Software Manual

Institute of Bioinformatics, Johannes Kepler University Linz

PrOCoil — A Web Service and an R Package for Predicting the Oligomerization of Coiled Coil Proteins

Ulrich Bodenhofer

Institute of Bioinformatics, Johannes Kepler University Linz
Altenberger Str. 69, 4040 Linz, Austria
procoil@bioinf.jku.at

Version 2.0.2, April 12, 2016

Scope and Purpose of this Document

This document is a user manual for PrOCoil, the software suite accompanying the paper [**?**]. It provides a gentle introduction into how to use PrOCoil. Not all features of the R package are described in full detail. Such details can be obtained from the documentation enclosed in the R package. Further note the following: (1) this is not an introduction to coiled coil proteins; (2) this is not an introduction to R; (3) this is not an introduction to support vector machines. If you lack the background for understanding this manual, you first have to read introductory literature on the subjects mentioned above.

Contents

1 Introduction

This user manual describes the PrOCoil software suite that accompanies the paper [**?**]. This software is concerned with analyzing coiled coil sequences in terms of their oligomerization behavior. PrOCoil does not only provide a prediction, but also detailed insights into which residues or subsequences are responsible for the predicted oligomerization.

PrOCoil is available both as an easy-to-use Web interface and an R package. Both variants offer the same prediction and analysis facilities. The following table summarizes the essential differences:

PrOCoil Web interface	PrOCoil R package
can be used instantly on any computer with Internet access and a Web browser	requires R and installation of package `procoil`
supports only the standard PrOCoil model	supports the standard PrOCoil model and the alternative model optimized for balanced accuracy; other models can be loaded from files
graphics are produced in a non-customizable standard format	graphics are customizable
only single sequences or at most pairs of sequences with the same heptad register can be analyzed at a time	multiple sequences can be analyzed per run
beside standard input (amino acid sequence + aligned heptad annotation), also PairCoil2 output format is supported; Marcoil output can be used upon a separate pre-processing step	only standard input (amino acid sequence(s) + aligned heptad annotation(s))

We recommend beginners to use the Web interface. Experienced users can benefit from the greater flexibility of the R package. R package users can use the Web interface for converting PairCoil2 and/or Marcoil output into the input format the R package requires.

2 Input Data

As already mentioned, PrOCoil predicts whether a given coiled coil segment of an amino acid sequence is more likely to form a dimer or trimer. Such a segment has to consist of an amino acid (sub-)sequence and an aligned heptad annotation of the same length. As an example, the GCN4 yeast transcription factor is a dimer consisting of two equal sequences (i.e. it is a homo-dimer), the coiled coil parts of which (according to SOCKET [**?**]) look as follows:

```
MKQLEDKVEELLSKNYHLENEVARLKKLV
abcdefgabcdefgabcdefgabcdefga
```

The heptad annotation is essential for PrOCoil to work and cannot be omitted. The letters ‘`a`’–‘`g`’ correspond to the usual annotation of positions within the heptad motif. PrOCoil can also process

full amino acid sequences containing one or more coiled coil segments as sub-sequences, where the coiled coil segments must be annotated with the usual symbols ‘a’–‘g’ and non-coiled coil residues must be annotated with dashes ‘-’. As an example, let us consider the following sample sequence (annotation derived from running Marcoil[1] [**?**] on this sequence with a cut-off of 50%):

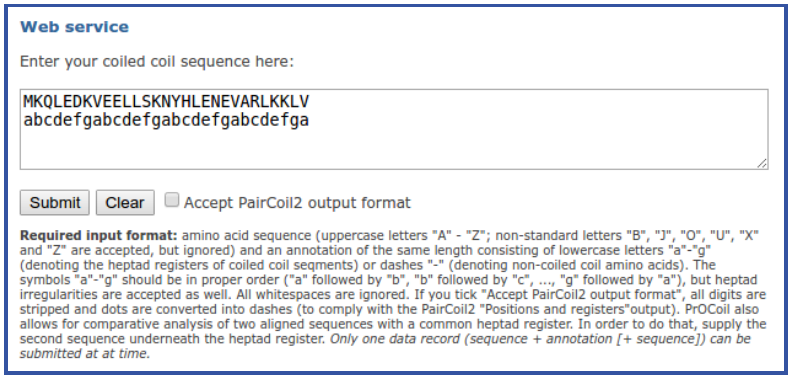
```
MGECDQLLVFMITSRVLVLSTLIIMDSRQVYLENLRQFAENLRQNIENVHSFLENLRADLENLRQKFPGKWYSAMPGRHG
------------------------------abcdefgabcdefgabcdefgabcdefgabcdefg--------------
```

If the heptad annotation contains dashes, PrOCoil extracts all coiled coil regions (contiguous non-dash regions) and classifies them separately. From the above sample sequence, PrOCoil will extract and classify the following coiled coil segment (residues 32–66 of the above example):

```
LENLRQFAENLRQNIENVHSFLENLRADLENLRQK
abcdefgabcdefgabcdefgabcdefgabcdefg
```

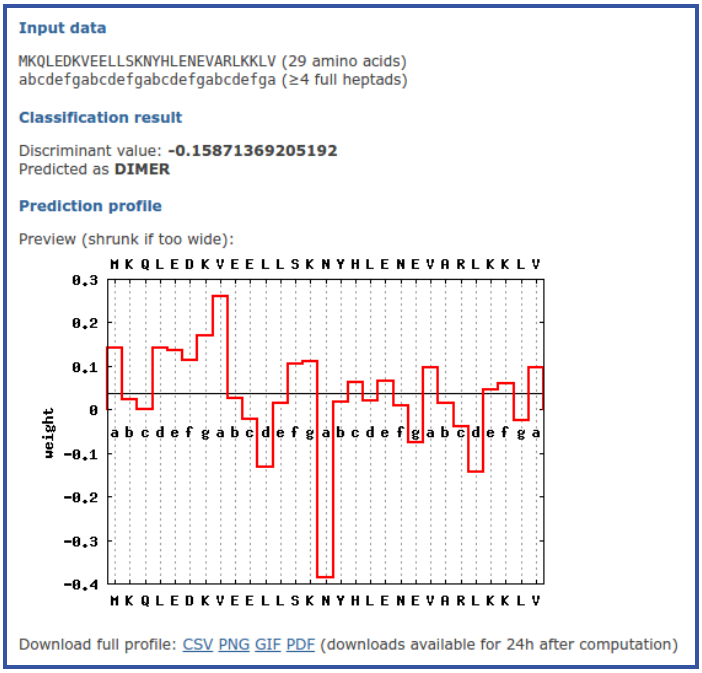
3 Predictions Using the Web Interface

The PrOCoil Web interface can be accessed directly via the PrOCoil homepage.[2] The page contains an input field that is pre-filled with the GCN4 wild type as an example:



The PrOCoil server first checks your input for validity (see Section **??**). If the input is correct, all coiled coil segments are extracted from the input and analyzed separately. For the standard example, the GCN4 wild type, the output looks as follows:

[1] http://www.isrec.isb-sib.ch/webmarcoil/webmarcoilC1.html
[2] http://www.bioinf.jku.at/software/procoil/



The output of the PrOCoil Web interface is structured into three sections:

Input data: amino acid sequence and heptad annotation of the coiled coil segment;

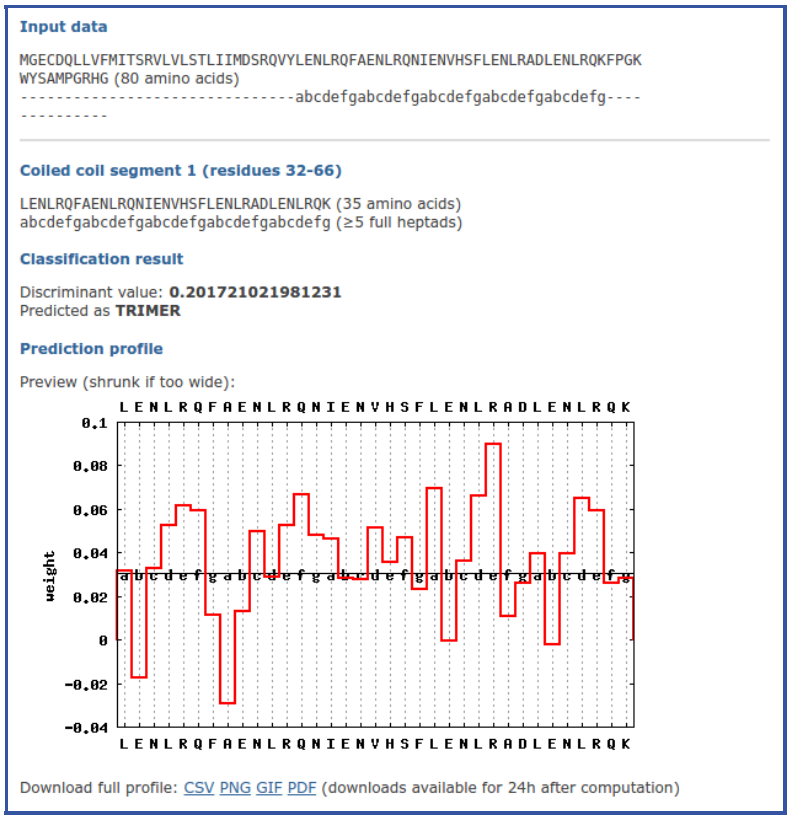
Classification result: discriminant function value and final classification; a positive value means that the sequence is classified as trimer, a negative value means that the sequence is classified as dimer. The higher the absolute value, the clearer the oligomerization tendency is. The closer the value is to zero, the more the sequence can be considered a borderline case.

Prediction profile: this plot shows the contribution of each residue to the discriminant function value. The more positive a value is for a residue, the more this residue contributes to an oligomerization tendency towards trimers. The more negative the value is, the more this residue contributes to an oligomerization tendency towards dimers. The horizontal line corresponds to the base line of the classifier (see **??**). The discriminant function value is actually obtained as the area above the grey baseline minus the area below the grey baseline. The links below the prediction profile plot allow for downloading the profile in various formats.

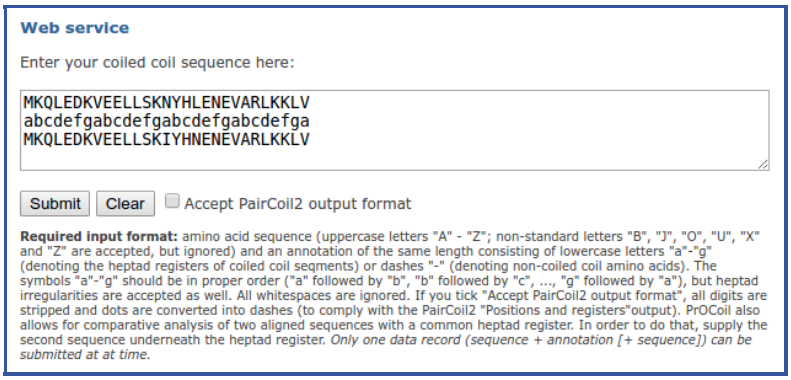
Note: Values in the prediction profile cannot be understood as general rules for which oligomerization behavior a given amino acid at a given heptad position is indicative for. PrOCoil takes pairwise interactions of amino acids into account. Therefore, the values in the prediction profile are always to be considered in the context of the given sequence. The same amino acid at the same heptad position might have a completely different value in the prediction profile of a different sequence.

If the heptad annotation contains at least one dash '-', PrOCoil first extracts all coiled coil segments, i.e. all contiguous sub-sequences with no dashes in the heptad annotation. Then all these

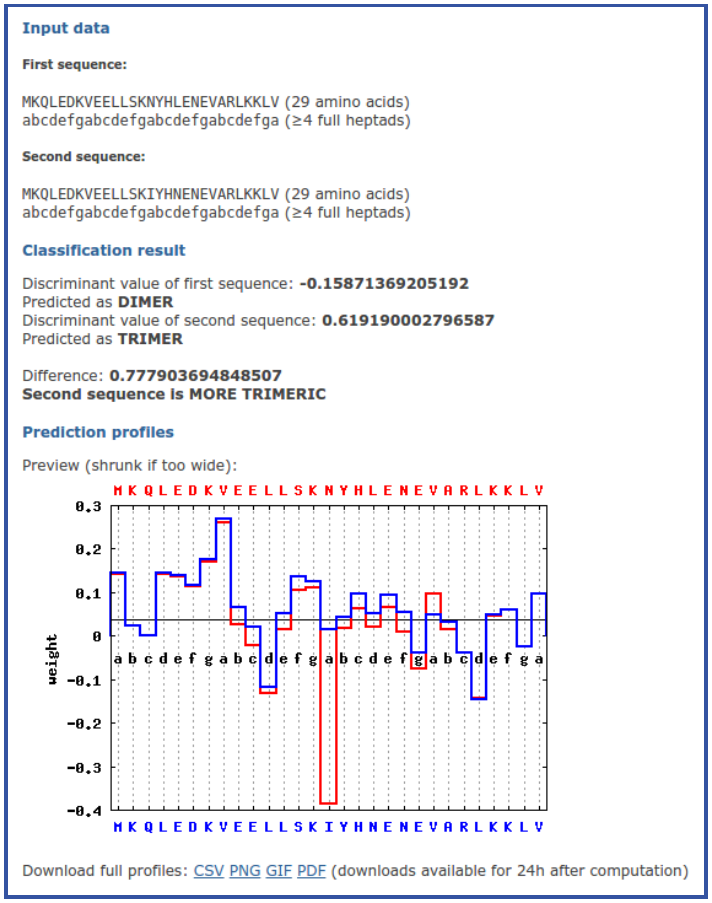
coiled coil segments are analyzed independently and the results are presented sequentially in the order they appear in the input sequence in the same format as above. The only notable difference is that the section “Input data” shows the complete input sequence, whereas the extracted coiled coil segment is displayed in a separate section “Coiled coil segment” along with its consecutive number and the corresponding start and end positions in the input sequence. For the above sample sequence, the output looks as follows:



The PrOCoil Web interface also facilitates easy analysis of mutations of coiled coil segments. This feature is limited to two sequences at a time. The two sequences must have the same length and heptad register. If these conditions are met, the second sequence can simply be written underneath the heptad register (with the first sequence remaining above the heptad register):



The output is structured as described for single sequences. The section “Classification result” shows discriminant function values and predictions of both sequences along with a comparison whether the second sequence is more dimeric or more trimeric than the first sequence. Only one profile plot is produced in which both profiles are visualized, the profile of the first sequence in red and the second sequence’s profile in blue:



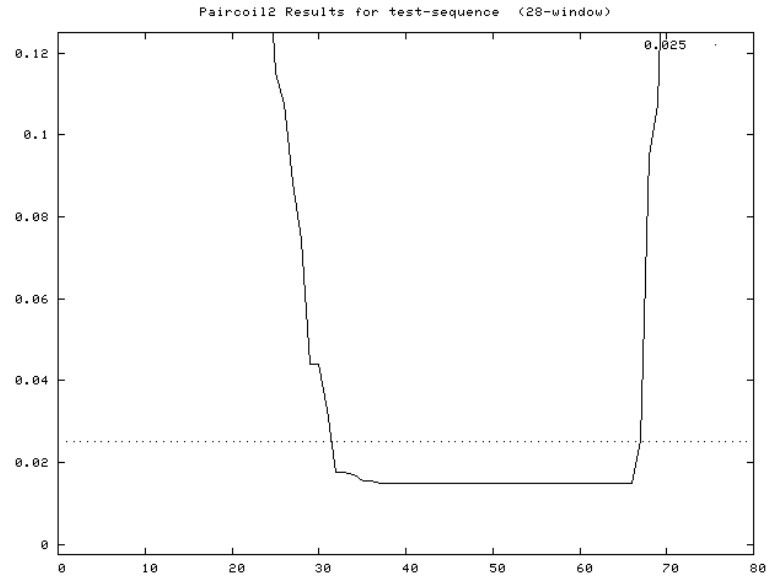
4 Preprocessing Predicted Coiled Coil Segments Using the Web interface

We expect PrOCoil to be used mainly for sequences the exact 3D structures of which are unknown (otherwise the correct oligomerization can be determined using SOCKET [**?**]). For structurally unresolved sequences, a coiled coil predictor must be used first to determine the coiled coil segments and their presumed heptad annotations. Currently the two most commonly used predictors for this task are **Marcoil** [**?**] and **PairCoil2** [**?**, **?**]. Both programs have their own output formats that do not comply with the format described in Section **??**. In order to ease integration with Marcoil and PairCoil2, the PrOCoil Web interface offers preprocessing tools that allow to use the outputs of Marcoil and PairCoil2 for processing with PrOCoil.

4.1 Processing PairCoil2 results

The PairCoil2 Web interface[3] produces the following kind of output for the Marcoil sample sequence:

P-score graph:



These Paircoil2 scores are available in the following formats:

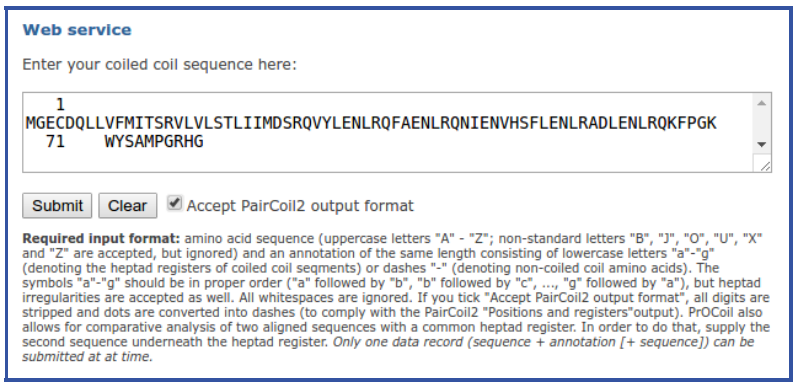
- Per-residue scores
- Positions and registers
- A graph in GIF format
- A graph in Postscript format

If you click the link “Positions and registers”, a text file with the predicted heptad annotation is displayed. Select the lines with the amino acids and heptad annotations in the following way:

```
Cutoff for scoring a coiled coil as positive was 0.03

1       test-sequence

        0.0149 @  37 : f

        123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
  1     MGECDQLLVFMITSRVLVLSTLIIMDSRQVYLENLRQFAENLRQNIENVHSFLENLRADLENLRQKFPGK
 71     WYSAMPGRHG

        123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
  1     ..............................abcdefgabcdefgabcdefgabcdefgabcdefg....
 71     ..........

 [0.0149@  32-  66:a;  End: 80]
```

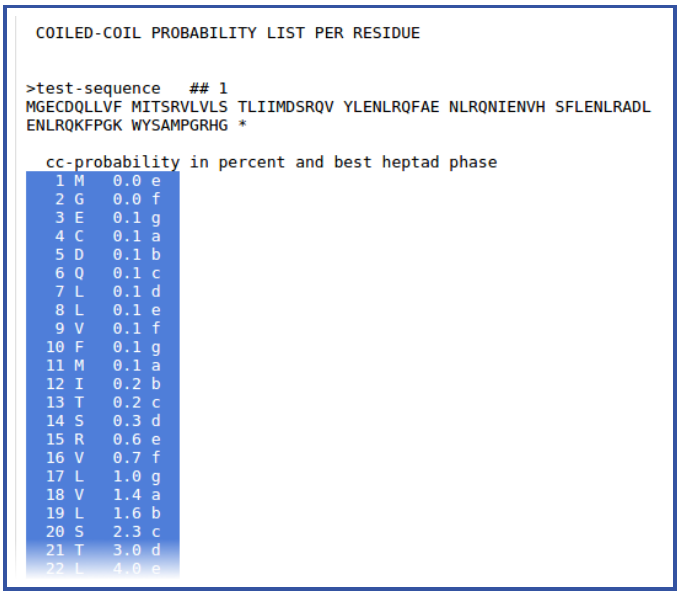
Now copy this selection into the input field of the PrOCoil Web interface. If you check “Accept PairCoil2 output format”, you can directly process the PairCoil2 output with PrOCoil:

[3] http://groups.csail.mit.edu/cb/paircoil2/paircoil2.html



4.2 Processing Marcoil results

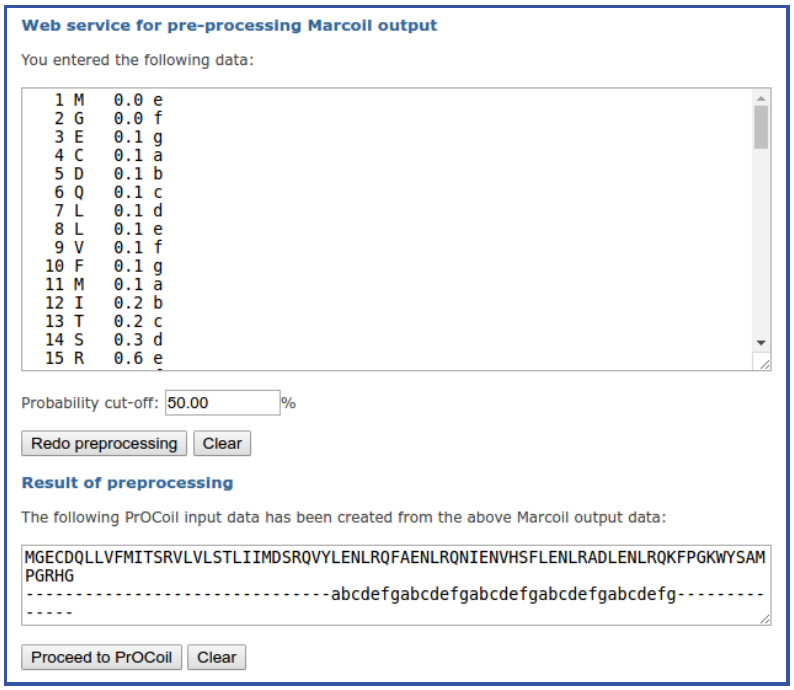
If you submit a sequence to the Marcoil Web interface,[4] the results are shown in your Web browser. These results cannot be processed by PrOCoil directly, since coiled coil segments must first be singled out by applying a probability cut-off threshold. This can be one as follows: first go to the “Prob List” tab and click “Export”, then a plain text file with results is shown as follows:



As in the above screenshot, select the four-column table with residue numbers, residues, probabilities, and most probable heptad positions. Then copy and paste it into the text field at the bottom of the PrOCoil Web page as follows:

[4] http://toolkit.tuebingen.mpg.de/marcoil

After clicking "Submit", an output page is displayed that shows the result of applying the chosen probability cut-off in a read-only field at the bottom:



If you are satisfied with the result, you can directly pass the data to PrOCoil by clicking "Proceed to PrOCoil". If you think you should have used a different threshold, your input is displayed in the input field on top of the page. Select a different threshold until the result meets your expectation and then pass the data to PrOCoil.

5 PrOCoil R Package

5.1 Installation

The PrOCoil R package (current version: 2.0.2) is available via Bioconductor. The simplest way to install the package is the following:

```
source("http://www.bioconductor.org/biocLite.R")
biocLite("procoil")
```

If you wish to install the package manually instead, you can download the package archive that fits best to your computer system from the Bioconductor homepage.

5.2 Getting started

To load the PrOCoil package, enter

```
library(procoil)
```

in your R session. If this command terminates without any error message or warning, you can be sure that the PrOCoil package has been installed successfully. If so, the PrOCoil package is ready for use now and you can start predicting the oligomerization of coiled coils.

As a first example, the following command makes a prediction for the GCN4 wild type already used above:

```
GCN4wt <- predict(PrOCoilModel,
                  "MKQLEDKVEELLSKNYHLENEVARLKKLV",
                  "abcdefgabcdefgabcdefgabcdefga")
```

The object ‘`PrOCoilModel`’ is an object of class ‘`CCModel`’ in which the PrOCoil model is stored. Since `predict()` is a generic function that determines the function to call from the type of the first argument, it is essential that you provide an object of class ‘`CCModel`’ as first argument of `predict()`. For more information about ‘`CCModel`’ objects, enter `?CCModel`.

Note that the sequence need not be a plain character string. Firstly, multiple sequences can be supplied at a time as a character vector, but also via objects of several other classes. Heptad registers need not be supplied via the ‘`reg`’ argument. Instead, they can be attached to the sequence objects via attributes or metadata (see **??** for details).

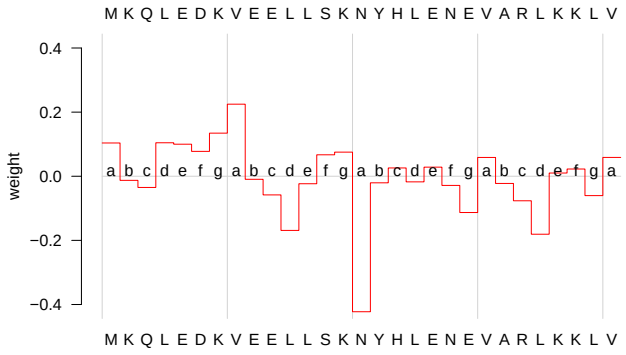
As mentioned in Section **??**, the sequence and the register annotation must have equal lengths. The function `predict()` creates an object of class ‘`CCProfile`’ in which prediction results along with various additional information is stored. To obtain more information about this class, enter `?CCProfile`. Basic information about the result can be displayed by `show(GCN4wt)` or simply by entering the name of the object:

```
GCN4wt

## An object of class  "CCProfile"
##
## Sequence:
##
##   A AAVector instance of length 1
##     width seq
## [1]    29 MKQLEDKVEELLSKNYHLENEVARLKKLV
##
## gappy pair kernel: k=1, m=5, annSpec=TRUE
##
## Baseline:  0.03698699
##
## Profile:
##           Pos 1        Pos 2            Pos 28       Pos 29
## [1]  0.140762197  0.024184153  ... -0.023390414  0.095688066
##
##
## Predictions:
##           Score  Class
## [1] -0.158713692  dimer
```

The discriminant value and its interpretation are the same as described in Section **??** above.

A prediction profile can be plotted simply as follows:

```
plot(GCN4wt)
```



The `plot()` function for '`CCProfile`' objects provides various ways for customizing the plot and for writing directly to graphics files (see also **??** and the documentation available via `?plot.CCProfile`).

5.3 Predictions for non-trimmed sequences containing coiled coil segments

Like the Web version described above, the R package is also capable of handling sequences that contain non-coiled-coil sub-sequences. If the heptad annotation contains at least one dash "-", `predict()` first extracts all coiled coil segments, i.e. all contiguous sub-sequences with no dashes in the heptad annotation. Then all these coiled coil segments are analyzed independently and the results are returned as a list, the components of which are the prediction results of the coiled coil segments in the order they appear in the sequence. Let us consider the Marcoil sample sequence again:

```
res <- predict(PrOCoilModel,
"MGECDQLLVFMITSRVLVLSTLIIMDSRQVYLENLRQFAENLRQNIENVHSFLENLRADLENLRQKFPGKWYSAMPGRHG",
"------------------------------abcdefgabcdefgabcdefgabcdefgabcdefg--------------")
```

The returned object '`res`' is a '`CCProfile`' object that contains the predictions and profiles of all coiled coil segments that were found in the sequences. In the given example, this is just one segment:

```
res


## An object of class  "CCProfile"
##
## Sequence:
##
##   A AAVector instance of length 1
##     width seq
## [1]    35 LENLRQFAENLRQNIENVHSFLENLRADLENLRQK
##
## gappy pair kernel: k=1, m=5, annSpec=TRUE
##
## Baseline:  0.03064637
##
## Profile:
##           Pos 1        Pos 2            Pos 34       Pos 35
## [1]  0.031568559 -0.017609488  ...  0.025883765  0.028083833
##
##
## Predictions:
##           Score  Class
## [1]  0.201721022 trimer
```

If the untrimmed sequences were unnamed, the segments will also remain unnamed in the resulting '`CCProfile`' object. However, if the sequences were named, the segments are named according to a specific strategy. This strategy is best explained by the following example: suppose we have three sequences that are named '`S1`', '`S2`', and '`S3`', and assume that sequence '`S1`' contains one coiled coil segment, '`S2`' contains three coiled coil segments, and '`S3`' contains no coiled coil

segment. Then the resulting 'CCProfile' object will contain predictions and profiles of a total of four coiled coil segments whose names are 'S1.1', 'S2.1', 'S2.2', and 'S2.3'.

The prediction profile must be plotted for each coiled coil segment separately. This can be done by accessing the respective entry in the CCProfile object explicitly:

```
plot(res[1])
```

5.4 Comparative mutation analysis

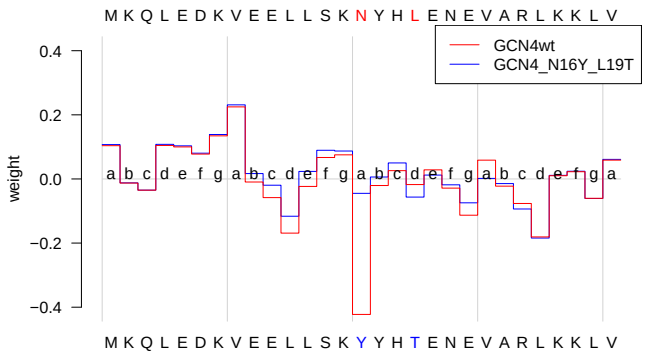
The PrOCoil R package allows not only to consider input sequences completely independently of one another, but also allows for overlaying profiles of aligned coiled coil segments. This feature is useful, for example, for studying effects of different mutations. As an example, we consider multiple mutations of GCN4:

```
GCN4mSeq <- c("GCN4wt"         ="MKQLEDKVEELLSKNYHLENEVARLKKLV",
              "GCN4_N16Y_L19T"="MKQLEDKVEELLSKYYHTENEVARLKKLV",
              "GCN4_E22R_K27E"="MKQLEDKVEELLSKNYHLENRVARLEKLV",
              "GCN4_V23K_K27E"="MKQLEDKVEELLSKNYHLENEKARLEKLV")
GCN4mReg <- rep("abcdefgabcdefgabcdefgabcdefga", 4)

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

## An object of class  "CCProfile"
##
## Sequences:
##
##   A AAVector instance of length 4
##     width seq                                               names
```

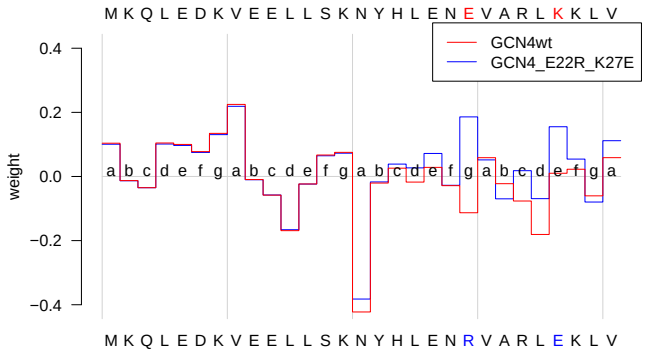
```
## [1]    29 MKQLEDKVEELLSKNYHLENEVARLKKLV             GCN4wt
## [2]    29 MKQLEDKVEELLSKYYHTENEVARLKKLV             GCN4_N16Y_L19T
## [3]    29 MKQLEDKVEELLSKNYHLENRVARLEKLV             GCN4_E22R_K27E
## [4]    29 MKQLEDKVEELLSKNYHLENEKARLEKLV             GCN4_V23K_K27E
##
## gappy pair kernel: k=1, m=5, annSpec=TRUE
##
## Baselines:  0.03698699 0.03698699 0.03698699 0.03698699
##
## Profiles:
##                       Pos 1        Pos 2           Pos 28       Pos 29
##         GCN4wt  0.140762197  0.024184153  ... -0.023390414  0.095688066
## GCN4_N16Y_L19T  0.144175109  0.024770521  ... -0.023957537  0.098008113
## GCN4_E22R_K27E  0.137580719  0.023637548  ... -0.042715408  0.148626188
## GCN4_V23K_K27E  0.141592659  0.024326834  ... -0.047527521  0.152960221
##
##
## Predictions:
##                            Score  Class
##              GCN4wt -0.158713692  dimer
##       GCN4_N16Y_L19T  0.420763995 trimer
##       GCN4_E22R_K27E  0.623458294 trimer
##       GCN4_V23K_K27E -0.500406810  dimer
```

The `plot()` function allows for plotting two profiles in one plot:

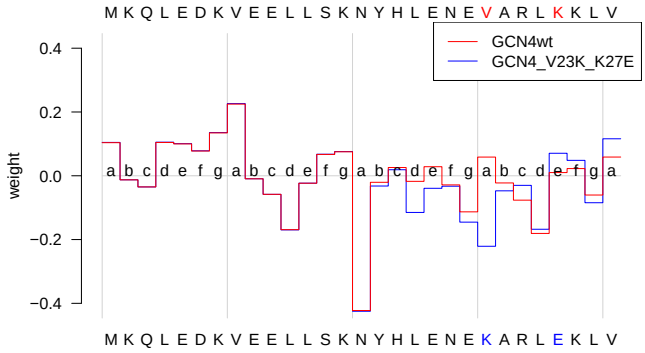
```
plot(GCN4mut[c(1, 2)])
```




```
plot(GCN4mut[c(1, 3)])
```




```
plot(GCN4mut[c(1, 4)])
```

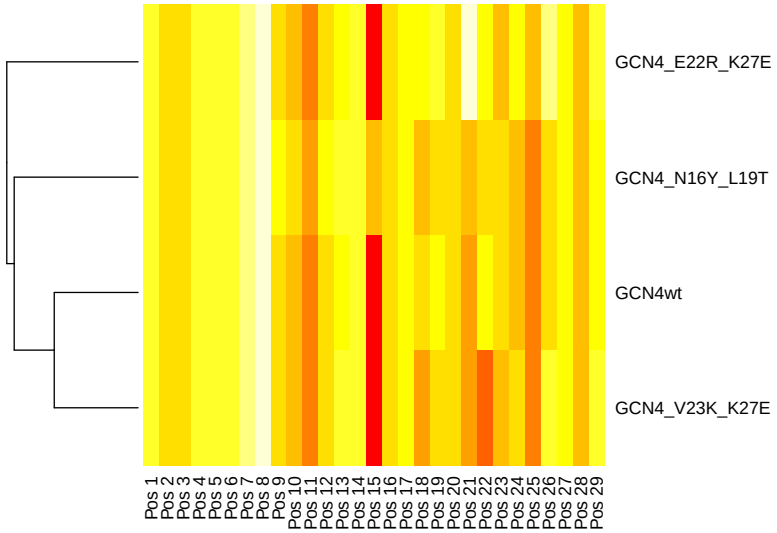


By default, the first profile is plotted in red and the second profile is plotted in blue. The first sequence is shown above the graph and the second sequence is shown below the graph. Black letters correspond to residues that are the same in the two sequences, whereas mutations are highlighted in the corresponding colors. More information on how to customize plots is available in **??** and the documentation available via `?plot.CCProfile`.

Note: Profile overlays also work if the two sequences are not equally long and/or if their heptad annotations are not aligned to each other. However, this hardly makes sense, so the overlay of profiles is primarily made for showing profiles of aligned sequences with matching heptad registers.

For three or more sequences, the profiles can also be visualized as heatmaps:

```
heatmap(GCN4mut)
```



In this view, profiles/sequences are even hierarchically clustered. The dendrogram is shown to the left of the heatmap. In the example above, obviously, the two trimers are clustered together as rows 1–2 and the two dimers are clustered together as rows 3–4. The note from above applies to heatmaps too: heatmaps make most sense for showing profiles of aligned sequences with matching heptad registers.

5.5 Miscellanea

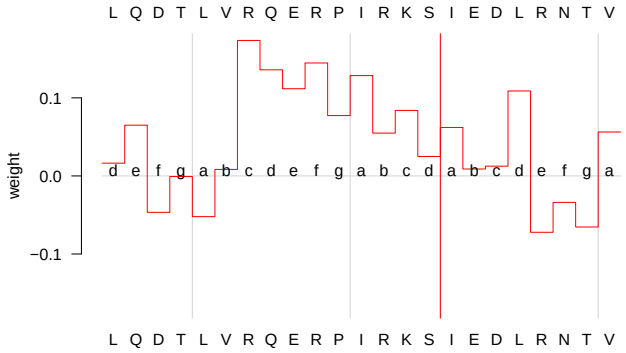
5.5.1 Processing predicted coiled coil segments with the R package

The PrOCoil R package itself is not able to process Marcoil or PairCoil2 output. Users who want to process prediction results obtained from these programs are recommended to use the PrOCoil Web interface to convert Marcoil or PairCoil2 output (see Section **??**) into the correct input format that can be processed by the PrOCoil R package.

5.5.2 Heptad irregularities

Some coiled coils occurring in nature have heptad irregularities, e.g. incomplete heptads in which an ‘a’ position follows after a ‘d’ position. PrOCoil can also process heptad annotations with such irregularities. In the profile plots created by the PrOCoil Web interface, such irregularities can only be seen in the register labels in the middle of the plot. In the profile plots created by the PrOCoil R package, heptad irregularities are additionally visualized by a vertical red line between the two positions that do not conform to the usual regular pattern.

```
plot(predict(PrOCoilModel,
             "LQDTLVRQERPIRKSIEDLRNTV",
             "defgabcdefgabcdabcdefga"))
```



5.5.3 Alternative models

Trimers occur less frequently than dimers. Correspondingly, there were more dimers and trimers in the data set used for training the default model 'PrOCoilModel'. Since this model was optimized for (standard) classification accuracy, it tends to concentrate on classifying the larger class — dimers — properly. That is why the model performs better in terms of specificity/true negative rate (correctly classified dimers) than in terms of sensitivity/true positive rate (correctly classified trimers). In case one wants to attribute equal importance to sensitivity and specificity, we have trained a second model that is optimized for *balanced accuracy*, i.e., the average of sensitivity and specificity (see publication supplement of [**?**]). This model can be used simply by calling `predict()` with 'PrOCoilModelBA' as first argument. Like the standard model 'PrOCoilModel', the alternative model 'PrOCoilModelBA' is automatically available once the PrOCoil R package has been loaded.

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. In order to see how such a model file should be organized, the two models included in the PrOCoil R package are available for download at:

```
http://www.bioinf.jku.at/software/procoil/PrOCoilModel_v2.CCModel
http://www.bioinf.jku.at/software/procoil/PrOCoilModelBA_v2.CCModel
```

It is also possible to read them directly using the `readCCModel()` function:

```
URL <- "http://www.bioinf.jku.at/software/procoil/PrOCoilModel_v2.CCModel"
myModel <- readCCModel(URL)
```

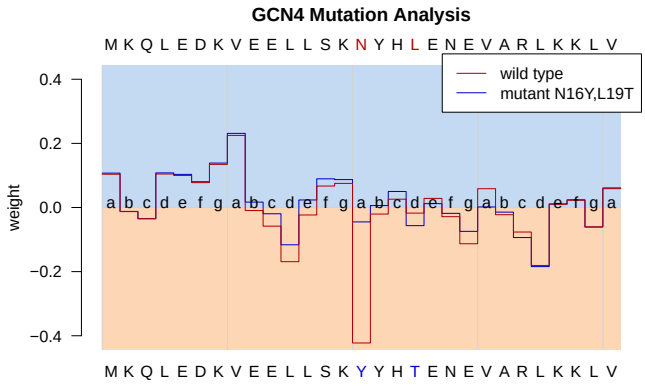
The present version of the R package includes newer models than the ones reported in [**?**] (which were used in PrOCoil versions before 2.0.0). However, for backward compatibility, these models can still be downloaded and used with the PrOCoil R package:

```
http://www.bioinf.jku.at/software/procoil/PrOCoilModel_v1.CCModel
http://www.bioinf.jku.at/software/procoil/PrOCoilModelBA_v1.CCModel
```

5.5.4 Customizing and saving plots

The PrOCoil R package provides several opportunities to customize plots. For more detailed documentation, see `?plot.CCProfile`. We simply provide an example here that uses legends, shading, custom coloring of profiles, and a plot header:

```
plot(GCN4mut[c(1, 2)], legend=c("wild type", "mutant N16Y,L19T"),
     col=c(rgb(0.7, 0, 0), rgb(0, 0, 0.8)), main="GCN4 Mutation Analysis",
     shades=c(rgb(0.77, 0.85, 0.95), rgb(0.99, 0.84, 0.71)))
```



R graphics have a fixed size and aspect ratio by default. That is why PrOCoil prediction profiles appear quite stretched for shorter and constricted for longer sequences. When exporting profile plots to graphics files, we therefore recommend to adjust the graphics dimensions in order to achieve a consistent and optically pleasing apperance:

- Use a fixed value for the *height* of the graphics device; we recommend a value between 5” and 7” for vector formats (PDF and (encapsulated) PostScript) and a few hundred pixels for bitmap image formats (BMP, JPEG, TIFF, PNG).
- Scale the *width* according to the length of the sequence; we recommend 1/24 of the graphic’s height per residue as a rule of thumb.

Examples:

```
pdf(file="GCN4wt.pdf", height=6,
    width=max(nchar(sequences(GCN4wt))) * 6 / 24)
plot(GCN4wt)
dev.off()

bmp(file="GCN4wt.bmp", height=480,
    width=max(nchar(sequences(GCN4wt))) * 480 / 24)
plot(GCN4wt)
dev.off()
```

5.5.5 Alternative ways of supplying heptad registers to `predict()`

As mentioned briefly in Subsection **??**, the `predict()` function requires a string that contains the heptad register annotation of the sequence to be classified. Normally, this string is passed as the third argument '`reg`'. However, the following alternatives exist as well:

- If the second argument '`seq`' is a character vector, the heptad register can also be supplied via the attribute '`reg`':

```
GCN4wtSeq1 <- "MKQLEDKVEELLSKNYHLENEVARLKKLV"
attr(GCN4wtSeq1, "reg") <- "abcdefgabcdefgabcdefgabcdefga"
res <- predict(PrOCoilModel, GCN4wtSeq1)
```

- If the second argument '`seq`' is a single amino acid sequence given as an '`AAString`' object, the heptad register can be specified via a metadata component '`reg`':

```
GCN4wtSeq2 <- AAString("MKQLEDKVEELLSKNYHLENEVARLKKLV")
metadata(GCN4wtSeq2)$reg <- "abcdefgabcdefgabcdefgabcdefga"
res <- predict(PrOCoilModel, GCN4wtSeq2)
```

- If the second argument '`seq`' is a set of amino acid sequences given as an '`AAStringSet`' or '`AAVector`' object, the heptad registers can be specified via a metadata component '`reg`' as above or via the annotation metadata:

```
GCN4mSeq2 <- AAStringSet(GCN4mSeq)
annotationMetadata(GCN4mSeq2, annCharset="abcdefg") <- GCN4mReg
res <- predict(PrOCoilModel, GCN4mSeq2)
```

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. If no heptad register is found in any of the places mentioned above, `predict()` quits with an error message.

5.6 Upgrading from a version prior to 2.0.0

Version 2.0.0 of the PrOCoil R package has not only brought improved prediction models (see **??**), but also several changes to the functionality of the package. While the basic functionality of the `predict()` function and the `plot()` function remained unchanged, users who upgrade from versions prior to 2.0.0 should take the following caveats into account:

1. The classes '`CCModel`' and '`CCProfile`' have changed signficantly internally. Objects saved with versions prior to 2.0.0 cannot be used with version 2.0.0 or newer.

2. Calling `predict()` for a sequence that contains multiple coiled coil segments no longer results in a list of '`CCProfile`' objects, but will result in a single '`CCProfile`' object that contains the predictions for all segments. '`CCProfile`' objects are now subsettable to extract a subset of predictions/profiles.

3. The `plot()` method with signature ('`CCProfile`', '`CCProfile`') is no longer available for plotting two prediction profiles together. Instead, the user has to call `predict()` for a sequence object that contains two or more sequences. Consequently, all predictions are returned in a single '`CCProfile`' object (see **??** for examples).

4. The argument '`legend.pos`' of the `plot()` method has been renamed to '`legendPos`' for the sake of compatibility with the KeBABS package. The argument '`rng`' is no longer available.

5. The `readCCModel()` function expects patterns in a slightly different format now: instead of patterns that only refer to the heptad position of the first amino acid of the pattern (e.g. '`N..La`'), `readCCModel()` now expects the patterns to specify also the heptad position of the second amino acid in the pattern (e.g. '`N..La..d`'; see also **??**).

6 More Details About PrOCoil

6.1 How the prediction works

PrOCoil's classification models are based on support vector machines [**?**, **?**, **?**, **?**] with a kernel designed specifically for coiled coil classification:

$$k(x, y) = \sum_{p \in P} N(p, x) \cdot N(p, y)$$

In this formula, x and y are the two input sequences that are to be compared, P is a set of coiled coil-specific patterns, and $N(p, x)$ is the number of occurrences of pattern p in sequence x.

PrOCoil uses the following set of patterns P: pairs of amino acids at fixed heptad 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 finally an aligned sequence starting and ending with a letter '`a`'–'`g`' denoting the heptad register position of the first and the 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 an 'L' (Leucine) at the next 'd' position. For instance, the GCN4 wild type has one occurrence of this pattern:

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

So, obviously, PrOCoil considers pair interactions of amino acids at given heptad positions which are no more than m positions apart. The kernel described above resembles the spatial sample kernel [**?**] (however, with a heptad-specific property) and the kernel described in [**?**] (however, considering interactions within one sequence and not restricting to a particular subset of interactions).

The two models included in PrOCoil further employ kernel normalization [**?**] to correct for different sequence lengths. This means that the support vector machines have been trained with the kernel

$$k'(x,y) = \frac{k(x,y)}{\sqrt{k(x,x) \cdot k(y,y)}},$$

where $k(.,.)$ is the *coiled coil kernel* described above.

Using an explicit representation of the feature mapping underlying the kernel, the support vector machines have been transformed into linear classifiers on sequence features (cf. [**?**] for details). Thus, PrOCoil's prediction models consist of weights for specific sequence features (i.e. *patterns*) and a constant offset b. Let us assume that x denotes a new coiled coil sequence. Without kernel normalization, the discriminant function value of the new sequence x is given as

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

where b is the constant offset of the support vector machine and $w(p)$ is the weight of pattern p.

If kernel normalization is employed, the following, slightly more complicated, representation is obtained:

$$f(x) = b + \left(\sum_{p \in P} N(p,x) \cdot w(p) \right) \Big/ \underbrace{\sqrt{\sum_{p \in P} N(p,x)^2}}_{=R(x)}$$

Obviously, $R(x)$ is the value that corrects for the sequence length. The longer the sequence, the larger $R(x)$.

6.2 PrOCoil's built-in models

As already mentioned above, PrOCoil provides a default model that is optimized for classification accuracy (object '`PrOCoilModel`' in the PrOCoil R package) and an alternative model that is optimized for balanced accuracy (object '`PrOCoilModelBA`'). Both models have been trained on the same training set which consisted of verified coiled coils from the RCSB Protein Data Bank

(PDB)[5] [?] enriched by putatively similar sequences that were obtained by BLAST [?] in conjunction with Marcoil [?]. The default model was created with the KeBABS package [?] using the coiled coil kernel mentioned above (in the lingo of the KeBABS package called *annotation-specific gappy pair kernel*) and the frontend to LibLineaR package, which provides an R interface to LIBLINEAR [?]. The current models (since Version 2.0.0) use the normalized coiled coil kernel with $m = 5$ and a penalty parameter of $C = 2$. The alternative modes was created with PSVM [?] using the normalized coiled coil kernel with $m = 8$, class balancing, and regularization parameters $C = 8$ and $\varepsilon = 0.8$.

In the PrOCoil R package, all parameters describing a model are stored in the corresponding '`CCModel`' object:

```
PrOCoilModel

## An object of class  "CCModel"
##
## Model parameters:
##  coiled coil kernel with m=5 and kernel normalization
##  offset b= -1.073
##
## Feature weights:
##    1.6363 ... L...Vd...a
##    1.5382 ... R....Eg....e
##    1.2903 ... R.Ec.e
##    1.2284 ... E..Ve..a
##    1.2040 ... I...Id...a
##       ... ... ...
##   -1.1330 ... K..La..d
##   -1.2192 ... E.Ec.e
##   -1.2290 ... L..Ld..g
##   -1.4273 ... L...Nd...a
##   -1.7811 ... N..La..d

weights(PrOCoilModel)["N..La..d"]

##  N..La..d
## -1.781079

PrOCoilModelBA

## An object of class  "CCModel"
##
## Model parameters:
##  coiled coil kernel with m=8 and kernel normalization
```

[5] http://www.rcsb.org/

```
##  offset b= -2.696
##
## Feature weights:
##    8.8966 ... L...Vd...a
##    5.7107 ... E..Ie..a
##    5.6860 ... I...Id...a
##    5.5755 ... R....Eg....e
##    5.4611 ... L..La..d
##       ... ... ...
##   -3.7145 ... A.Lb.d
##   -3.7442 ... K........Lb........d
##   -3.7636 ... N..La..d
##   -3.8979 ... E....Lc....a
##   -4.6041 ... L...Id...a
```

```
weights(PrOCoilModelBA)["N..La..d"]
```

```
##  N..La..d
## -3.763633
```

In both models, the pattern weights are sorted decreasingly, i.e., from most trimeric to most dimeric. Thus, the user also has easy access to the most indicative patterns. Here we provide an example how to extract the 25 most trimeric (i.e. the first 25 patterns in the list) and the 25 most dimeric patterns (i.e. the last 25 patterns in the list) from the PrOCoilmodel:

```
noP <- length(weights(PrOCoilModel))
names(weights(PrOCoilModel))[1:25]
```

```
##  [1] "L...Vd...a"   "R....Eg....e" "R.Ec.e"       "E..Ve..a"
##  [5] "I...Id...a"   "D.Vf.a"      "M...Id...a"   "E.Ke.g"
##  [9] "R..Ag..c"     "I...Na...e"  "A.Ib.d"       "E...Ig...d"
## [13] "R....Ic....a" "N...Id...a"  "IKde"         "SQbc"
## [17] "M..Ed..g"     "I..Ia..d"    "S..Qf..b"     "K..Qg..c"
## [21] "K..Kf..b"     "K...Ae...b"  "L....Ra....f" "RQcd"
## [25] "E.Ig.b"
```

```
names(weights(PrOCoilModel))[noP:(noP - 24)]
```

```
##  [1] "N..La..d"       "L...Nd...a"     "L..Ld..g"       "E.Ec.e"
##  [5] "K..La..d"       "A.Lb.d"         "L...Rd...a"     "L...Id...a"
##  [9] "A...Ef...c"     "K.If.a"         "N.....Ea.....g" "RLcd"
## [13] "I...La...e"     "LKab"           "L...Kd...a"     "E...Lg...d"
## [17] "NYab"           "A...Eb...f"     "E..Kf..b"       "QLfg"
## [21] "E.Lf.a"         "VKab"           "Q...Ec...g"     "Q.Ee.g"
## [25] "L....Yd....b"
```

6.3 How prediction profiles are obtained

Regardless of whether kernel normalization is employed or not, the essential component of the discriminant function is the sum

$$\sum_{p \in P} N(p, x) \cdot w(p).$$

It is obvious that every match of a pattern p contributes $w(p)$ to the sum. In order to find out the extent to which each residue contributes to the final classification, we can rewrite the sum as

$$\sum_{p \in P} N(p, x) \cdot w(p) = \sum_{i=1}^{L} c_i(x), \tag{1}$$

where L is the length of the sequence x and $c_i(x)$ is the contribution of the i-th residue of x. The contribution $c_i(x)$ can simply be computed as half the sum of weights of patterns matching the i-th residue.[6] The weights $c_i(x)$ (or $c_i(x)/R(x)$ in case kernel normalization is employed) can be understood as a profile that can be plot over the sequence. We have already introduced these values as *prediction profiles* above. In order to have a unified terminology, let us denote the prediction profiles as

$$s_i(x) = \begin{cases} c_i(x)/R(x) & \text{if kernel normalization is employed,} \\ c_i(x) & \text{otherwise.} \end{cases}$$

A positive values $s_i(x)$ indicates that the i-th residue is participating more strongly in patterns that are indicative for trimers. A negative values $s_i(x)$ indicates that the i-th residue is participating more strongly in patterns that are indicative for dimers. A value $s_i(x)$ or zero or close to zero either means that the i-th residue is not participating in any indicative patterns or that it participates in dimer and trimer patterns the contributions of which compensate/cancel each other.

PrOCoil computes prediction profiles each time it computes a prediction. It is clear that we can recover the discriminant value $f(x)$ as follows:

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

If $b \neq 0$, which is the normal case, the values $s_i(x)$ do not provide enough information to infer the final classification from the prediction profile. In order to facilitate a more sensible analysis, let us reformulate the above equality as

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

Hence, if we plot the profile values $s_i(x)$ along with a horizontal line at $-\frac{b}{L}$, we can recover the discriminant value $f(x)$ as the difference of the area above the $-\frac{b}{L}$ minus the area below $-\frac{b}{L}$. The value $-\frac{b}{L}$ exactly corresponds to the "base line" mentioned above.

[6] As all patterns match two distinct residues, each weight must be split to the two residues in order to ensure that the equality (**??**) holds.

7 Acknowledgments

Several colleagues have contributed to PrOCoil and its implementation: **Carsten C. Mahrenholz**, **Ingrid G. Abfalter**, **Rudolf Volkmer**, and **Sepp Hochreiter**, who are also the co-authors of the first PrOCoil paper [**?**], have contributed to the inception and success of PrOCoil with their biochemical knowledge, machine learning expertise, and by putting together the first coiled data set that was published along with [**?**] and used for training he original PrOCoil models.

The data set underlying the PrOCoil 2.0.0 models was created by **Annette Jacyszyn** who also performed the hyperparameter selection and training of the new models.

Last but definitely not least, the new version of the package would have been completely impossible without **Johannes Palme**'s amazing KeBABS package.

8 How to Cite PrOCoil

If you use PrOCoil for research that is published later, you are kindly asked to cite it as follows:

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

To obtain this reference in BibTEX format, you can enter the following into your R session:

```
toBibtex(citation("procoil"))
```

References

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.

[2] B. Berger, D. B. Wilson, E. Wolf, T. Tonchev, M. Milla, and P. S. Kim. Predicting coiled coils by use of pairwise residue correlations. *Proc. Natl. Acad. Sci. USA*, 92:8259–8263, 1995.

[3] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The Protein Data Bank. *Nucleic Acids Res.*, 28(1):235–242, 2000.

[4] U. Bodenhofer, K. Schwarzbauer, M. Ionescu, and S. Hochreiter. Modeling position specificity in sequence kernels by fuzzy equivalence relations. In J. P. Carvalho, D. Dubois, U. Kaymak, and J. M. C. Sousa, editors, *Proc. Joint 13th IFSA World Congress and 6th EUSFLAT Conference*, pages 1376–1381, Lisbon, July 2009.

[5] C. J. M. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2:121–167, 1998.

[6] C. Cortes and V. N. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1986.

[7] M. Delorenzi and T. Speed. An HMM model for coiled-coil domains and a comparison with PSSM-based predictions. *Bioinformatics*, 18(4):617–625, 2002.

[8] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: a library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, 2008.

[9] J. H. Fong, A. E. Keating, and M. Singh. Predicting specificity in bZIP coiled-coil protein interactions. *Genome Biol.*, 5:R11, 2004.

[10] S. Hochreiter and K. Obermayer. Support vector machines for dyadic data. *Neural Comput.*, 18:1472–1510, 2006.

[11] P. Kuksa, P.-H. Huang, and V. Pavlovic. A fast, large-scale learning method for protein sequence classification. In *8th Int. Workshop on Data Mining in Bioinformatics*, pages 29–37, Las Vegas, NV, 2008.

[12] C. C. Mahrenholz, I. G. Abfalter, U. Bodenhofer, R. Volkmer, and S. Hochreiter. Complex networks govern coiled coil oligomerization — predicting and profiling by means of a machine learning approach. *Mol. Cell. Proteomics*, 10(5):M110.004994, 2011.

[13] A. V. McDonnell, T. Jiang, A. E. Keating, and B. Berger. Paircoil2: improved prediction of coiled coils from sequence. *Bioinformatics*, 22(3):356–358, 2006.

[14] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Trans. Neural Networks*, 12(2):181–201, 2001.

[15] J. Palme, S. Hochreiter, and U. Bodenhofer. KeBABS: an R package for kernel-based analysis of biological sequences. *Bioinformatics*, 31(15):2574–2576, 2015.

[16] B. Schölkopf and A. J. Smola. *Learning with Kernels*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2002.

[17] B. Schölkopf, K. Tsuda, and J.-P. Vert, editors. *Kernel Methods in Computational Biology*. MIT Press, Cambridge, MA, 2004.

[18] J. Walshaw and D. N. Woolfson. SOCKET: a program for identifying and analysing coiled-coil motifs within protein structures. *J. Mol. Biol.*, 307(5):1427–1450, 2001.