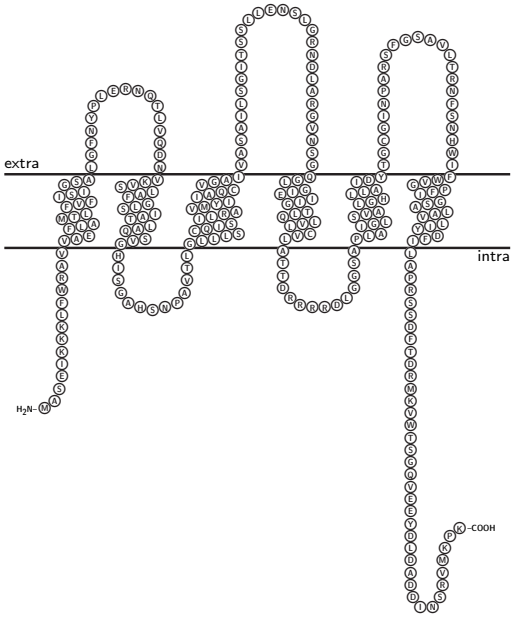
The TEXtopo package

for shaded membrane protein topology plots*

Eric Beitz†

v1.5; 2011/06/02



*Please cite: Eric Beitz (2000), TEXtopo: shaded membrane protein topology plots in LaTeX 2ε. *Bioinformatics* **16**: 1050–1051.

†University of Kiel, Pharmaceutical Chemistry, Gutenbergstrasse 8, D-24118 Kiel, Germany; send electronic mail to ebeitz@pharmazie.uni-kiel.de; for further information, updates and on-line documentation see my homepage at www.pharmazie.uni-kiel.de/chem/Prof_Beitz/biotex.html

Contents

1 Package Overview

After `textopo.ins` is run through TEX the following files should appear in the directory:

`textopo.sty`	the style file with all TEXtopo commands
`textopo.def`	an example parameter file with the standard parameter settings
`AQPpro.MSF`	an example protein alignment (`MSF`-format)
`AQPpro1.shd`	shading information calculated from the file `AQPpro.MSF`
`AQP2spec.ALN`	a further protein alignment (minimal `ALN`-file)
`AQP1.phd`	secondary structure information (PHD-format)
`AQP1.hmm`	secondary structure information (HMMTOP-format)
`AQP1.tpo`	secondary structure information extracted from `AQP1.phd`
`AQP1.SP`	protein database entry (SwissProt-format)
`AQP1.swp`	sequence and feature information extracted from `AQP1.SP`
`biotex.sty`	this style file organizes the interaction with TEXshade, see 3.3.5

The alignment file examples as well as the topology data file are needed for TEXing this documentation and can serve as illustrations for the `MSF` and `ALN` file format.
The following subsections give an overview of the capabilities of the TEXtopo package. All commands are described in detail later on.

1.1 Version History

v1.5 2011/06/02

Compatibility with the current TEXshade version was re-established.

v1.4 2005/02/14

A new topology input format has been implemented: HMMTOP, see 3.1.2. A rotation issue with classical flat helical wheels has been addressed and some minor bugs have been fixed.

v1.3 2002/04/15

The unnecessary restriction to the DVIPS driver for `color.sty` has been removed[1]. Any color.sty compatible driver option can be given with the `\usepackage{textopo}` call and is then passed to the `color` package. Further, `rotating.sty` is no longer needed. The maximal helix length has been increased to 36 aa. Introduction of two new helical wheel styles (`net` and `wheel`) and the display of the hydrophobic moment. Corresponding commands: `\helixstyle`, `\showmoment`, `\hidemoment`, `\momentcolor`, `\scalemoment`, `\Hmean`, `\muH`, `\muHmean`, `\mudelta`).

v1.2 2001/03/09

Several new commands were introduced: `\movelegend` for a free relocation of the figure legend, `\footloop` adds a foot to a specified loop and thus keeps the distance between the transmembrane domains small, `\broadenmembrane` and `\thickenmembrane` allow one to change the dimensions of the membrane. In the helicalwheel environment number series can be written with a dash, e. g. `{1-5}` instead of `{1,2,3,4,5}`. In commands that move labels the new position can be given in x/y-values besides the ⟨*direction*⟩ and ⟨*distance*⟩ parameters.

v1.1 2000/07/12

One major improvement was achieved by changing the handedness of the transmembrane helices to be consistently left-handed. See the cover figure! The documentation now contains instructions where to find basic LaTeX documents and how to increase TeX's parameter settings.

v1.0a 2000/05/16 – v1.0c 2000/06/03

Minor corrections of the documentation and bug fixes in the `\place`, `\addtagtoNterm` and `\addtagtoCterm` commands. Improvement of the TeXshade compatibility.

v1.0 2000/03/18

First release.

[1] As suggested by Eckhart Guthöhrlein.

1.2 LaTeX basics

1.2.1 Typesetting documents with LaTeX

In order to use any of the macros provided by the BibTeX-project (see 3.3.5) efficiently a basic understanding of the TeX typesetting system and its usage is required. Several books are available on this topic, but a rather quick and easy introduction is the *Not so short introduction to LaTeX*. This document is available from all Comprehensive TeX Archive Network (CTAN) servers, e. g. from `ftp://ftp.dante.de/pub/tex/documentation/lshort/`, in many different languages and formats besides LaTeX, such as PostScript and on-line viewable PDF. I also put a link from the BibTeX (TeXshade/TeXtopo) homepage to the document collection (`http://homepages.uni-tuebingen.de/beitz/biotex.html`).

1.2.2 Memory shortness when using TeXtopo

If you are using TeXtopo to plot topologies of larger proteins (> 600 residues), LaTeX will probably stop compiling and quit with one of the following messages: `! TeX capacity exceeded, sorry [main memory size=384000]` or `! TeX capacity exceeded, sorry [stack size=300]`.
TeX allocates space for different kinds of internal variables. Plotting topologies of big membrane proteins needs lots of memory, usually more than for typesetting plain text. Thus, the parameter settings of a standard TeX installation might not be sufficient for certain plotting projects. This becomes obvious when TeX complains about insufficient memory by displaying error messages and the setting process is interrupted. There is no reason to be concerned. The parameters can be set by hand. Unfortunately, each TeX system hides its default parameter file in a different place in the system.
In the following, an excerpt from a FAQ-list to TeXshade, an alignment setting macro for LaTeX, is added. This explains how to increase the settings in OzTeX for the Macintosh, MikTeX for Windows and teTeX for *NIX TeX distributions. Please contribute to this list!

1. **OzTeX 4.0 for the Macintosh:**

   Find the file 'OzTeX:TeX:Configs:Default'. This file contains all memory settings. Look for the section '% TeX parameters' and increase the values that TeX complains about during the run. You will have to restart OzTeX before the changes are active.

For older versions of OzTEX the configuration file has the same name but the path is somewhat different.

2. **teTEX for *NIX:** (contributed by Joerg Daehn)

   Find the file: '/usr/share/texmf/web2c/texmf.cnf' or use `locate texmf.cnf` at the command prompt to find it.

   Login as super user. Backup 'texmf.cnf' in case you destroy something and then open the 'texmf.cnf' file in your favorite text editor and use its search function to locate `main_memory`. This variable is set to 384000. Change this to some higher value, i.e. 4000000 (works fine for me!). The total amount of memory should not exceed 8000000, so check the other values in that section.

   Next, you want to change the stack size. Search for `stack_size`. This will be set to 300. I changed it to 4000 and it works fine.

   There might be complains by TEX about further specific parameters such as `stack_size`. You find all those in the same file.

   After this you have to run 'texconfig init'.

   Logout as root.

   After this all should be set for large plots. Happy TEXing!

   The information on how to achieve this was derived from a mail in the teTEX mail archive. The original question was posted by Pascal Francq and answered by Rolf Nieprasch.

3. **MiKTEX for Windows:**

   The MiKTEX documentation describes very detailed how the memory settings can be changed. In brief, you must locate the configuration file 'miktex/config/miktex.ini'. In the [MiKTeX] section of this file you find all the parameters you need, e. g. `mem_min`, `mem_max`, `buf_size`, `stack_size` etc.

   It appears, that the standard settings of MiKTEX are bigger than that of other TEX installations, so it may not always be necessary to increase the values.

1.3 System requirements

TEXtopo requires at least LATEX 2_ε and `color.sty`. David Carlisle's `color.sty` is part of the Standard LATEX 'Graphics Bundle' [1]. This

package can be downloaded from any TEX archive, e.g. `ftp.dante.de`; usually it is already included in a comprehensive TEX installation.
The `color` style allows one to use several `[`⟨*options*⟩`]`, e. g. `dvips`, `pdftex` or `dviwin`. These provide the commands which different devices/programs need to display colored output. It is advisable to make yourself familiar with the `color.sty` manual. You should define a default driver in the file `color.cfg`. Since there is no direct call of `color.sty` by the user, the option can be stated when TEXtopo is loaded, see next subsection. If no option is stated the `DVIPS` driver will be loaded as was default before.
With the `[dvips]` option for example the output DVI-file can be converted to PostScript using the `DVIPS` program and can later be viewed or printed with the public domain `Ghostview` program which is available for almost all computer platforms. Further, more and more standard TEX viewers are to a certain extent PostScript compatible, e. g. OzTEX on the Macintosh. The option `pdftex` makes the conversion to a PDF file easy etc.
TEXtopo is compatible with TEXshade (version 1.3 or newer) which is a mighty alignment shading package for LATEX 2_ε. In combination with TEXshade the capability of TEXtopo is greatly enhanced, e. g. by the automatic application of calculated shading from protein alignments or shading due to functional properties, such as charge or accessible side chain area, see 1.5.

1.4 TEXtopo's environments

In order to make TEXtopo available for your document declare it in the document header section:

```
\usepackage[⟨option⟩]{textopo}
```

Make sure that the file '`textopo.sty`' is present in a directory searched by TEX (see the installation notes in the file '`textopo.txt`').
The ⟨*option*⟩ given here is passed to `color.sty` which handles the color commands for a particular output device, see previous subsection and the `color.sty` manual.
The package provides two new environments, i. e. the `textopo` and the `helicalwheel` environment. Both are described in greater detail below.

1.4.1 The `textopo` environment

This environment displays schematic topology plots of membrane proteins. TEXtopo can import sequence and topology data directly from PHD or HMMTOP predictions, SwissProt database files (see the example files `AQP1.PHD`, `AQP1.hmm` and `V2.SP` for their structure) or alignment files (MSF and ALN format; example files are also provided). When SwissProt files are used TEXtopo will automatically extract all the information about special domains, variations, mutations etc. from the database file and label the respective positions in the plot. Alternatively, one can manually enter the sequence and the positions of the membrane spanning domains within the environment. Based on this data TEXtopo produces a first plot. Then, the output can be further adjusted to one's needs by adding labels, special styles for the appearance of the residues, shading (automatic [see1.5] or manual) and legends.

The usage of the `textopo` environment is easy:

```
\begin{textopo}[⟨optional parameterfile⟩]
  further TEXtopo commands
\end{textopo}
```

In the optional parameter file (section 2) any TEXtopo command can be given in order to fix user specific settings. This option provides fast and consistent outputs. At least one command is necessary whithin the environment which loads the sequence and topology of the protein to be plotted, i. e. `\getsequence` [3.1.1] or `\sequence` + `\MRs` [3.1.5,3.1.4].

1.4.2 The `helicalwheel` environment

This second environment provides essentially the same functionality as `textopo`. Here, the output depicts the helical transmembrane spans as helical wheels (perspective or flat) or as helical nets which is basically the same kind of display as in the topology plots. One can choose all or a subset of transmembrane domains and set any desired order. Views from the outside onto the cell membrane or vice versa are possible.

The usage is as easy as this:

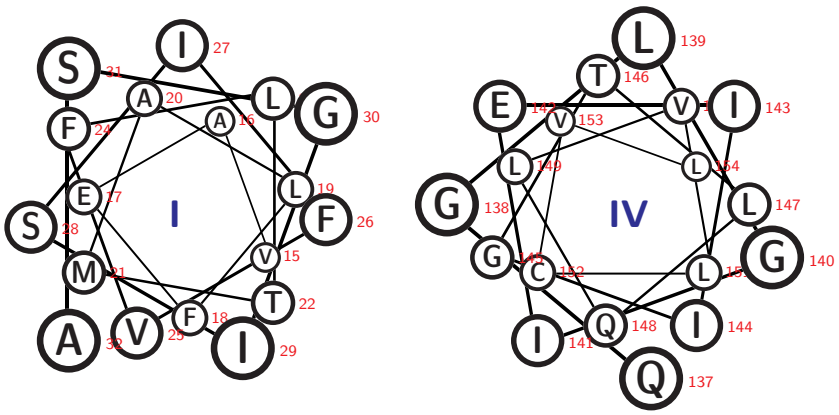


Figure 1: A basic helical wheel example.

```
\begin{helicalwheel}[⟨parameterfile⟩]{⟨helixlist⟩}
  further TEXtopo commands
\end{helicalwheel}
```

The optional parameter file can be used as described above. A list of the helices which are to be displayed is mandatory, i. e. `{1,2,3,4,5,6}`; or for displaying a subset with changed order: `{1,3,2,5}`. Series of numbers can be typed with a dash, e. g. `{1-3,9,5-7}`. Further, each helix number can be followed by an optional parameter which indicates an angle by which the transmembrane domain is rotated (only integer values). If a series of helix numbers are to be rotated by the same angle use the following scheme: `{1-3[90],4-6,7[135]}`.
A basic example shows helices 1 and 4 of an aquaporin and rotates helix no.4 by 50° (Fig.1):

```
\begin{helicalwheel}{1,4[50]}
  \getsequence{PHD}{AQP1.PHD}
\end{helicalwheel}
```

1.5 TEXshade (v1.3 and up) compatibility

TEXshade is a very comprehensive LATEX 2_ε package for displaying and shading protein and nucleotide alignments [2]. Package and documentation are available from the same source as the TEXtopo package, i. e. any CTAN site, e. g. `ftp.dante.de`, or from the TEXshade homepage `http://homepages.uni-tuebingen.de/beitz/tse.html`.
Since version 1.3 TEXshade provides its full functionality for TEXtopo, i. e. protein topology plots can be shaded automatically due to functional properties of the amino acid residues or to sequence conservation based on protein alignments. Most of the more than 100 TEXshade commands are applicable in addition to the commands provided by TEXtopo to customize the output or to define new shading modes.
A simple example is shown in Fig.2. It loads the sequence and topology data from a `PHD` file and applies shading calculated from an alignment in the `MSF` format.

```
\begin{textopo}
  \getsequence{PHD}{AQP1.phd}
  \applyshading{similar}{AQPpro.MSF}
  \allmatchspecial
\end{textopo}
```

Shading can also be applied to helical wheels as shown in Fig.3:

```
\begin{helicalwheel}{1-4}
  \getsequence{PHD}{AQP1.PHD}
  \applyshading{functional}{chemical}
\end{helicalwheel}
```

1.6 Customization of the output

The previously shown basic outputs may not be satisfactory enough in terms of flexibility, additional shading, or application of labels. Therefore TEXtopo provides commands which enable the user to modify and refine the plot in many ways.
Special domains in the protein sequence can be highlighted by the use of shading colors or of squares and diamonds representing the residues

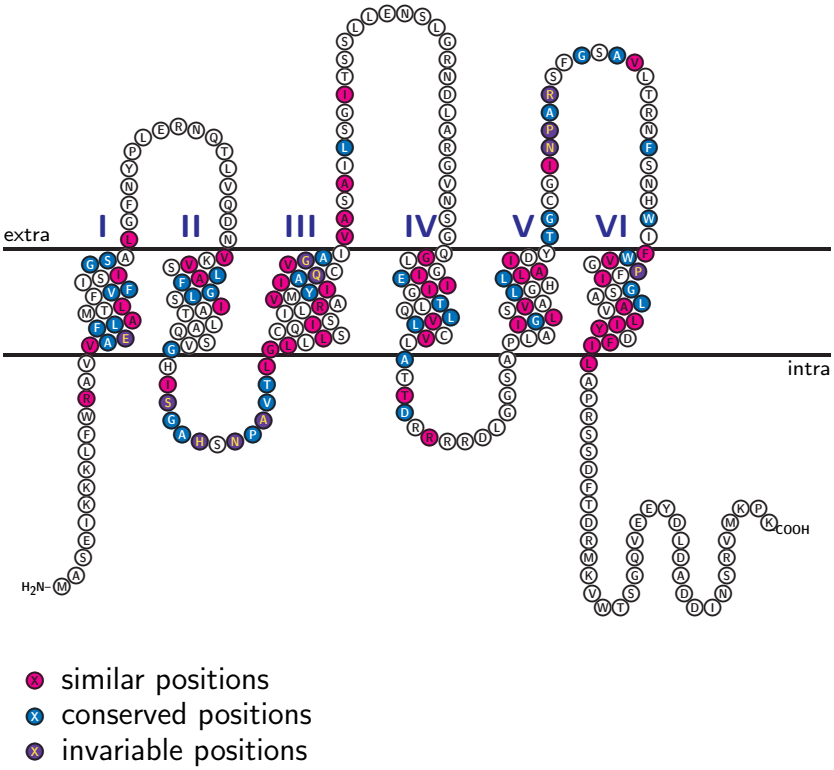


Figure 2: Topology plot with shading calculated on the basis of a protein alignment.

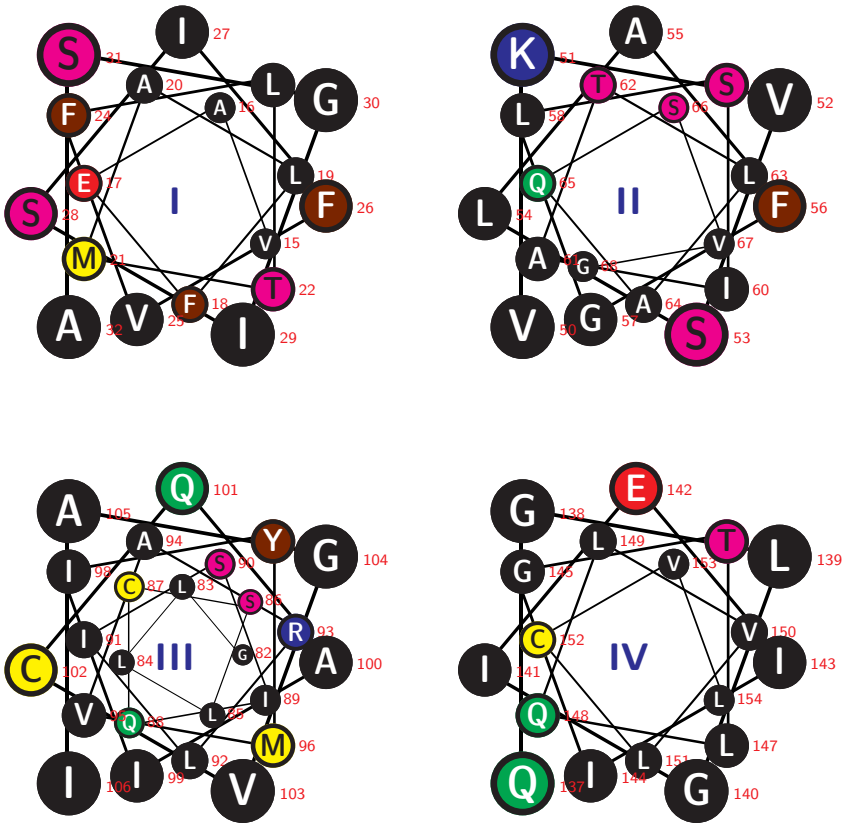


Figure 3: Some helical wheels with 'chemical' shading.

instead of circles. These additions will be automatically included in the legend. Labels can be attached to single residues or stretches. Secondary modifications, such as phosphorylation and glycosylation, may be shown as an encircled 'P' and a tree, respectively. The appearance of the membrane is adjustable.
Further, the display of the structure itself can be altered by setting values for the maximal extension of each loop, by defining so-called 'half-loops' which are invaginations of short lipophilic stretches into the membrane or by declaring membrane anchors, such as GPI-anchors or bound lipids. One can change the location of the N-terminus from intra- to extracellular and vice versa.
The description of the usage of all necessary commands is topic of the following sections.

2 Use of a TEXtopo parameter file

Using predefined parameter files for repeatedly occuring situations can save a lot of typing and makes the output throughout the publication or presentation more consistent. Further, such files are an easy way to exchange self-defined shading modes or new color schemes (i. e. for a satisfying grayscale output) with other users. If you have created a parameter file, which you think is of interest for others, please submit it to me[2] as an e-mail attachment together with a short description. I will take care of those files and post them—with a reference to the author—together with the next TEXtopo distribution to make them available for all interested users.
No special file format is required for parameter files. TEXtopo simply calls the file using the `\input` command right after resetting all parameters to default. An example parameter file is present containing the standard parameters of TEXtopo called `textopo.def`. This file can be changed freely and can be used as a template for the creation of personal parameter files.

3 TEXtopo user commands

The TEXtopo package must be loaded by the `\usepackage` command in the document header section.

`\usepackage[`⟨*option*⟩`]{textopo}`

[2] `eric.beitz@uni-tuebingen.de`

Then, the `textopo` and `helicalwheel` environments are ready to use as described in 1.4. See also section 2 for a description of the optional parameter file. All other commands provided by TEXtopo must be used within the `textopo`/`helicalwheel` environments. The following sections mainly focus on plotting topologies rather than helical wheels. For the latter a special section is reserved [3.4]. Nevertheless, almost all commands behave the same in both environments.
The TEXtopo command syntax mainly follows the LATEX conventions. Mandatory parameters are indicated by braces (`{}`), optional parameters are set in brackets (`[]`). Sometimes, optional parameters can be included in mandatory parameter definitions in order to save a lot of additional commands:

`\command[`⟨*general option*⟩`]{`⟨*mandatory*⟩`[`⟨*optional*⟩`]}`

This syntax is not used in standard LATEX commands. The following descriptions explain exactly in which commands this new kind of declaration can be used.

3.1 Sequence and topology data sources

As pointed out earlier, there are several sources of data which can be accessed by TEXtopo: (a) PHD topology predictions [3], (b) HMMTOP topology prdictions [8], (c) SwissProt database files, (d) alignment files in the MSF- (GCG PileUp) or ALN- (Clustal) format and (e) manually provided sequences. The latter two sources do not contain topological data, therefore the location of the transmembrane domains must be entered by hand using `\MRs` [3.1.4] and the location of the N-terminus must be set by `\Nterm` [3.1.4]. Let us go through all options:

3.1.1 PHD files

The sequence together with the topology is presented near the bottom of the file (see example `AQP1.phd`). TEXtopo analyzes the lines starting out with `AA` which contain the amino acid sequence and those beginning with `PHDThtm` to obtain the topology prediction. This information is converted into TEXtopo commands which are subsequently stored in a file named *filename*`.tpo`. This has the advantage that the entries are editable for further TEX runs. TEXtopo will not overwrite existing `tpo`-files in order to keep user made modifications of these

files, but it can be forced to overwrite them by using the optional parameter `[make new]`.

Syntax: `\getsequence[make new]{PHD}{`⟨*PHD-file*⟩`}`

3.1.2 HMMTOP files

HMMTOP predictions have various possibilities for the output format. Choose the extended format in TEXT-mode, because this contains the sequence in addition to the position of the termini and transmembrane domains (see example `AQP1.hmm`). This information is in analogy to PHD-files, s. a., converted into TEXtopo commands which are subsequently stored in a file named *filename*`.htp`.

Syntax: `\getsequence[make new]{HMMTOP}{`⟨*HMMTOP-file*⟩`}`

3.1.3 SwissProt files

These files provide next to the amino acid sequence (at the very bottom, `SQ`) much more information. Have a look at the example file `AQP1.SP`. The lines starting out with `FT` contain data about sequence features. Here, the positions of the transmembrane domains (`TRANSMEM`) are listed. All additional features will automatically be displayed in the topology plot as shaded sequence stretches or as labels. Unfortunately, the locations of the transmembrane domains are not always listed. In this case TEXtopo will complain about missing definitions of membrane regions and those have to be entered by hand, see 3.1.4. As in 3.1.1 a new file is written by TEXtopo with a name like this: *filename*`.swp` to enable easy customization.

Syntax: `\getsequence[make new]{SwissProt}{`⟨*SwissProt-file*⟩`}`

3.1.4 Alignment files

In order to extract a sequence from an alignment file the respective sequence number has to be stated based on the top sequence which is defined as no.1; if no number is indicated TEXtopo loads the first sequence. Two different alignment file formats are readable by TEXtopo, see the examples `AQPpro.MSF` and `AQP2spec.ALN`.

Syntax: `\getsequence[`⟨*seqnum*⟩`]{alignment}{`⟨*Alignment-file*⟩`}`

The positions of the membrane regions are declared by the command `\MRs{`⟨*start1..stop1,start2..stop2,...,start n..stop n*⟩`}`. If the

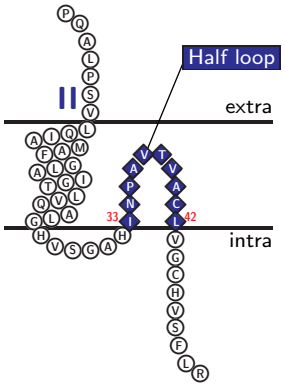


Figure 4: A 'half loop' example

*M*embrane *R*egion*s* are located for example from position 88 to 109 and from 123 to 150 enter `\MRs{88..109,123..150}`. Due to the thickness of the lipid bilayer an α-helical transmembrane spanning region is about 21 amino acids long. TEXtopo accepts definitions in the range of 14–36 amino acids. If the number of residues is below 14, which is definitively to short to span the membrane, a so-called 'half-loop' is assumed as shown in the topology clipping in Fig.4.

The orientation of the protein in the membrane is determined by the location of the N-terminus. This information is provided by PHD-, HMMTOP- and SwissProt files, when using alignment files in turn the command `\Nterm{`⟨*location*⟩`}` with *location* = `intra` or `extra` can help out. If the N-terminus is not set TEXtopo assumes the N-terminus to be intracellular.

3.1.5 Manual entry

Finally, the `\sequence` command allows one to enter the sequence manually directly in the `textopo` or `helicalwheel` environment.

Syntax: `\sequence{`⟨*Amino acid sequence*⟩`}`

This command provides a second possibility to define membrane domains in addition to `\MRs` overcoming the obstacle that one needs to know the position numbers in order to use `\MRs`, which may result in an annoying counting job. With `\sequence` the membrane regions can be indicated by brackets directly in the amino acid sequence:

```
\sequence{MASEIKKKLFWRAV[VAEFLAMTLFVFISIGSA]LGFNYPLERN
QTLVQDN[VKVSLAFGLSIATLAQSVG]HISGAHSNPAVTL[GLLLSCQISILR
AVMYIIAQCVGAI]VASAILSGITSSLLENSLGRNDLARGVNSG[QGLGIEIIG
TLQLVLCVL]ATTDRRRRDLGGSA[PLAIGLSVALGHLLAIDY]TGCGINPARS
FGSAVLTRNFSNHWI[FWVGPFIGSALAVLIYDFI]LAPRSSDFTDRMKVWTSG
QVEEYDLDADDINSRVMKPK}
```

Another feature of `\sequence` is its ability to print messages containing position information during the TEX run. Thus, if one needs to know the position number of a special residue, say a secondary modification site, this residue can be labeled with asterisks and the number will be displayed on the screen.

```
\sequence{MASEIKKKLFWRAV[VAEFLAMTLFVFISIGSA]LGFNYPLER*N*
QTLVQDN[VKVSLAFGLSIATLAQSVG]HISGAHSNPAVTL[GLLLSCQISILR
AVMYIIAQCVGAI]VASAILSGITSSLLENSLGRNDLARGVNSG[QGLGIEIIG
TLQLVLCVL]ATTDRRRRDLGGSA[PLAIGLSVALGHLLAIDY]TG*C*GINPARS
FGSAVLTRNFSNHWI[FWVGPFIGSALAVLIYDFI]LAPRSSDFTDRMKVWTSG
QVEEYDLDADDINSRVMKPK}
```

Screen output: `(pos 'N': 42) (pos 'C': 189)`

In addition, shading and labels can be set directly within the `\sequence` command; this will be described later [3.3.2]. Do not forget to define the N-terminus location by `\Nterm` [3.1.4] if it is extracellular.

3.2 Structure modifications

3.2.1 Output size

TEXtopo tries to select a font size for the residue symbols that makes the plot fit onto the page without receiving TEX error messages. Actually, it only checks the width, so the user has to take care of the proper height. If the size is not satisfactory one can change it using `\scaletopo{`⟨*fixed or relative size*⟩`}`. There are ten different sizes to choose from which are referred to by the numbers 1 (very small) to 10 (huge). Any fixed size can be set by indicating the respective number, e. g. `\scaletopo{5}`. Another possibility is to increase or decrease the size based on the calculation TEXtopo has made. Those relative settings are done by entering a number with a '`+`' or '`-`'. For exam-

ple, `\scaletopo{+2}` will increase the font size by two steps relative to the calculation. After increasing the font size `overful hbox` error messages will most likely appear.

3.2.2 Loop modifications

The height of the topology plot can be controlled by values that define the extent of each loop above or beneath the membrane. The command `\loopextent[`⟨*loop*⟩`]{`⟨*extent*⟩`[`⟨*distance*⟩`]}` takes three values which have the following effects:

⟨***extent***⟩ is the only mandatory value needed by `\loopextent`. It sets the maximal number of residues in the straight ascending or decending parts of the loop including the residues in the bend. Default setting is '30'.

⟨***distance***⟩ (optional) defines the minimum distance of the loops from the membrane if the loop is plotted in a meandrine shape. The default setting is '5'. Altering this setting might be necessary when flipping the termini to the interior of the protein, see below.

⟨***loop***⟩ (optional) restricts the settings to a particular loop number incl. N- and C-termini ('`N`', '`C`'). If this value is not set every loop is changed according to the ⟨*extent*⟩ and ⟨*distance*⟩ values.

Example A: `\loopextent[N]{50[10]}` sets the N-terminal loop to a maximal extent of 50 residues with a minimal distance of 10.

Example B: `\loopextent[3]{30}` sets the third loop to a 30 residue extent keeping the default for ⟨*distance*⟩.

Example C: `\loopextent{40}` sets a general maximum of 40 residues to all loops keeping the default minimal distance.

Setting the maximal and minimal distances from the membrane might not be sufficient for an optimal plot if the respective loop is very long. When it is necessary to switch to the meandrine style the distance between the loop flanking transmembrane domains gets bigger and bigger. This can be avoided if a foot with a fixed width in its '*neck*'-part is added to the loop. The command `\loopfoot{`⟨*loop*⟩`}{`⟨*direction*⟩`[`⟨*neck*⟩`]}` does exactly that. Note that ⟨*loop*⟩ is mandatory now with the termini excluded. The ⟨*direction*⟩ parameter can be `left`, `right` or `center`. This defines the direction of the foot extension. Finally, the optional ⟨*neck*⟩ value

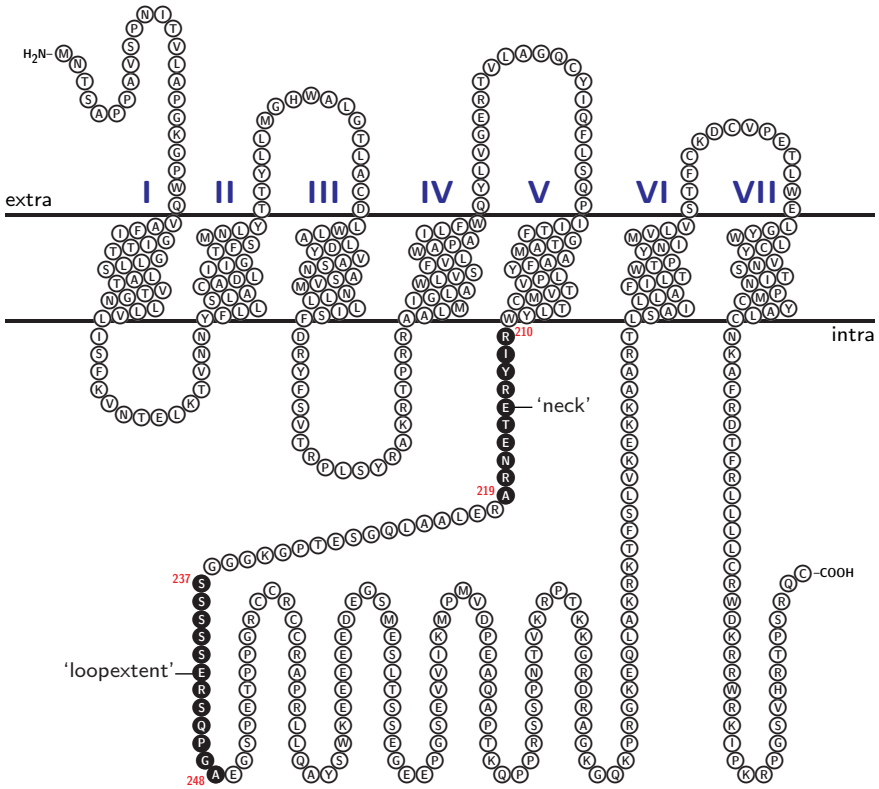


Figure 5: Example of the `loopfoot` command. Shown is the muscarinic acetylcholine receptor with its relatively big loop E. Settings are: `left` for 'direction' and 10 for 'neck' with a `loopextent` of 12.

sets the number of residues in the short straight part of the foot—I call it the *neck*—and thus the distance from the membrane to the start of the opening of the foot. Default setting here is '5'. The actual loop is plotted atop of the foot according to the `\loopextent` value. This means, that loops with a foot have a greater extent than loops without a foot. Thus, one might want to adjust the `\loopextent` setting for those loops. The optional parameter `[`⟨*distance*⟩`]` in the `\loopextent` command is ignored in `\loopfoot`. Figure 5 gives an example.

A further change in the output can be achieved by flipping the termini to the interior part of the protein. This leads to a more compact plot on the one hand but makes labeling more difficult due to

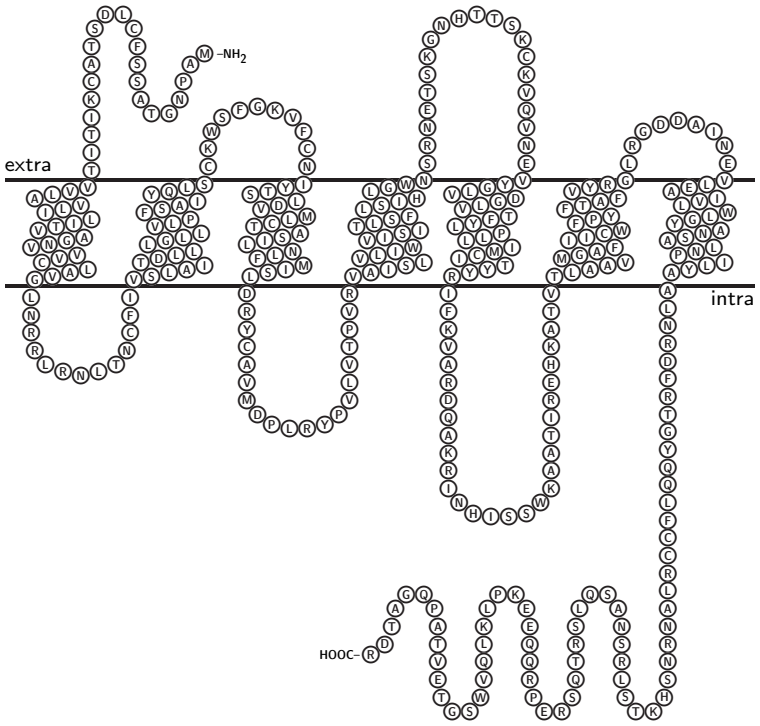


Figure 6: The human gastric histamine receptor (H_2). An example for flipped termini.

less space on the other hand. There are two commands available one for the N-terminus (`\flipNterm`) and one for the C-terminus (`\flipCterm`). This kind of structural change might result in collisions with other loops. In such a case one has to adjust the loop settings using `\loopextent` or `\loopfoot`, see above. Fig.6 shows the flipping effect.

3.2.3 Membrane domains

The `\MRs` command has already been extensively described [3.1.4]. But in some cases it might be helpful to be able to clear the membrane domain settings, e. g. if the definitions or preditions of a SwissProt- or PHD-file are not acceptable and new settings should be made. There-

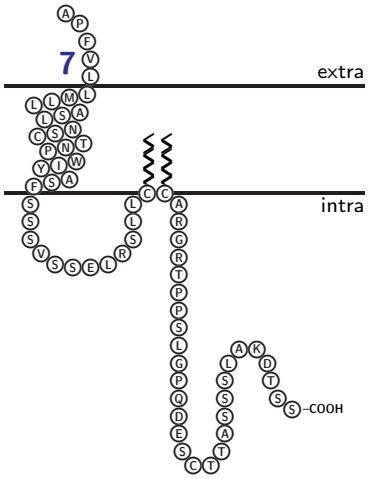


Figure 7: An example for lipid anchors. The V_2-receptor (only TM7 and C-terminus shown) contains two anchors which are directly vicinal.

fore, `\clearMRs` was implemented. This command is self-explanatory. It has an immediate clearing effect on the settings before the command.
Another structural feature should be discussed as a membrane domain topic, i. e. lipid membrane anchors (`\anchor{`⟨*pos*⟩`}`). This very easy to use command draws a symbolic lipid chain to the residue at position ⟨*pos*⟩ and attaches it to the membrane. See Fig.7 for an example.

3.2.4 Cosmetics on the membrane

As a default the membrane is shown as two horizontal lines representing the borders. If a more solid appearance is desired the command `\membranecolors{`⟨*border*⟩`}{`⟨*interior*⟩`}` can be employed. It accepts PostScript color names (see section4) for ⟨*border*⟩ and ⟨*interior*⟩, e. g. `\membranecolors{WildStrawberry}{Bittersweet}`. The thickness of the border lines can be set by `\borderthickness{`⟨*length*⟩`}`, e. g. `\borderthickness{4mm}`.
The short texts that indicate which side of the membrane is oriented to the extracellular and to the intracellular space can be changed by `\labeloutside[`⟨*pos*⟩`]{`⟨*text*⟩`}` and `\labelinside[`⟨*pos*⟩`]{`⟨*text*⟩`}`. The optional parameter ⟨*pos*⟩ tells TEXtopo whether to put the la-

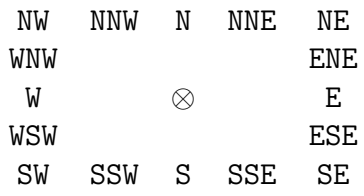


Figure 8: Directions for label movements from the center position.

bel `left` or `right` on the membrane.

Example: `\labeloutside[left]{blood}` `\labelinside{cytosol}`.

No indication of a preferential side leads to printing on the `left` for the outside label and on the `right` for the inside. For a fine adjustment of the label positions use

`\moveinsidelabel{`⟨*direction,distance*⟩ or ⟨*x,y*⟩`}` and
`\moveoutsidelabel{`⟨*direction,distance*⟩ or ⟨*x,y*⟩`}`.

The parameter allows one to move the label into ⟨*direction*⟩ (see Fig.8) for the amount of ⟨*distance*⟩ units; only integer values are accepted here. One unit equals to 1/5th of the diameter of the residue symbol. This scheme is also used for most of the other move-commands which are described later. An example would be: `\moveinsidelabel{WSW,10}`.
Since v1.2 intuitive *x/y*-values can be used to define the new position besides the method described above. An example would be: `\moveinsidelabel{10,-37}` which moves the label 10 units to the right and 37 units down.
The standard width of the membrane is one residue symbol broader than the extension of the N- and C-termini. If the termini are flipped to the inside, the calculation of the width is based on the transmembrane domains. In order the change the width manually use the command `\broadenmembrane{`⟨*left/right*⟩`}{`⟨*length*⟩`}`. The first parameter selects which end of the membrane is to be changed. The ⟨*length*⟩ is an integer value which tells TEXtopo by how much the width should be changed. One unit represents again 1/5th of the residue symbol. Negativ values are permitted to shorten the membrane, e. g. `\broadenmembrane{left}{-20}`. Analogous to broadening the membrane the thickness can be changed by `\thickenmembrane{`⟨*top/bottom*⟩`}{`⟨*length*⟩`}`.
`\hidemembrane` makes the membrane totally disappear, whereas `\showmembrane` brings it back again.

3.3 Putting labels on the plot

3.3.1 Labeling loops and membrane domains

By default transmembrane domains are labeled with upper case roman numerals. This is achieved by using the command `\labelTMs{`⟨*style*⟩`}` with ⟨*style*⟩ = `\Romancount` in the standard settings. All available ⟨*style*⟩ options are shown in the table below:

counter	*display*
`\numcount`	1, 2, 3 ...
`\alphacount`	a, b, c ...
`\Alphacount`	A, B, C ...
`\romancount`	i, ii, iii ...
`\Romancount`	I, II, III ...

Mind the backslash! This option is actually a command which is executed in the very moment the label is printed. One can also use combinations of text and a counter, e. g. `\labelTMs{TM\numcount}`.
In order to set a label for one particular transmembrane domain use `\labelTM[`⟨*direction,distance*⟩ or ⟨*x,y*⟩`]{`⟨*num*⟩`}{`⟨*label*⟩`}` (singular! no 's'). ⟨*num*⟩ indicates the number of the TM which is to be labeled with the text in ⟨*label*⟩. The optional parameter can be used as described before [3.2.4]. Here, *x/y*-values also work.
One can move individual transmembrane domain labels without having to take care of the label text by applying the command `\moveTMlabel{`⟨*num*⟩`}{`⟨*direction,distance*⟩ or ⟨*x,y*⟩`}`. The first parameter ⟨*num*⟩ refers to the domain number, the next pair of parameters corresponds to the ones described above. The color of the labels is set by `\TMlabelcolor{`⟨*color*⟩`}`. For a description of the color codes see section4. The font styles are also adjustable, see section3.5. One final command concerning transmembrane domain labels is the self-explanatory `\hideTMlabels`.
Labels for the extra- and intracellular loops are handled exactly in the same way as the transmembrane domain labels by the following set of commands:

`\labelloops{`⟨*style*⟩`}`
`\labelloop[`⟨*direction,distance*⟩ or ⟨*x,y*⟩`]{`⟨*num*⟩`}{`⟨*label*⟩`}`
`\movelooplabel{`⟨*num*⟩`}{`⟨*direction,distance*⟩ or ⟨*x,y*⟩`}`
`\looplabelcolor{`⟨*color*⟩`}`
`\hidelooplabels`

Two pairs of special commands show or hide the extensions (H_2N– and –COOH) at the N- and C-termini; these are `\showNterm`, `\hideNterm`, `\showCterm` and `\hideCterm`.

3.3.2 Shading and labeling sequence features

The first thing to do before a certain residue or a sequence domain can be labeled is to define an appropriate shading style for this sequence stretch. Use the command `\labelstyle{`⟨*name*⟩`}{`⟨*shape*⟩`}{`⟨*frame color*⟩`}{`⟨*background color*⟩`}{`⟨*char color*⟩`}{`⟨*legend text*⟩`}` to set all necessary informations which are needed to define the shading. The first parameter ⟨*name*⟩ is an 'identification' of this specific label style. This is needed to be able to refer to it. Then, the ⟨*shape*⟩ (`circ`, `box` or `diamond`) and the colors for the symbol's rim, its background and the character in the center are set. The available colors are described in section4. Finally, ⟨*legend text*⟩ contains the text which is displayed in the figure legend [3.3.6].

Example: `\labelstyle{BlueDiamond}{diamond}`
`{Black}{Blue}{Yellow}{Example}`

This new definition can be used from now on to shade and label one or several single residues or sequence regions. It is a good idea to store a collection of style definitions in a parameter file (section2) to have them at hand whenever needed in future projects. The next command attaches the label to the positions to be labeled: `\labelregion[`⟨*direction,distance*⟩ or ⟨*x,y*⟩`]{`⟨*list of regions*⟩`}{`⟨*style name*⟩`}{`⟨*label text*⟩`}`.
This command is more complex than it seems at first sight. The optional parameter `[`⟨*direction,distance*⟩`]` can be used to move the label to a new position. The usage is as in `\moveinsidelabel` [3.2.4]. The third parameter ⟨*style name*⟩ calls the style definitions, i. e. for the example above it would be `{BlueDiamond}`. The complexity lies is the second and especially the fourth parameter. The ⟨*list of regions*⟩ has a similar syntax as the list in the `\MRs` [3.1.4] command. But here, the definition of both, the start and the stop position of each region can be followed by an optional ⟨*direction*⟩ parameter, i. e.

`{`⟨*start1`[`*⟨*direction*⟩`]..stop1[`*⟨*direction*⟩`]`⟩,. . . ,
⟨*start n`[`*⟨*direction*⟩`]..stop n[`*⟨*direction*⟩`]`⟩`}`.

It happens sometimes—especially in the bends of loops—that the residue number is being printed over another residue. In such a case the ⟨*direction*⟩ parameter lets one choose a new direction in which

the number will be displayed. All direction definitions shown in Fig.8 are permitted. Note that here no setting of the distance is needed. If an asterisk is used as parameter the number will not be displayed at all. This might be useful when positions are being labeled where not enough space is available for the number, e. g. within the dense packing of a helical domain.
Now, for the actual label text. The easiest way is to use plain text as label. Then an example would simply be `{not fancy}`. If one wants to add colors this has to be declared by an optional parameter right after the text, e. g. `{not fancy but red[Red]}`. This text can further be boxed by extending the argument like this: `{box:not fancy but red[Red]}`. A white box with a black frame will be printed. Maybe colors would be nicer; an optional extension does the job: `{box[Blue,Yellow]:not fancy but red[Red]}`. This will produce a blue framed yellow box around the red text "not fancy but red" which is quite fancy now. If the box frame and background are supposed to have the same color it is enough to indicate this only once, e. g. `{box[yellow]: ...}`. In addition to framed boxes two more symbols are at hand: `{circ[col1,col2]: ...}` and `{diamond[col1,col2]: ...}`. There is only space for one letter in a circle or a diamond. If longer text is used it will be printed over the rims of the symbol which looks rather ugly. An appropriate application might be an encircled 'P' to indicate a phosphorylation site. A last symbol `{tree}` does not accept any text; it is meant to indicate glycosylation sites.
The following example uses the previously defined shading style 'BlueDiamond' for the residues and prints a red colored text in a blue framed yellow box to label the sequence stretch from position 20 to 30 and the single residue 76. Further, the labels are moved westwards by 10 units (= 2 residue diameters) and the first position number is hidden, the second is displayed beneath the residue, whereas the third is not altered.

```
\labelregion[W,10]{20[*]..30[S],76..76}{BlueDiamond}
                  {box[Blue,Yellow]:red plain text[Red]}
```

It should be mentioned that there are already two shading styles predefined in TEXtopo called `standard` and `noshade`. The former style uses the definitions of the standard residues for the labeled positions, the latter style can be used to attach a label to a certain residue or domain without influencing the residues. Such kinds of labels are handy when using calculated shading [3.3.5] that should not be depended on

any kind of labels.
It is also possible to make almost all these settings directly in the `\sequence` [3.1.5] definition similar to the declaration of the membraneous domains [3.1.4] without knowing the position numbers. However, the command structure will get rather complex and makes the readability of the sequence worse the more optional parameters are defined. The following example uses the settings as the example above.

```
\sequence{MLNLFMISLDRYCAVMDPL
([W,10]BlueDiamond[box[Blue,Yellow]:red plain text[Red]]=
YPVLVTPVRVA)ISLVLIWVISITLSFLSIHLGWNSRNETSKGNHTTSKCKVQVNEV
([W,10]BlueDiamond[box[Blue,Yellow]:red plain text[Red]]=
G)LVDGLVTFYLPLLIMCITYYRIFKVARDQAKRINHISSW ...}
```

As I said, it gets complex. One might figure out how to use this shading definition from the shown example. There is one restriction of this method: the printing direction of the position numbers relative to the residue can not be influenced. Nevertheless, this kind of labeling might be useful for brief plain labels. For more complex labels one better takes one extra step to figure out the exact position numbers by using '`*`'s in the `\sequence` command [3.1.5] in order to set the label afterwards with `\labelregion`.
Two often occuring modifications can be labeled using some kind of 'short-cut' commands, i. e. `\phosphorylation{`⟨*list of positions*⟩`}` and `\glycosylation{`⟨*list of positions*⟩`}`.

Example: `\phosphorylation{10,45,99} \glycosylation{123}`

Minor alterations concern the color of the residue number which can be changed by `\countercolor{`⟨*color*⟩`}` (again, for colors see section4), and the thickness of the line that connects the label with the residue (`\rulethickness{`⟨*thickness*⟩`}`. Two examples:

```
\countercolor{Blue} \rulethickness{2pt}
```

3.3.3 Placing additional labels

All kinds of labels discussed before are attached or related to some protein segments. The command `\place[`⟨*num*⟩`]{`⟨*x%*⟩`,`⟨*y%*⟩`}{`⟨*label*⟩`}` allows one to place any kind of text to any position in the figure. The optional parameter ⟨*num*⟩ is only necessary when plotting helical wheels [3.4]. The second parameter lets one set the position of the label. The system underlying the position calculation differs from that

described in the previous commands. Here, the x- and y-positions are expressed as percentages ($x\%$ and $y\%$) of the total width and height of the plot with the origin in the lower left corner. So, `{0,0}` places the label in this corner, `{0,100}` in the upper left, `{100,0}` in the lower right, `{100,100}` in the upper right corner and `{50,50}` in the center of the figure. Of course, any other setting is also permitted. ⟨*label*⟩ finally, holds the label text which can further contain any desired LaTeX style modification command.

Example: `\place{10,90}{\textbf{\Large Topology example}}`

3.3.4 Adding protein tags and changing the numbering

In some cases the protein which is plotted contains artificial tags for affinity purification, e. g. a oligo-His-tag, for antibody detection, e. g. a myc-tag, or for other purposes, e. g. a GFP-fusion etc. Using TeXtopo one can attach those tags to the termini of the protein without altering its numbering, i. e. the sequence tagged to the N-terminus will be numbered with negative digits so that the original start-methionine will still be number one. The syntax of the commands for adding tags to the N- and C-terminus is very similar to the `\labelregion` command [3.3.2]:

`\addtagtoNterm[`⟨*direction,distance*⟩ or ⟨*x,y*⟩`]{`⟨*tag sequence*⟩`}`
`{`⟨*style name*⟩`}{`⟨*label text*⟩`}`
`\addtagtoCterm[`⟨*direction,distance*⟩ or ⟨*x,y*⟩`]{`⟨*tag sequence*⟩`}`
`{`⟨*style name*⟩`}{`⟨*label text*⟩`}`

In contrast to `\labelregion` no definition of the positions is necessary because the location is clear anyway (N- or C-terminus). Instead of that the sequence of the tag needs to be entered as the second parameter. Everything else is identical to `\labelregion`.

Example: `\addtagtoNterm[N,8]{MEQKLISEEDAAA}{myc}{myc-tag}`

This attaches a myc-tag and a spacer of three alanines to the N-terminus, shades the tag with a style named `myc` and prints the label text `myc-tag` which is moved to the North by eight units. The following original start-methionine still is no.1, whereas the new start-methionine of the myc-tag is no.−13, then counting up to −1 for the last of the three alanines.
Another possibility to change the residue numbering is the command `\seqstart{`⟨*num*⟩`}`. This lets one set any number—except 0—as the sequence start. One application would be proteins with pro-peptides.

Here, the pro-peptide could be negatively numbered ending up with no.1 at the starting position of the mature protein.

3.3.5 Applying calculated shading

As already pointed out in section1.5, a special feature of TEXtopo is its ability to communicate with TEXshade and use the shading calculated by this comprehensive alignment shading programm in a topology plot to indicate residue conservation or functional aspects, see examples in Fig. 2 and 3.
In order to use TEXshade's calculated residue shading in topology plots the package must be loaded in the document header by `\usepackage[`⟨*option*⟩`]{texshade}`. This command must be given *before* loading the TEXtopo package! This is due to some re-definitions TEXtopo does on TEXshade commands. So, the document header must contain the following two commands:

```
\usepackage[⟨option⟩]{texshade}
\usepackage[⟨option⟩]{textopo}
```

A safer possibility is to load both biological packages by declaring `\usepackage[`⟨*option*⟩`]{biotex}` instead of the commands above. BIOTEX does not provide new command definitions; it only organizes the loading of the packages (so far TEXshade and TEXtopo are available—but the collection is going to be extended) in the correct order and checks for the appropriate version numbers. The BIOTEX style has been produced automatically when TEXtopo was extracted from the docstrip archive. Keep this file in a directory searched by TEX, e. g. together with your TEXtopo files.
At this point it is refered to the TEXshade manual for an extensive description of the different shading modes. Here, only a basic overview will be given.

Identity mode: This basic type of shading is provided by almost any alignment program. All identical residues at a position are shaded if the number of matching residues is higher than a given threshold percentage.

Similarity mode: Consider an alignment position where three out of five residues are basic arginines and two more residues are also basic lysines. In similarity mode TEXshade shades similar residues in a different color to distinguish them from the consensus residue. Even when none of the residues alone reaches the

threshold but a group of similar residues does these are shaded in the 'similarity' color. This case is given for instance when at a position in a five sequence alignment two aliphatic valines and two also aliphatic isoleucins are present and the threshold is set to 50%. Neither residue exceeds this percentage but as a group of similars they do.

Functionality modes: Displaying functional peptide similarities is one of TEXshade's strong capabilities. Six functional shading modes are predefined, see references [4–7]; further user specific modes can easily be created.

- `charge`: residues which are charged at physiological pH (7.4) are shaded if their number at a position is higher than the threshold
- `hydropathy`: discrimination between acidic and basic, polar uncharged and hydrophobic nonpolar residues
- `structure`: displays the potential localization within the tertiary structure of the protein
- `chemical`: residues are shaded due to chemical properties of their functional groups
- `standard area`: this shading displays the surface area sizes of the different amino acid's sidechains
- `accessible area`: here, the surface area which can be accessed by solvent molecules is used as a basis for shading; low accessibility means hydrophobic (i. e. strongly buried residues), whereas highly accessible sidechains are hydrophilic (compare to `hydropathy` and `structure`)

For the first two modes (`identical`, `similar`) an alignment file containing the sequence to be plotted is needed on which the shading calculation will be based. The syntax is as follows:

`\applyshading[`⟨*num*⟩`]{`⟨*mode*⟩`}{`⟨*filename*⟩`}`

The optional ⟨*num*⟩ tells TEXtopo which sequence number within the alignment corresponds to the sequence to be plotted. If no ⟨*num*⟩ is indicated the top sequence will be taken (no.1). The second parameter ⟨*mode*⟩ selects the shading mode. Options are `identical` and `similar`. Finally, the file name of the alignment is the argument of

the third parameter. For the alignment file formats see the TEXshade manual and the example files `AQPpro.MSF` and `AQP2spec.ALN` or 3.1.4. All TEXshade commands are applicable to obtain the desired shading, e. g. `\threshold`, `\shadingcolors` or `\allmatchspecial`. The function of these commands is to set the threshold percentage for the consensus, e. g. `\threshold{50}`, select another colorscheme (see section5), e. g. `\shadingcolors{greens}`, or use a special color for positions where all residues match. The counterpart of `\allmatchspecial` is `\allmatchspecialoff`.

```
\applyshading[3]{similar}{file.MSF} \allmatchspecial
```

This example calculates shading for the third sequence of the alignment file `file.MSF` and shades all similar and conserved positions plus the positions with 100% identity in different colors. See also the example on page 10.
The colors for each matching quality are changeable by the next four commands:

`\standardresidues{`⟨*style*⟩`}{`⟨*frame*⟩`}{`⟨*background*⟩`}{`⟨*char*⟩`}`
`\similarpositions{`⟨*style*⟩`}{`⟨*frame*⟩`}{`⟨*background*⟩`}{`⟨*char*⟩`}`
`\conservedpositions{`⟨*style*⟩`}{`⟨*frame*⟩`}{`⟨*background*⟩`}{`⟨*char*⟩`}`
`\invariablepositions{`⟨*style*⟩`}{`⟨*frame*⟩`}{`⟨*background*⟩`}{`⟨*char*⟩`}`

The ⟨*style*⟩ options are: `circ`, `box`, and `diamond`. The next three parameters are color definitions for the symbol ⟨*frame*⟩, its ⟨*background*⟩, and the residue ⟨*char*⟩.

Example: `\conservedpositions{circ}{Black}{Blue}{White}`

Mostly, in alignments the starting methionine is shaded with the color for highly conserved residues, simply because every protein starts out with a methionine, except for the cases where some kind of maturation takes place. Anyway, this methionine has usually no particular function in the protein. Hence, the shading for it is deactivated by default in TEXtopo by the command `\donotshadestartMet` in the standard settings. To re-active the shading use `\shadestartMet`.
The third group of functional shading modes is calculated on the topology sequence itself. Hence, no alignment file must be loaded which changes the `\applyshading` command somewhat:

`\applyshading[`⟨*parameterfile*⟩`]{`⟨*mode*⟩`}{`⟨*group*⟩`}`

Here, ⟨*mode*⟩ is `functional` and ⟨*group*⟩ is one of the amino acid groupings described above (`charge`, `hydropathy`, `structure`, `chemical`, `sidechain area`, and `accessible area`). The optional

parameter loads a TEXshade parameter file. This allows one to apply shading modes from a user-defined collection without any hassles.

Example: `\applyshading{functional}{charge}`

This shades all charged residues in the topology plot according to the color definitions of the functional shading mode `charge`, see section5. Again, it is refered to the TEXshade manual for an extensive exploitation of the given possibilities.

3.3.6 The figure legend

Legends are automatically printed when necessary, i. e. when labels are used [3.3.2] or calculated shading is applied [3.3.5]. Then, an example residue is shown and the explanation of the shading behind it, as defined in `\labelstyle` or in the shading mode. The output of the legend can be suppressed by `\hidelegend`, or if needed turned on again by `\showlegend`. If the position of the legend below the figure is not satisfying one can move it using the command `\movelegend{`⟨*x-offset*⟩`}{`⟨*y-offset*⟩`}`. The two parameters are TEX lengths, e. g. `\movelegend{5cm}{-3cm}` moves the legend 5 cm to the right and 3 cm up.

3.4 Plotting helical wheels

Almost everything that has been said before is also true for using the `helical wheel` environment [1.4.2]. One can load and label the sequence exactly as described before. There is one minor restriction to the labeling: in helical wheels no label text will be displayed. But the `\place` [3.3.3] command is fully functional. Here, the optional parameter refers to the wheel, i. e. the number of the transmembrane domain, in which the label is to be placed. The ⟨*x%*⟩ and ⟨*y%*⟩ values are now according to the dimensions of one wheel rather than the whole figure.
The style of the helix (perspective, flat wheel or net) can be set by the command `\helixstyle{`⟨*style*⟩`}` with `perspective`, `wheel` or `net` as ⟨*style*⟩. The different outputs are shown in Figure 9.
In the `net` display the optional parameter which can follow the wheel number in the `\begin{helicalwheel}` call (see 1.4.2) does not represent an angle but defines where to start the net plot. Numbers from 1–7 are allowed here. This can be handy if a certain viewing angle on the helix should be displayed in order to show e.g. a hydrophobic

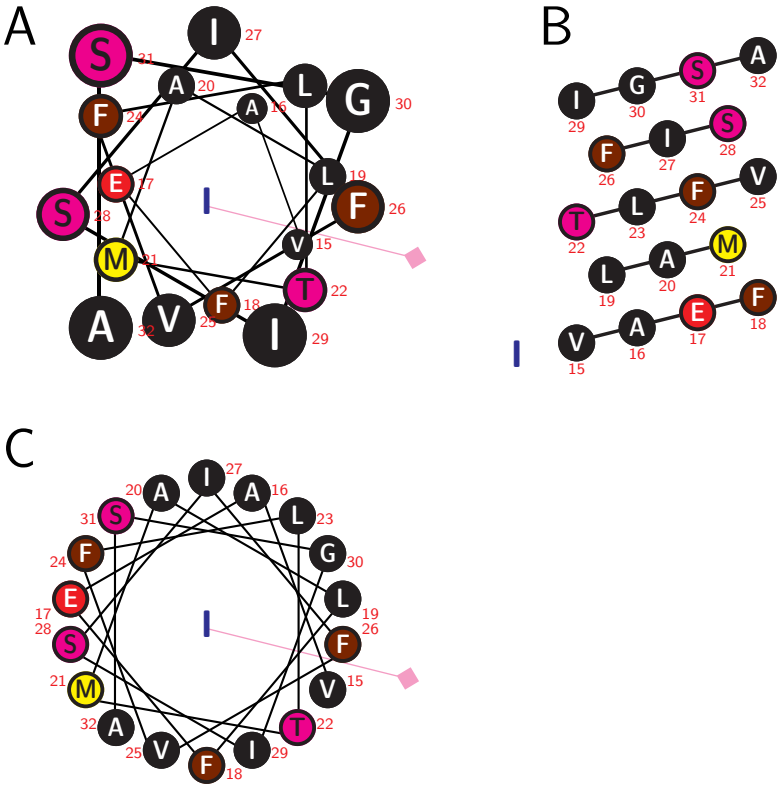


Figure 9: The different helical wheel styles `perspective` (A), `net` (B) and `wheel` (C) with indication of the hydrophobic moment.

core. Figure 10 examplary shows all possible starting points of the same helical sequence.
The size of the wheels can be changed by the `\scalewheel{`⟨*scale%*⟩`}` command. The wheel's original diameter is multiplied by the given percentage to increase (⟨*scale%*⟩ > 100) or decrease (⟨*scale%*⟩ < 100) the output.

Example: `\scalewheel{50}` reduces the diameter to 50%

The size of the residue symbols can then be adjusted by the command `\symbolsize{`⟨*size*⟩`}`. Three ⟨*size*⟩s are applicable: `small`, `medium` and `large`.
Depending on the diameter of the wheel TEXtopo calculates the highest possible number of wheels which can be printed in one row without colliding with the text width settings. If the number of wheels per line should be altered the command `\wheelsperline{`⟨*num*⟩`}` will help. Changing the calculated amount of wheels per line will most probably result in TEX run-time warnings due to `overful hbox`es.
One can choose the viewing direction on the helical wheels by `\viewfromextra` and `\viewfromintra`. The first setting will show the helices as if viewed from the extracellular space onto the cell membrane (recognizable by the residue numbering) and vice versa for the second case.
In the perspective and the wheel display the hydrophobic moment according to David Eisenberg of the helix can be shown as a line with a square at the end. The direction of the line depicts the angle and the area of the square the moment. To turn this on use `\showmoment` and `\hidemoment` turns it off again. Different values obtained from the calculation can be printed for every helix by the following commands with ⟨*helixnum*⟩ refering to the helix in question. These commands can only be used outside the `helicalwheel` environment or in the caption.

`\Hmean{`⟨*helixnum*⟩`}`	hydrophobicity per residue
`\muH{`⟨*helixnum*⟩`}`	(μH) hydrophobic moment
`\muHmean{`⟨*helixnum*⟩`}`	hydrophobic moment per residue
`\mudelta{`⟨*helixnum*⟩`}`	angle of the hydrophobic moment

The color of the moment indicator can be changed by the command `\momentcolor{`⟨*color*⟩`}`. The length of the line can be scaled by `\scalemomemt{`⟨*scale%*⟩`}` just like in the `\scalewheel` command mentioned before.

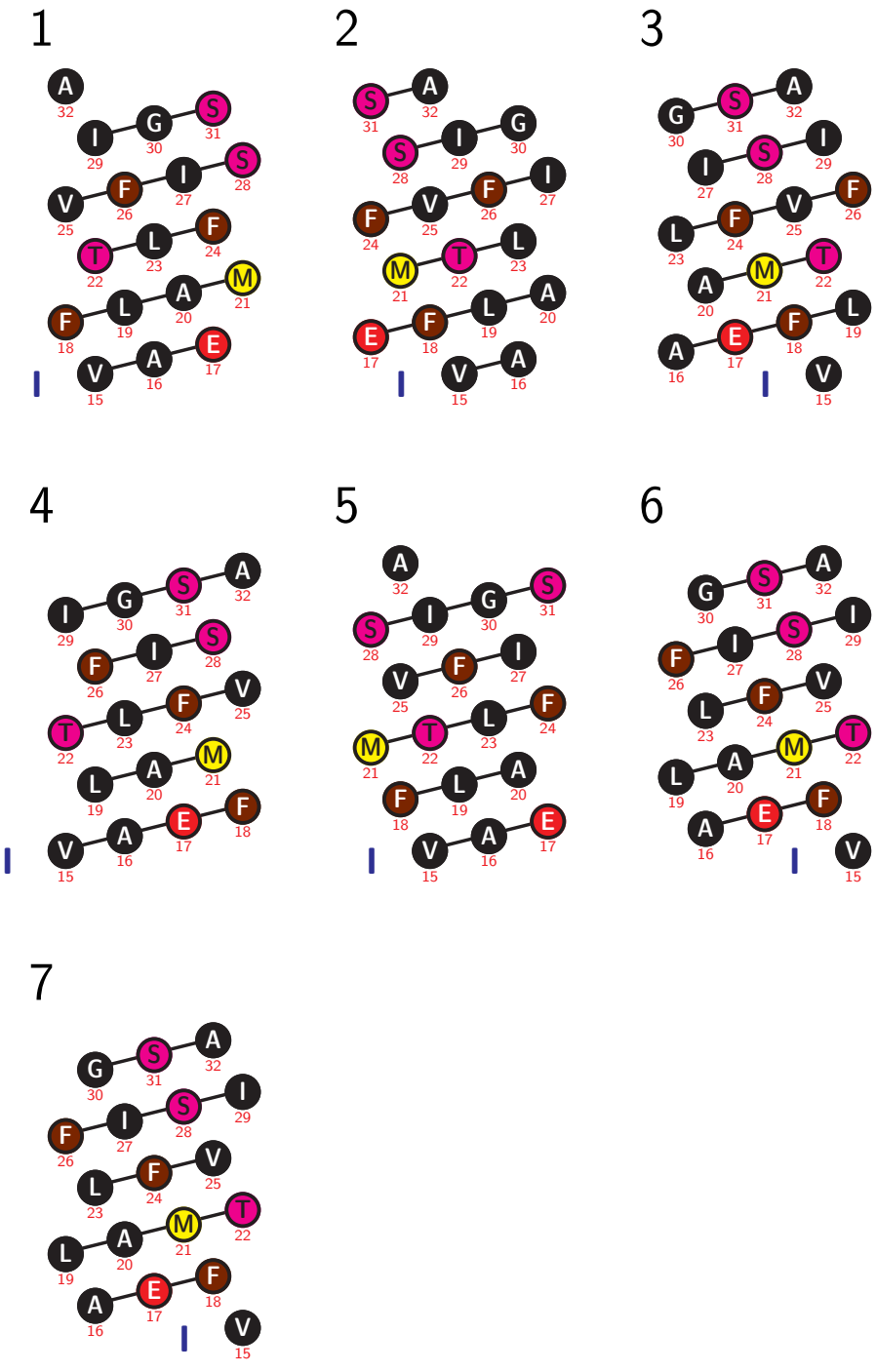


Figure 10: Net starting points 1–7 of the same sequence. In the first net the hydrophilic residues are at the sides, whereas net #3 shows them in the center.

Finally, two command pairs show or hide the residue numbering and the bonds between the residue symbols:

```
\showwheelnumbering
\hidewheelnumbering
\showbonds
\hidebonds
```

3.5 Changing font styles

The font styles for the legends, and the loop-, TM- and residue labels can be changed by several commands.

\setfamily{⟨*text*⟩}{⟨*family*⟩}
\setseries{⟨*text*⟩}{⟨*series*⟩}
\setshape{⟨*text*⟩}{⟨*shape*⟩}
\setsize{⟨*text*⟩}{⟨*size*⟩}

The first parameter selects the text whose style is to be changed. Possible first parameters are `legend`, `labels`, `looplabels` and `TMlabels`.

The style is set by the second parameter:

command	⟨*2. parameter*⟩	
\setfamily	rm	modern roman font family
	sf	sans serif font family
	tt	typewriter font family
\setseries	bf	bold face series
	md	normal series
\setshape	it	italics shape
	sl	slanted shape
	sc	small capitals shape
	up	upright shape
\setsize	tiny	the known TEX sizes
	scriptsize	
	footnotesize	
	small	
	normalsize	
	large	
	Large	
	LARGE	
	huge	
	Huge	

Example: `\setfamily{labels}{it} \setseries{labels}{bf}`

With the command

`\setfont{`⟨*text*⟩`}{`⟨*family*⟩`}{`⟨*series*⟩`}{`⟨*shape*⟩`}{`⟨*size*⟩`}`

all four font attributes of one ⟨*text*⟩ can be changed simultaneously. The order of the parameters is as indicated.

Example: `\setfont{labels}{rm}{it}{bf}{normalsize}`

Further, short commands are provided to change single font attributes quickly. The following commands show examplary how to change the settings for labels.

```
\labelsrm   \labelstiny
\labelssf   \labelsscriptsize
\labelstt   \labelsfootnotesize
\labelsbf   \labelssmall
\labelsmd   \labelsnormalsize
\labelsit   \labelslarge
\labelssl   \labelsLarge
\labelssc   \labelsLARGE
\labelsup   \labelshuge
            \labelsHuge
```

Corresponding sets are provided for loop labels (`\looplabelsrm` etc.), TM labels (`\TMlabelsrm` etc.) and legend texts (`legendrm` etc.).

4 The DVIPS color selection scheme

PostScript provides 64 standard colors. All these colors are predefined in the `dvips` package. Each color has a pictorial name such as `Bittersweet` and a distinct composition, e. g. 0% cyan + 75% magenta + 100% yellow + 24% black—the so-called CMYK scheme. TEXtopo enhances this color scheme by gray scales in 5% steps. The following colors and grays can be used in TEXtopo by simply declaring the name of the color in the respective command, e. g. `\membranecolors`:

name	*CMYK*	*name*	*CMYK*
GreenYellow	0.15,0,0.69,0	Yellow	0,0,1,0
Goldenrod	0,0.10,0.84,0	Dandelion	0,0.29,0.84,0
Apricot	0,0.32,0.52,0	Peach	0,0.50,0.70,0
Melon	0,0.46,0.50,0	YellowOrange	0,0.42,1,0
Orange	0,0.61,0.87,0	BurntOrange	0,0.51,1,0

Bittersweet	0,0.75,1,0.24	RedOrange	0,0.77,0.87,0
Mahagony	0,0.85,0.87,0.35	Maroon	0,0.87,0.68,0.32
BrickRed	0,0.89,0.94,0.28	Red	0,1,1,0
OrangeRed	0,1,0.50,0	RubineRed	0,1,0.13,0
WildStrawberry	0,0.96,0.39,0	Salmon	0,0.53,0.38,0
CarnationPink	0,0.63,0,0	Magenta	0,1,0,0
VioletRed	0,0.81,0,0	Rhodamine	0,0.82,0,0
Mulberry	0.34,0.90,0,0.02	RedViolet	0.07,0.90,0,0.34
Fuchsia	0.47,0.91,0,0.08	Lavender	0,0.48,0,0
Thistle	0.12,0.59,0,0	Orchid	0.32,0.64,0,0
DarkOrchid	0.40,0.80,0.20,0	Purple	0.45,0.86,0,0
Plum	0.50,1,0,0	Violet	0.79,0.88,0,0
RoyalPurple	0.75,0.90,0,0	BlueViolet	0.86,0.91,0,0.04
Periwinkle	0.57,0.55,0,0	CadetBlue	0.62,0.57,0.23,0
CornflowerBlue	0.65,0.13,0,0	MidnightBlue	0.98,0.13,0,0.43
NavyBlue	0.94,0.54,0,0	RoyalBlue	1,0.50,0,0
Blue	1,1,0,0	Cerulean	0.94,0.11,0,0
Cyan	1,0,0,0	ProcessBlue	0.96,0,0,0
SkyBlue	0.62,0,0.12,0	Turquoise	0.85,0,0.20,0
TealBlue	0.86,0,0.34,0.02	Aquamarine	0.82,0,0.30,0
BlueGreen	0.85,0,0.33,0	Emerald	1,0,0.50,0
JungleGreen	0.99,0,0.52,0	SeaGreen	0.69,0,0.50,0
Green	1,0,1,0	ForestGreen	0.91,0,0.88,0.12
PineGreen	0.92,0,0.59,0.25	LimeGreen	0.50,0,1,0
YellowGreen	0.44,0,0.74,0	SpringGreen	0.26,0,0.76,0
OliveGreen	0.64,0,0.95,0.40	RawSienna	0,0.72,1,0.45
Sepia	0,0.83,1,0.70	Brown	0,0.81,1,0.60
Tan	0.14,0.42,0.56,0		
White (Gray0)	0,0,0,0	Black (Gray100)	0,0,0,1
Gray5	0,0,0,0.05	Gray10	0,0,0,0.10
Gray15	0,0,0,0.15	Gray20	0,0,0,0.20
Gray25	0,0,0,0.25	Gray30	0,0,0,0.30
LightGray	0,0,0,0.33	Gray35	0,0,0,0.35
Gray40	0,0,0,0.40	Gray45	0,0,0,0.45
Gray50	0,0,0,0.50	Gray	0,0,0,0.50
Gray55	0,0,0,0.55	Gray60	0,0,0,0.60
Gray65	0,0,0,0.65	DarkGray	0,0,0,0.66
Gray70	0,0,0,0.70	Gray75	0,0,0,0.75
Gray80	0,0,0,0.80	Gray85	0,0,0,0.85
Gray90	0,0,0,0.90	Gray95	0,0,0,0.95

Type the color names with the upper case letters exactly as described above. For the definition of new colors use the `dvips` command in the document header section:

`\DefineNamedColor{named}{`⟨*name*⟩`}{cmyk}{`⟨*C,M,Y,K*⟩`}`

The ⟨*name*⟩ can be chosen freely, the values for the color composition must be in the range 0–1, i e. 0–100% of the respective component ('C' – cyan, 'M' – magenta, 'Y' – yellow, 'K' – black) separated by commas.

Example:

```
\DefineNamedColor{named}{Salmon}{cmyk}{0,0.53,0.38,0}
```

5 Colors used in the different shading modes

Color scheme *blues*:

res.color	*shad.color*	*residues*
Black	White	no match
Black	Magenta	similar
White	RoyalBlue	identical
Goldenrod	RoyalPurple	all match

Color scheme *greens*:

res.color	*shad.color*	*residues*
Black	White	no match
Black	GreenYellow	similar
White	PineGreen	identical
YellowOrange	OliveGreen	all match

Color scheme *reds*:

res.color	*shad.color*	*residues*
Black	White	no match
Black	YellowOrange	similar
White	BrickRed	identical
YellowGreen	Mahagony	all match

Color scheme *grays*:

res.color	*shad.color*	*residues*
Black	White	no match
Black	LightGray	similar
White	DarkGray	identical
White	Black	all match

Color scheme *black*:

res.color	*shad.color*	*residues*
Black	White	no match
Black	White	similar
White	Black	identical
White	Black	all match

Functional mode *charge*:

res.color	*shad.color*	*residues*
Black	White	no match
White	Red	acidic
White	Blue	basic

Functional mode *hydropathy*:

res.color	*shad.color*	*residues*
Black	White	no match
White	Red	acidic
White	Blue	basic
Black	Yellow	polar uncharged
White	Green	hydrophobic nonpolar

Functional mode *chemical*:

res.color	*shad.color*	*residues*
Black	White	no match
White	Red	acidic
White	Black	aliphatic
White	Green	amide
White	Brown	aromatic
White	Blue	basic
Black	Magenta	hydroxyl
Black	Orange	imino
Black	Yellow	sulfur

Functional mode *structure*:

res.color	*shad.color*	*residues*
Black	White	no match
Black	Orange	external
Black	Yellow	ambivalent
White	Green	internal

Functional mode *standard area*:

res.color	*shad.color*	*residues*
Black	White	no match
Black	BrickRed	G
Black	Orange	A, S
Black	Yellow	C, P
Black	YellowGreen	T, D, V, N
White	PineGreen	I, E
Black	SkyBlue	L, Q, H, M
White	RoyalPurple	F, K
White	RedViolet	Y
White	Black	R, W

Functional mode *accessible area*:

res.color	*shad.color*	*residues*
Black	White	no match
Black	BrickRed	C
Black	Orange	I, V, G
Black	Yellow	F, L, M, A
Black	YellowGreen	W, S, T, H
White	PineGreen	P
Black	SkyBlue	Y, D, N
White	RoyalPurple	E, Q
White	RedViolet	R
White	Black	K

6 Quick Reference

The logos

`\TeXtopo` `\TeXshade` `\BioTeX`

The TEXtopo environments (7 ff.)

`\begin{textopo}[`⟨*optional parameterfile*⟩`]`
further TEXtopo *commands*
`\end{textopo}`

`\begin{helicalwheel}[`⟨*parameterfile*⟩`]{`⟨*helixlist*⟩`}`
further TEXtopo *commands*
`\end{helicalwheel}`

Sequence and topology sources

Structure modifications

Output size

Loop modifications

Membrane domains

Plotting helical wheels

\viewfromintra [33]

\showmoment [33]
\hidemoment [33]
\Hmean{⟨*helixnum*⟩} [33]
\muH{⟨*helixnum*⟩} [33]
\muHmean{⟨*helixnum*⟩} [33]
\mudelta{⟨*helixnum*⟩} [33]
\momentcolor{⟨*color*⟩} [33]
\scalemoment{⟨*scale%*⟩} [33]

\showwheelnumbering [35]
\hidewheelnumbering [35]
\showbonds [35]
\hidebonds [35]

Changing font styles

\setfamily{⟨*text*⟩}{⟨*family*⟩} [35]
\setseries{⟨*text*⟩}{⟨*series*⟩} [35]
\setshape{⟨*text*⟩}{⟨*shape*⟩} [35]
\setsize{⟨*text*⟩}{⟨*size*⟩} [35]
\setfont{⟨*text*⟩}{⟨*family*⟩}{⟨*series*⟩}{⟨*shape*⟩}{⟨*size*⟩} [36]

\labelsrm	\labelstiny	[36]
\labelssf	\labelsscriptsize	
\labelstt	\labelsfootnotesize	
\labelsbf	\labelssmall	
\labelsmd	\labelsnormalsize	
\labelsit	\labelslarge	
\labelssl	\labelsLarge	
\labelssc	\labelsLARGE	
\labelsup	\labelshuge	
	\labelsHuge	

Corresponding sets are provided for loop labels (\looplabelsrm etc.), TM labels (\TMlabelsrm etc.) and legend texts (legendrm etc.).

References

[1] Carlisle, D. The Standard LaTeX 'Graphics Bundle', `color.sty`.
[2] Beitz, E. (2000) TeXshade: shading and labeling multiple sequence alignments using LaTeX 2_ε. *Bioinformatics*: **16**, 135–139.
[3] Rost, B.; Sander, C. (1994) Combining evolutionary information and neural networks to predict protein secondary structure. *Proteins: structure, function and genetics*: **19**, 55–72.
[4] Karlin, S.; Ghandour, G. (1985) Multiple-alphabet amino acid sequence comparisons of the immunoglobulin κ-chain constant domain. *Proc. Natl. Acad. Sci. USA*: **82**, 8597–8601.
[5] Kyte, J.; Doolittle, R. F. (1982) A simple method for displaying the hydropathic character of a protein. *J. Mol. Biol.*: **157**, 105–132.
[6] Rose, G. D.; Geselowitz, A. R.; Lesser, G. J.; Lee, R. H.; Zehfus, M. H. (1985) Hydrophobicity of amino acid residues in globular proteins. *Science*: **229**, 835–838.
[7] Lesser, G. J.; Rose, G. D. (1990) Hydrophobicity of amino acid subgroups in proteins. *Proteins: structure, function and genetics*: **8**, 6–13.
[8] Tusnady, G.E.; Simon, I. (2001) The HMMTOP transmembrane topology prediction server. *Bioinformatics*: **17**, 849-850.