Value

Upon successful completion, `readCCModel` returns a `CCModel` object. `writeCCModel` returns an invisible `NULL`.

Note

The PrOCoil model is available on on `http://www.bioinf.jku.at/software/procoil/PrOCoilModel_v2.CCModel`. in exactly the format the function `readCCModel` requires. Analogously for the alternative model optimized for balanced accuracy (see `CCModel`): `http://www.bioinf.jku.at/software/procoil/PrOCoilModelBA_v2.CCModel`. The original models described in *[Mahrenholz et al., 2011]* are available on `http://www.bioinf.jku.at/software/procoil/PrOCoilModel_v1.CCModel` and `http://www.bioinf.jku.at/software/procoil/PrOCoilModelBA_v1.CCModel`, respectively. So, by loading one of these files, the original models can still be used.

Author(s)

Ulrich Bodenhofer

References

`https://github.com/UBod/procoil/`

Mahrenholz, C.C., Abfalter, I.G., Bodenhofer, U., Volkmer, R., and Hochreiter, S. (2011) Complex networks govern coiled coil oligomerization - predicting and profiling by means of a machine learning approach. Mol. Cell. Proteomics 10(5):M110.004994. DOI: doi:10.1074/mcp.M110.004994.

See Also

`procoil`, `CCModel-class`

Examples

```
## load small example model file for testing purposes
## NOTE: this is an incomplete model that will probably not provide
##       meaningful predictions
file <- system.file("examples", "testModel.CCModel", package="procoil")
testModel <- readCCModel(file)

testModel

## Not run:
## read original model from file
URL <- "http://www.bioinf.jku.at/software/procoil/PrOCoilModel_v1.CCModel"
PrOCoilModelV1 <- readCCModel(URL)

## display summary of example model
PrOCoilModelV1

## display 10 heightes pattern weights
weights(PrOCoilModelV1)[1:10]

## End(Not run)
```

CCProfile-class *Class "CCProfile"*

Description

S4 class for representing coiled coil prediction results

Objects from the Class

In principle, objects of this class can be created by calls of the form `new("CCProfile")`, although it is not advised to do so. Most importantly, the `predict` function of returns its results in objects of this type.

Slots

This class extends the class `PredictionProfile` from the **kebabs** package directly and therefore inherits all its slots and methods. The following slots are defined for `CCProfile` objects additionally:

`disc`**:** Object of class `numeric` containing the discriminant function value(s) (see `CCModel` for details)

`pred`**:** Object of class `factor` containing the final classification(s). Upon a call to `predict`, it is either “trimer” or “dimer”.

Prediction profiles

As described in `CCModel`, the discriminant function of the coiled coil classifier is essentially a weighted sum of numbers of occurrences of certain patterns in the sequence under consideration, i.e. every pattern occurring in the sequence contributes a certain weight to the discriminant function. Since every such occurrence is uniquely linked to two specific residues in the sequence, every amino acid in the sequence contributes a unique weight to the discriminant function value which is nothing else but half the sum of weights of matching patterns in which this amino acid is involved. If we denote the contribution of each position i with $s_i(x)$, it follows immediately that

$$f(x) = b + \sum_{i=1}^{L} s_i(x),$$

where L is the length of the sequence x. The values $s_i(x)$ can then be understood as the contributions that the i-th residue makes to the overall classification of the sequence x, which we call *prediction profile*. These profiles can either be visualized as they are without taking the offset b into account or by distributing b equally over all residues. These are the so-called *baselines* that are included in `CCProfile` objects. They are computed as $-b/L$.

Methods

plot `signature(x="CCProfile", y="missing")`: see `plot`

heatmap `signature(x="CCProfile", y="missing")`: if the `CCProfile` object `x` contains the profiles of at least three sequences, the profiles are visualized as a heatmap. This method is inherited from the **kebabs** package; for details, see `heatmap`.

show `signature(object="CCProfile")`: displays the most important information stored in the `CCProfile` object `object`, such as, the sequences, kernel parameters, baselines, profiles, and classification results.

Accessor-like methods

The `CCProfile` class inherits all accessors from the `PredictionProfile` class, such as, `sequences`, `baselines`, `profiles`, and the indexing operator `x[i]`. Additionally, the **procoil** package defines the following two methods:

profile `signature(fitted="CCProfile")`: for compatibility with previous versions, a method `profile` is available, too. It extracts the profile(s) in the same way as `profiles`

fitted `signature(object="CCProfile")`: extracts the final classifications. This function returns a factor with levels “dimer” and “trimer”. If `decision.values=TRUE` is specified, a numeric vector is attached to the result as an attribute `"decision.values"` which also contains the discriminant function values.

Author(s)

Ulrich Bodenhofer

References

https://github.com/UBod/procoil/

Mahrenholz, C.C., Abfalter, I.G., Bodenhofer, U., Volkmer, R., and Hochreiter, S. (2011) Complex networks govern coiled coil oligomerization - predicting and profiling by means of a machine learning approach. Mol. Cell. Proteomics 10(5):M110.004994. DOI: doi:10.1074/mcp.M110.004994.

Palme, J., Hochreiter, S., and Bodenhofer, U. (2015) KeBABS: an R package for kernel-based analysis of biological sequences. Bioinformatics 31(15):2574-2576. DOI: doi:10.1093/bioinformatics/btv176.

See Also

CCModel, plot, plot, PredictionProfileAccessors,

Examples

```
showClass("CCProfile")

## predict oligomerization of GCN4 wildtype
GCN4wt <- predict(PrOCoilModel,
                  "MKQLEDKVEELLSKNYHLENEVARLKKLV",
                  "abcdefgabcdefgabcdefgabcdefga")

## display summary of result
GCN4wt

## show raw prediction profile
profile(GCN4wt)

## plot profile
plot(GCN4wt)

## define four GCN4 mutations
GCN4mSeq <- c("GCN4wt"        ="MKQLEDKVEELLSKNYHLENEVARLKKLV",
              "GCN4_N16Y_L19T"="MKQLEDKVEELLSKYYHTENEVARLKKLV",
              "GCN4_E22R_K27E"="MKQLEDKVEELLSKNYHLENRVARLEKLV",
              "GCN4_V23K_K27E"="MKQLEDKVEELLSKNYHLENEKARLEKLV")
GCN4mReg <- rep("abcdefgabcdefgabcdefgabcdefga", 4)

## predict oligomerization
GCN4mut <- predict(PrOCoilModel, GCN4mSeq, GCN4mReg)

## display summary of result
GCN4mut

## display predictions
fitted(GCN4mut)

## overlay plot of two profiles
plot(GCN4mut[c(1, 2)])

## show heatmap
```

```
heatmap(GCN4mut)
```

---

plot-methods *Plotting prediction profiles*

Description

Functions for plotting prediction profiles

Usage

```
## S4 method for signature 'CCProfile,missing'
plot(x, col=c("red", "blue"),
     standardize=TRUE, shades=NULL, legend="default",
     legendPos="topright", xlab="", ylab="weight",
     lwd.profile=1, lwd.axis=1, las=1,
     heptads=TRUE, annotate=TRUE, ...)
```

Arguments

`x`	Object of class `CCProfile` to be plotted with `plot`
`col`	Character string containing the name(s) of the color(s) in which the profile(s) should be plotted.
`standardize`	If `FALSE`, the profile values s_i are displayed as they are with the value $y = -b/L$ superimposed as a light gray line. If `TRUE` (default), the profile(s) is/are shifted by the baseline values $-b/L$ and the light gray line is displayed at $y = 0$.
`shades`	Vector of at least two color specifications (default: NULL). If not NULL, the background area above and below the base line $y = -b/L$ are shaded in colors `shades[1]` and `shades[2]`, respectively.
`legend`	A character string containing the legend/description of the profile. If `"default"`, the names of the sequences/profiles are used. If no names are available, the profiles are simply enumerated (as long as two profiles should be plot together; if only a single unnamed profile is to be plotted, no legend is shown). If `legend` is an empty string, no legend is displayed at all.
`legendPos`	position specification for legend (if `legend` is specified). Can either be a vector with coordinates or a single keyword like “topright” (see `legend`).
`xlab`	label of horizontal axis, empty by default.
`ylab`	label of vertical axis, defaults to “weight”.
`lwd.profile`	profile line width as described for parameter `lwd` in `par`
`lwd.axis`	axis line width as described for parameter `lwd` in `par`
`las`	see `par`
`heptads`	if `TRUE` (default), the heptad structure is indicated by vertical light gray lines separating the different heptads. Heptad irregularities are indicated with red lines.

`annotate`	if `TRUE` (default), the heptad annotation information is shown in the center of the plot.
`...`	all other arguments are passed to the `plot` method from the **kebabs** package

Details

The `plot` function displays a prediction profile as a step function over the sequence with the steps connected by vertical lines. The sequence and the heptad register are visualized below and above the profile, respectively. The baseline value $-b/L$ and the light gray line has the following meaning: It is obvious that we can rewrite

$$f(x) = b + \sum_{i=1}^{L} s_i(x)$$

as

$$f(x) = \sum_{i=1}^{L} (s_i(x) - (-\frac{b}{L}))$$

so the discriminant function value $f(x)$ can be understood as the sum of values $s_i(x) - (-\frac{b}{L})$, i.e. the area between the constant value $-b/L$ and the prediction profile. If the area above the light gray line is greater than the area below the light gray line, the sequence is predicted as trimer, otherwise as dimer.

If `plot` is called for a `CCProfile` object that contains profiles of two sequences, the two profiles are plotted together to facilitate a comparison of profiles (e.g. wild type sequences versus mutants). Although the `plot` function tolerates profiles/sequences with different lengths and/or unaligned heptad registers, it is obvious that the superimposition of profiles of two unaligned, unrelated sequences makes little sense.

The `plot` functions gives an error if is called for a `CCProfile` object that contains profiles of three or more sequences.

The given function is only a wrapper around the `plot` function provided by the **kebabs** package. The only difference is that heptad seperators (argument `heptads`) and the heptad annotation (argument `annotate`) are displayed by default. Moreover, presently, no legend is displayed by default if a single profile is plotted for an unnamed sequence.

Value

This function does not return any value.

Author(s)

Ulrich Bodenhofer

References

https://github.com/UBod/procoil/

Mahrenholz, C.C., Abfalter, I.G., Bodenhofer, U., Volkmer, R., and Hochreiter, S. (2011) Complex networks govern coiled coil oligomerization - predicting and profiling by means of a machine learning approach. Mol. Cell. Proteomics 10(5):M110.004994. DOI: doi:10.1074/mcp.M110.004994.

Palme, J., Hochreiter, S., and Bodenhofer, U. (2015) KeBABS: an R package for kernel-based analysis of biological sequences. Bioinformatics 31(15):2574-2576. DOI: doi:10.1093/bioinformatics/btv176.

See Also

procoil, CCModel, CCProfile

Examples

```
## predict oligomerization of GCN4 wildtype
GCN4wt <- predict(PrOCoilModel,
                  "MKQLEDKVEELLSKNYHLENEVARLKKLV",
                  "abcdefgabcdefgabcdefgabcdefga")

## plot profile
plot(GCN4wt)

## define two GCN4 mutations
GCN4mSeq <- c("GCN4wt"       ="MKQLEDKVEELLSKNYHLENEVARLKKLV",
              "GCN4_N16I_L19N"="MKQLEDKVEELLSKIYHNENEVARLKKLV")
GCN4mReg <- rep("abcdefgabcdefgabcdefgabcdefga", 2)

## predict oligomerization
GCN4mut <- predict(PrOCoilModel, GCN4mSeq, GCN4mReg)

## overlay plot of the two profiles
plot(GCN4mut)
```

`predict-methods` *Predict oligomerization of one or more coiled coil segments*

Description

Function for predicting the oligomerization of one or multiple coiled coil segments

Usage

```
## S4 method for signature 'CCModel'
predict(object, seq, reg)
```

Arguments

`object`	The model to be considered; can either be one of the models included in the package (`PrOCoilModel` and `PrOCoilModelBA`) or any other model loaded or created by the user. For a detailed explanation of the two default models, see `CCModel`.
`seq`	One or several amino acid sequences; valid characters are all uppercase letters except 'B', 'J', 'O', 'U', 'X', and 'Z'; invalid characters are tolerated, but ignored by the prediction. This argument can be a character vector, an `AAString` object, an `AAStringSet` object, or an `AAVector` object
`reg`	a character vector containing the heptad register(s); valid characters are the lowercase letters 'a'-'g' and dashes '-'. Can also be omitted, see details below.

Details

The function `predict` is the most important one in the **procoil** package. It is used to apply a coiled coil prediction model to coiled coil sequences/segments. It uses the discriminant function described in `CCModel`. By default the final classification is computed on the basis of the discriminant function value $f(x)$. If $f(x) >= 0$, the sequence x is predicted as trimer, otherwise as dimer.

If the `reg` argument is missing, `predict` looks whether the object passed as argument `seq` includes heptad register information, either as an attribute `reg` (if `seq` is a character vector), as metadata field `reg` (if `seq` is an `AAString` or `AAStringSet` object), or via annotation metadata (if `seq` is an `AAStringSet` or `AAVector` object; see `annotationMetadata`). In any case, the `reg` argument has priority over all other ways of specifying the heptad annotation. In other words, if `reg` is specified and `seq` contains heptad annotations in one of the ways described above, the `reg` argument has priority and the heptad annotation in `seq` is ignored.

The `reg` argument must have exactly as many elements as `seq` has sequences, and the registers must be aligned to the sequences, i.e. the first register must be exactly as long as the first sequence, and so on.

If heptad registers contain dashes, the `predict` function extracts all contiguous coiled coil segments and computes predictions for all of them. The returned `CCProfile` object then contains profiles/predictions of all coiled coil segments that were extracted from `seq` (see example below).

Value

returns a `CCProfile` object

Author(s)

Ulrich Bodenhofer

References

`https://github.com/UBod/procoil/`

Mahrenholz, C.C., Abfalter, I.G., Bodenhofer, U., Volkmer, R., and Hochreiter, S. (2011) Complex networks govern coiled coil oligomerization - predicting and profiling by means of a machine learning approach. Mol. Cell. Proteomics 10(5):M110.004994. DOI: doi:10.1074/mcp.M110.004994.

See Also

procoil, CCModel, CCProfile

Examples

```
## predict oligomerization of GCN4 wildtype
GCN4wt <- predict(PrOCoilModel,
                  "MKQLEDKVEELLSKNYHLENEVARLKKLV",
                  "abcdefgabcdefgabcdefgabcdefga")

## show result
GCN4wt

## example with four GCN4 mutations
GCN4mSeq <- c("GCN4wt"        ="MKQLEDKVEELLSKNYHLENEVARLKKLV",
              "GCN4_N16Y_L19T"="MKQLEDKVEELLSKYYHTENEVARLKKLV",
              "GCN4_E22R_K27E"="MKQLEDKVEELLSKNYHLENRVARLEKLV",
              "GCN4_V23K_K27E"="MKQLEDKVEELLSKNYHLENEKARLEKLV")

## to illustrate the alternative interface, we convert this
## character vector to an 'AAStringSet' object and add
## heptad registers as annotation metadata
GCN4mAA <- AAStringSet(GCN4mSeq)
annotationMetadata(GCN4mAA, annCharset="abcdefg") <-
    rep("abcdefgabcdefgabcdefgabcdefga", 4)

## predict oligomerization (note: no 'reg' argument!)
GCN4mut <- predict(PrOCoilModel, GCN4mAA)

## display summary of result
GCN4mut

## predict oligomerization of unknown sequence (Marcoil example)
MarcoilEx <- predict(PrOCoilModel,
   "MGECDQLLVFMITSRVLVLSTLIIMDSRQVYLENLRQFAENLRQNIENVHSFLENLRADLENLRQKFPGKWYSAMPGRHG",
   "--------------------------------abcdefgabcdefgabcdefgabcdefgabcdefg--------------")

## show results
MarcoilEx
```

Index