

The **TEXshade** package*

Typesetting
nucleotide and peptide alignments

Eric Beitz[†]

v1.3; 2000/03/03

Abstract

Setting alignments of nucleotides and peptides for publication or presentation purposes is usually a time consuming two-step process. First, a scientific software is used for the calculation of the alignment. This is done in a few minutes. Then, in order to highlight special sequence relationships and to label positions and regions of interest a second software with high quality output capability is needed. Manipulating sequence alignments with standard word processing or graphics programs takes its time—often several hours—and simple layout changes such as re-breaking lines, say from 50 to 40 residues per line, elongate the working time considerably.

TEXshade is an alignment shading software completely written in **TEX/L^ATEX** which can process multiple sequence alignments in the **.MSF** and the **.ALN** file format. It provides in addition to common shading algorithms special shading modes featuring functional aspects, e. g. charge or hydrophathy, and a plenitude of commands for handling shading colors, text styles, labels, legends and even allows the user to define completely new shading modes. **TEXshade** combines highest flexibility and the habitual **TEX** output quality—with reasonable time expenditure.

*Please cite: Eric Beitz (2000), **TEXshade**: shading and labeling multiple sequence alignments using **L^ATEX 2_ε**. *Bioinformatics*: **16**, 135–139.

[†]University of Tübingen, Pharmaceutical Chemistry, Morgenstelle 8, D-72076 Tübingen, Germany; send electronic mail to eric.beitz@uni-tuebingen.de; for further information, updates and on-line documentation see my homepage at <http://homepages.uni-tuebingen.de/beitz/>

Contents

1	Package Overview	4
1.1	Version History	4
1.2	System requirements	6
1.3	The <code>texshade</code> environment	6
1.4	Shading modes predefined in this package	7
1.4.1	Identity mode	7
1.4.2	Similarity mode	7
1.4.3	Diversity mode	8
1.4.4	Functionality modes	8
1.4.5	Secondary structures	11
1.4.6	Sequence fingerprints	14
1.5	Customization of alignment outputs	15
2	Format of alignment input files	16
2.1	The <code>.MSF</code> file format	16
2.2	The <code>.ALN</code> file format	19
3	Use of a <code>TeXshade</code> parameter file	20
4	<code>texshade</code> user commands	21
4.1	Using predefined shading modes	21
4.2	Creating new functional shading modes	26
4.3	Appearance of the consensus line	27
4.4	Appearance of the sequence lines	29
4.4.1	Names, numbers and gaps	29
4.4.2	Hiding, killing, separating and ordering	31
4.4.3	Residues per line and further settings	32
4.4.4	Fingerprinting	33
4.5	Individual shading and labeling of sequence stretches	33
4.5.1	Manual shading of regions and blocks	34
4.5.2	Emphasizing regions and blocks	34
4.5.3	Graphical labeling of sequence features	35
4.5.4	Including secondary structure information	38
4.6	Displaying and building legends	40
4.7	Font handling	41
4.7.1	Changing font styles	41
4.7.2	Using PostScript fonts	43
4.8	Goodies— <code>molweight</code> and <code>charge</code>	43

5	The DVIPS color selection scheme	44
6	Listing of the texshade default settings	45
6.1	Standard definitions	45
6.2	Colors used in the different shading modes	46
7	Quick Reference	50
8	References	55

1 Package Overview

After `texshade.ins` is run through \TeX the following files should appear in the directory:

<code>texshade.sty</code>	the style file with all \TeXshade commands
<code>texshade.def</code>	an example parameter file with the standard parameter settings
<code>AQPDNA.MSF</code>	an example nucleotide alignment (<code>.MSF</code> -format)
<code>AQPpro.MSF</code>	an example protein alignment (<code>.MSF</code> -format)
<code>AQP2spec.ALN</code>	a further protein alignment (minimal <code>.ALN</code> -file)
<code>AQP1.phd</code>	secondary structure information (PHD-format)
<code>AQP1.top</code>	topology data extracted from <code>AQP1.phd</code>
<code>standard.cod</code>	standard genetic code definitions
<code>ciliate.cod</code>	ciliate macronuclear genetic code

The alignment file examples as well as the topology data file are needed for \TeX ing 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 \TeXshade package. All commands are described in detail later on.

1.1 Version History

v1.3 2000/3/3

Corrections: Line scrambling occurred when features were set in the `ttop` row without a feature in the `top` row. Fixed. The incompatible command `\language` with the `babel` package has been replaced by `\germanlanguage` and `\englishlanguage`¹.

Introductions: (a) Now, translations of sequence stretches are possible. Either nucleotide or amino acid sources can be translated. This is done by the new `{translate}` option for the feature command. (b) The codons are defined by the new command `'codon'`. Complete codon sets can be loaded by `'geneticcode'`. (c) Further, the size and style of the nucleotide triplets of backtranslations can be set by `'backtranslabel'` and `'backtranstext'`. (d) Two more feature counter styles were introduced: `'Romancount'` and `'romancount'`. (e) \TeXshade is now compatible with \TeXtopo , a new \TeX software for drawing and shading topology plots of membrane proteins.

¹Thanks to Eckhart Guthöhrlein.

v1.2a 1999/6/24 (not released)

Minor corrections: ‘`namecolor`’ and ‘`numbercolor`’ are now really correctly reordered. Brackets (and) are now allowed in sequence names. The option `{case}` in ‘`funcshadingstyle`’ works now.

v1.2 1999/6/12

Corrections: (a) Functional group definitions of more than seven groups produced an error when displaying group number eight. These residues were skipped in the alignment. Fixed.

Introductions: (a) Protein secondary structure files in the DSSP, STRIDE and PHD format can be included and displayed automatically within the alignment by ‘`includeDSSP`’, ‘`includeSTRIDE`’, ‘`includePHDsec`’ and ‘`includePHDtopo`’ (4.5.4). (b) Which types of secondary structures are to be included or skipped in the alignment is chosen by ‘`showonDSSP`’, ‘`hideonDSSP`’, ‘`showonSTRIDE`’, ‘`hideonSTRIDE`’, ‘`showonPHDsec`’, ‘`hideonPHDsec`’, ‘`showonPHDtopo`’ and ‘`hideonPHDtopo`’. (c) The appearance of the labels is defined by ‘`appearance`’. (d) Internal counters for repeatedly occurring structure types can be activated by ‘`numcount`’, ‘`alphacount`’ and ‘`Alphacount`’. All commands are described in 4.5.4.

v1.1 1999/5/26

Corrections: (a) The activation of ‘`emphregion`’ lead to an emphasized following alignment. This has been corrected. (b) ‘`namecolor`’ and ‘`numbercolor`’ were not reordered with the command ‘`orderseqs`’. Fixed. (c) Sequence gaps at the beginning or the end of a sequence, i. e. before the first and after the last residue were labeled with the gap symbol. Now these positions are left blank.

Introductions: (a) In order to treat the preceeding and sequence following gaps correctly, `TeXshade` needs to know the length of the sequences. Therefore, the command ‘`seqlength`’ was introduced (4.4). (b) With ‘`gapcolors`’ (also 4.4) the color selction for gap symbols is independent from non conserved residues. (c) The divisions of the ruler where so long fixed to 10. Now, this value is changeable by ‘`rulersteps`’ (again 4.4). (d) ‘`hideresidues`’ and ‘`showresidues`’ turn off or on the residue names, i. e. one can choose between a display of shaded boxes only or with letters in the boxes (4.4.2). (e) The changes (c) through (d) were necessary for the introduction of ‘`fingerprint`’. This command allows one to display the complete sequence in one line for an easy survey of the alignment (4.4.4).

v1.0 1999/5/12

First release.

1.2 System requirements

`TEXshade` requires $\LaTeX 2_\epsilon$ and `color.sty` with the `dvips` option for shading. 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 usage of a `dvips` based style file implies that a POSTSCRIPT compatible \TeX viewer and the ability to print POSTSCRIPT files is required. This should be a minor problem due to the fact that the public domain POSTSCRIPT viewer/printer driver `Ghostview` 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.

1.3 The `texshade` environment

The commands provided by the `TEXshade` package are enabled by the following command in your document header section:

```
\usepackage{texshade}
```

Make sure that the file '`texshade.sty`' is present in a directory searched by \TeX (see the installation notes in the file '`texshade.txt`'). The `TEXshade` package provides only one single new environment: `texshade`. This environment has one mandatory and one optional argument, both of them designating file names which must be present in a directory searched by \TeX . The required file (*alignmentfile*) contains the aligned nucleotide or peptide sequences (see section 2). This file is needed, because `TEXshade` does no alignment by itself, it has to take a preprocessed alignment as input. The optional file is a parameter file (section 3) with definitions for the customized calculation of the consensus, special sequence features or labels etc. In this parameter file all `TEXshade` commands which are allowed in the `texshade` environment can be used and are fully functional. Within the environment further `TEXshade` commands can be given to replace or complete settings from the parameter file.

Thus, setting an alignment with `TEXshade` is as simple as this:

```

\begin{texshade}[\langle parameterfile \rangle]{\langle alignmentfile \rangle}
  further TeXshade commands, if needed
\end{texshade}

```

1.4 Shading modes predefined in this package

1.4.1 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.

```

80 TLGLLLSCQISILRAVMYIIAQCVGAIVASAAIL AQP1.PRO
72 TVACLVGCHVSFLRAAFYVAAQLLGAVAGAAIL AQP2.PRO
80 TFAMCFLAREPWIKLPIYTLAQTLLGAFLLGAGIV AQP3.PRO
101 TVAMVCTRKISIAKSVFYITAQCLGAIIGAGIL AQP4.PRO
73 TLALLIGNQISLLRAVFYVAAQLLVGAIAGAGIL AQP5.PRO

```

If you like, positions where all residues are identical can be shaded in a special color:

```

80 TLGLLLSCQISILRAVMYIIAQCVGAIVASAAIL AQP1.PRO
72 TVACLVGCHVSFLRAAFYVAAQLLGAVAGAAIL AQP2.PRO
80 TFAMCFLAREPWIKLPIYTLAQTLLGAFLLGAGIV AQP3.PRO
101 TVAMVCTRKISIAKSVFYITAQCLGAIIGAGIL AQP4.PRO
73 TLALLIGNQISLLRAVFYVAAQLLVGAIAGAGIL AQP5.PRO

```

1.4.2 Similarity mode

In many cases it is expedient—mostly when comparing protein sequences—to shade also residues which are not identical but similar to the consensus sequence. Consider a 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.

impressive when printed in grayscale. Enjoy them on your screen or use color printouts. As mentioned before, all colors can be changed to others or to grays without restrictions (see chapter 5).

charge: residues which are charged at physiological pH (7.4) are shaded if their number at a position is higher than the threshold

```

138  GLGIEIIGTLQLVLCVLATTDR. RRRDLGGSAPL  AQP1.PRO
130  AVTVELFLTMQLVLCIFASTDE. RRGNLGGSPAL  AQP2.PRO
153  GFFDQFIGTAALIVCVLAIVDPYNNPVPRGLEAF  AQP3.PRO
159  GLLVELIITFQLVFTIFASCDS. KRDTVTSVAL  AQP4.PRO
131  AMVVELILTFQLALCIFSSTDS. RRTSPVGGSPAL  AQP5.PRO
  
```

X acidic (-)
X basic (+)

hydropathy: discrimination between acidic and basic, polar uncharged and hydrophobic nonpolar residues

```

138  GLGIEIIGTLQLVLCVLATTDR. RRRDLGGSAPL  AQP1.PRO
130  AVTVELFLTMQLVLCIFASTDE. RRGNLGGSPAL  AQP2.PRO
153  GFFDQFIGTAALIVCVLAIVDPYNNPVPRGLEAF  AQP3.PRO
159  GLLVELIITFQLVFTIFASCDS. KRDTVTSVAL  AQP4.PRO
131  AMVVELILTFQLALCIFSSTDS. RRTSPVGGSPAL  AQP5.PRO
  
```

X acidic (-)
X basic (+)
X polar uncharged
X hydrophobic nonpolar

structure: displays the potential localization within the tertiary structure of the protein

```

      transmembrane domain 4      trans. dom. 5
      GM      MMMG
      AG      GGG A
      YN      NNNY
      α-helix
138  GLGIEIIGTLQLVLCVLATTDR. RRRDLGGSAPL  AQP1.PRO
130  AVTVELFLTMQLVLCIFASTDE. RRGNLGGSPAL  AQP2.PRO
153  GFFDQFIGTAALIVCVLAIVDPYNNPVPRGLEAF  AQP3.PRO
159  GLLVELIITFQLVFTIFASCDS. KRDTVTSVAL  AQP4.PRO
131  AMVVELILTFQLALCIFSSTDS. RRTSPVGGSPAL  AQP5.PRO
      loop D
  
```

- X external
- X ambivalent
- X internal

chemical: residues are shaded due to chemical properties of their functional groups

```

138  GLGIEIIGTLQLVLCVLAATDR. RRRDLGGSAPL  AQP1.PRO
130  AVTVELFLTMQLVLCIFASTDE. RRGDNLGSPAL  AQP2.PRO
153  GFFDQFIGTAALIVCVLAIVDPYNNPVPRGLEAF  AQP3.PRO
159  GLLVELIITFQLVFTIFASCDS. KRTDVTGSVAL  AQP4.PRO
131  AMVVELILTFQLALCIFSSTDS. RRTSPVGSVAL  AQP5.PRO

```

- X acidic (-)
- X aliphatic
- X amide
- X aromatic
- X basic (+)
- X hydroxyl
- X imino
- X sulfur

With `\shadeallresidues` the threshold is ignored and all residues are shaded due to their group assignment. This is *not* identical to a threshold of 0% where only the majority group would be shaded. See the difference:

```

138  GLGIEIIGTLQLVLCVLAATDR. RRRDLGGSAPL  AQP1.PRO
130  AVTVELFLTMQLVLCIFASTDE. RRGDNLGSPAL  AQP2.PRO
153  GFFDQFIGTAALIVCVLAIVDPYNNPVPRGLEAF  AQP3.PRO
159  GLLVELIITFQLVFTIFASCDS. KRTDVTGSVAL  AQP4.PRO
131  AMVVELILTFQLALCIFSSTDS. RRTSPVGSVAL  AQP5.PRO

```

standard area: this shading displays the surface area sizes of the different amino acid's sidechains

```

138  GLGIEIIGTLQLVLCVLAATDR. RRRDLGGSAPL  AQP1.PRO
130  AVTVELFLTMQLVLCIFASTDE. RRGDNLGSPAL  AQP2.PRO
153  GFFDQFIGTAALIVCVLAIVDPYNNPVPRGLEAF  AQP3.PRO
159  GLLVELIITFQLVFTIFASCDS. KRTDVTGSVAL  AQP4.PRO
131  AMVVELILTFQLALCIFSSTDS. RRTSPVGSVAL  AQP5.PRO

```

X	88.1 (G); Standard sidechain area (Å ²)
X	118.2 (A); 129.8 (S)
X	146.1 (C); 146.8 (P)
X	152.5 (T); 158.7 (D); 164.5 (V); 165.5 (N)
X	181.0 (I); 186.2 (E)
X	193.1 (L); 193.2 (Q); 202.5 (H); 203.3 (M)
X	222.8 (F); 225.8 (K)
X	238.8 (Y)
X	256.0 (R); 266.2 (W)

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**)

		<i>membr.</i>		<i>loop</i>		<i>membr.</i>		
		helix				helix		
138	GLGIEIIGTLQLVLCVLA	TTDR	.RRRD	LGG	SAPL			AQP1.PRO
130	AVTVELFLTMQLVLCIF	ASTDE	.RRGD	NLG	SPAL			AQP2.PRO
153	GFFDQFIGTAALIVCV	LAI	VD	PYNN	PVPR	GLEAF		AQP3.PRO
159	GLLVELIITFQLVFTI	FAS	CDS	.KRTD	VTGS	VAL		AQP4.PRO
131	AMVVELILTFQLALC	IFS	STDS	.RR	TSPV	GS	PAL	AQP5.PRO

X	13.9 (C); Accessible sidechain area (Å ²)
X	23.0 (I); 23.5 (V); 25.2 (G)
X	28.7 (F); 29.0 (L); 30.5 (M); 31.5 (A)
X	41.7 (W); 44.2 (S); 46.0 (T); 46.7 (H)
X	53.7 (P)
X	59.1 (Y); 60.9 (D); 62.2 (N)
X	72.3 (E); 74.0 (Q)
X	93.8 (R)
X	110.3 (K)

1.4.5 Secondary structures

Predicted protein secondary structures in the DSSP, STRIDE or PHD file format can be included and displayed in the alignment. As a sample, the following few commands show an aquaporin alignment with the PHD topology data for aquaporin type 1 (top sequence).

```

\begin{texshade}{AQPpro.MSF}
  \shadingmode[allmatchspecial]{similar}
  \includePHDtopo{1}{AQP1.phd}
\end{texshade}

```

Abbr.: *int.* – internal; *ext.* – external; *TM* – transmembrane domain

1	MAS.....EIKKKLFW	AQP1.PRO
1	MW.....ELRSIAFS	AQP2.PRO
1	M.....NRCG.....EMLHIRYR.....LL	AQP3.PRO
1	MSDGAAARRWGKCGPPCSRESIMVAFKGVWTQAFW	AQP4.PRO
1	MK.....KEVCSLAF	AQP5.PRO
	! *** * **	consensus

int. A

		<i>ext. B</i>	
	TM1		
12	RAVVAEFLAMTLFVVISIGSALGFNYPLERNQTLV		AQP1.PRO
11	RAVLAEFLATLLFVFFGLGSALQWA...SS...P		AQP2.PRO
16	RQALAECLGLTLLVMFGCGSVAQVVLSRGTHTGGF.		AQP3.PRO
36	KAVTAEFLAMLIFVLLSVGSTINWG...GSENPLP		AQP4.PRO
12	KAVFAEFLATLIFVFFGLGSALKWP...SA...L		AQP5.PRO
	****! *!*****!*****!**** *		consensus

int. A

	<i>ext. B</i>		
	TM2		
47	QDNVKVSLAFGLSIATLAQSVGHISGAHSNPAVTL		AQP1.PRO
39	PSVLQIAVAFGLGIGILVQALGHVSGAHINPAVTV		AQP2.PRO
50	...LTINLAFGFAVTLAILVAGQVSGAHLNPAVTF		AQP3.PRO
68	VDMVLISLCFGLSIATMVQCFGHISGGHINPAVTV		AQP4.PRO
40	PTILQISIAFGLAIGTLAQALGPVSGGHINPAITL		AQP5.PRO
	** *****!***** *!*****!*****!		consensus

int. C

ext. F

	-----	TM6	
205	..NFS.N.....HWIFWVGF	IGSALAVL..IYD	AQP1.PRO
197	..KFD.D.....HWVFWIGPLVGA	IIGSL..LYN	AQP2.PRO
221	LAGWGSEVFTTQQNW..WWVP	IVSPLLGSIGGVFV	AQP3.PRO
226	..NWE.N.....HWIYWVGPI	IGAVLAGA..LYE	AQP4.PRO
198	..RFS.SPS.....HWVFWVGP	IVGAMLAAI..LYF	AQP5.PRO
	*	*!***!***!*****	* **
		consensus	

229	FILAPRSSDFTDRMK.....VWTS....	AQP1.PRO	
221	YLLFPSAKSLQERL..AVLKG.LEPDTE	WEEREVR	AQP2.PRO
254	YQL.....	AQP3.PRO	
250	YV.FCPDVELKRRLKEAFSKAAQQT	KGSYMEVEDN	AQP4.PRO
223	YLLFPSLSLHDRV..AVVKGT	YEPEEDWEDHREE	AQP5.PRO
	*****	* ***	* **
		* *	* *
		consensus	

int. G

248	.GQVEEYDLDAD.....DINSRVEMKPK	AQP1.PRO	
253	RRQ..SVELHSPQSLPRG.....	AQP2.PRO	
257	..MIGCHLEQPPPSTEAEENV.KLAHM	KHKE....	AQP3.PRO
284	RSQVETEDLILKPGVVHVIDIDRGDEK	KGKDSSGE	AQP4.PRO
256	RKK..TIELTAH		AQP5.PRO
	* ** * *!	*	* *
		consensus	

int. G

270	AQP1.PRO
269	..SKA AQP2.PRO
284	...QI AQP3.PRO
319	VLSSV AQP4.PRO
266	AQP5.PRO
	consensus

1.4.6 Sequence fingerprints

To gain a quick overview of sequence similarities or properties the `fingerprint` command has been implemented. It can depict the complete sequence in one single line. The residues are presented as colored vertical lines. The implementation of this kind of output was inspired by the publication of KAI-UWE FRÖHLICH [6].

shown. Numbering and rulers can be displayed and set to any value. A powerful tool is the `\feature` command which allows one to label stretches of residues with bars, arrows, braces or any fill character and describing text. Legends are set automatically if desired, but user commands are also provided to build individual legends.

2 Format of alignment input files

`TeXshade` can handle two common alignment input formats, i.e. the `.MSF` format (multiple sequences format) and the `.ALN` format (alignment format). The `.MSF` format is used by PILEUP of the Unix GCG sequence analysis package². Files in the `.ALN` format are produced by CLUSTAL and CLUSTALV available for free for Unix, DOS and Macintosh. In addition to this software many alignment programs have export filters for the `.MSF` or the `.ALN` format, e.g. MACAW produces `.ALN` files. If you are not sure whether your favorite sequence aligner produces one of the required formats compare its output to the following examples. `TeXshade` determines the format from the internal file structure, thus the extensions `.MSF` or `.ALN` are not required. If you can choose between both formats `.MSF` is recommended, because this format gives information about the sequence type, i.e. peptide or nucleotide sequences, and length (for the correct setting of gaps at the sequence end).

2.1 The `.MSF` file format

Files of this type are divided into a header section and the multiple sequence alignment. The header may contain the following components:

File Type: (optional) The first header line reads for nucleic acids alignments
!!NA_MULTIPLE_ALIGNMENT 1.0 and for amino acid sequences
!!AA_MULTIPLE_ALIGNMENT 1.0 (all uppercase).

Description: (optional) Informative text describing what is in the file.

²For a description see <http://gene.md.huji.ac.il/Computer/GCG9doc>

Dividing line: (required!) Must include the following attributes:

MSF: Displays the number of bases or residues in the multiple sequence alignment.

Type: Displays the sequence type, 'P' for a peptide and 'N' for a nucleotide alignment.

Checksum: Displays an integer value that characterizes the contents of the file.

.. The two periods act as a divider between the descriptive information and the following sequence information.

Name/Weight: (required!) Must include the name of each sequence included in the alignment, as well as its length, checksum and weight.

Two slashes (//): (required!) This separating line divides the name/weight information from the sequence alignment

The alignment section consists of sequence blocks divided by an empty line. Each sequence line starts out with the sequence name. An example file is shown here:

```
AQP.MSF MSF: 87 Type: P May 1st, 1998 Check: 2586 ..
Name: AQP1.PRO Len: 66 Check: 1367 Weight: 1.00
Name: AQP2.PRO Len: 58 Check: 2176 Weight: 1.00
Name: AQP3.PRO Len: 83 Check: 1893 Weight: 1.00
Name: AQP4.PRO Len: 63 Check: 3737 Weight: 1.00
Name: AQP5.PRO Len: 59 Check: 3413 Weight: 1.00
//
          1                               45
AQP1.PRO MAS.....EIKKKLFWRAVVAEFLAM
AQP2.PRO MW.....ELRSIAFSRAVLAEFLAT
AQP3.PRO M.....NRCG.....EMLHIRYR.....LLRQALAECLGT
AQP4.PRO MSDGAAARRWGKCGPPCSRESIMVAFKGVWTQAFWKAVTAEFLAM
AQP5.PRO MK.....KEVCSLAFFKAVFAEFLAT

          45                               87
AQP1.PRO TLFVFISIGSALGFNYPLERNQTLVQDNVKVSLAFGLSIATL
AQP2.PRO LLFVFFGLGSALQWA...SS...PPSVLQIAVAFGLGIGIL
AQP3.PRO LILVMFGCGSVAQVVLSRGTGGF...LTINLAFGFAVTLA
AQP4.PRO LIFVLLSVGSTINWG...GSENPLPVDMLISLCFGLSIATM
AQP5.PRO LIFVFFGLGSALKWP...SA...LPTILQISIAFGLAIGTL
```

`TeXshade` extracts only this information from the file it really needs. So, do not mind all the checksums listed in the file—`TeXshade` does not either. The same is true for `Weight`. Required are the string `MSF:` for the identification of the file format and `Type:` for the determination of the sequence type (both in the dividing line), further all `Name:` definitions and finally `//`. The `.MSF` format allows one to comment out sequences. This is done by putting an exclamation point directly in front of the respective `Name`. These sequences are neither displayed nor used for the calculation of the consensus. This works for `TeXshade`, too. To comment out sequences without changing the input file use the `TeXshade` command `\killseq{⟨seqnumber⟩}` (4.4.2).

```

AQP.MSF MSF: 87  Type: P  May 1st, 1998  Check: 2586 ..
Name: AQP1.PRO  Len: 66  Check:  1367    Weight: 1.00
!Name: AQP2.PRO  Len: 58  Check:  2176    Weight: 1.00
!Name: AQP3.PRO  Len: 83  Check:  1893    Weight: 1.00
Name: AQP4.PRO  Len: 63  Check:  3737    Weight: 1.00
Name: AQP5.PRO  Len: 59  Check:  3413    Weight: 1.00
//

          1                                     45
AQP1.PRO  MAS.....EIKKKLFWRAVVAEFLAM
AQP2.PRO  MW.....ELRSIAFSRAVLAEFLAT
AQP3.PRO  M.....NRCG.....EMLHIRYR.....LLRQALAECLGT
AQP4.PRO  MSDGAAARRWGKCGPPCSRESIMVAFKGVWTQAFWKAVTAEFLAM
AQP5.PRO  MK.....KEVCSLAFFKAVFAEFLAT

          45                                     87
AQP1.PRO  TLFVFISIGSALGFNYPLERNQTLVQDNVKVSLAFGLSIATL
AQP2.PRO  LLFVFFGLGSALQWA...SS...PPSVLQIAVAFGLGIGIL
AQP3.PRO  LILVMFGCGSVAQVVLSRGTHGGF...LTINLAFGFAVTLA
AQP4.PRO  LIFVLLSVGSTINWG...GSENPLPVDMLISLFCGLSIATM
AQP5.PRO  LIFVFFGLGSALKWP...SA...LPTILQISIAFGLAIGTL

```

If the sequence lengths are given after `Len:` these values are used by `TeXshade`. Be aware that some alignment programmes calculate the sequence length by summing up residues and additionally gaps which is not really correct. In order to have the sequence break right after the last residue without printing further gap symbols make sure that the right lengths are assigned. You can either modify the `.MSF`-file or use the command `\seqlength` in the `TeXshade` environment.

2.2 The .ALN file format

.ALN files are quite similar to the above described .MSF files. They simply lack a defined header section. Nevertheless, describing text is allowed before the alignment part. `TeXshade` determines the number of sequences and their names from the last sequence block—so, no further text lines are allowed after this block! Due to a lacking declaration in the file the sequence type has to be set in the `texshade` environment by `\seqtype{<type>}` with ‘P’ for peptide and ‘N’ for nucleotide sequences; for the example below: `\seqtype{P}`. If no `\seqtype` command is used `TeXshade` assumes a nucleotide sequence.

```

                                profalign  May 1st, 1998, 16:58

of AQPpro.MSF{}

Multiple alignment parameter:

Gap Penalty (fixed):           10.00
Gap Penalty (varying):         .05
Gap separation penalty range:   8
Percent. identity for delay:   0%
List of hydrophilic residue:   GPSNDQEKRH
Protein Weight Matrix:         blosum

                                10      20      30      40
AQP1.PRO  MAS.....EIKKKLFWRAVVAEFLAM
AQP2.PRO  MW.....ELRSIAFSRAVLAEFLAT
AQP3.PRO  M.....NRCG....EMLHIRYR.....LLRQALAECLGT
AQP4.PRO  MSDGAAARRWGKCGPPCSRESIMVAFKGVWTQAFWKAVTAEFLAM
AQP5.PRO  MK.....KEVCSLAFFKAVFAEFLAT
          *                               .   ** *.

AQP1.PRO  TLFVFISIGSALGFNYPLERNQTLVQDNVKVSLAFGLSIATL
AQP2.PRO  LLFVFFGLGSALQWA...SS...PPSVLQIAVAFGLGIGIL
AQP3.PRO  LILVMFGCGSVAQVVLSRGTHGGF...LTINLAFGFAVTLA
AQP4.PRO  LIFVLLSVGSTINWG...GSENPLPVDMLISLCFGLSIATM
AQP5.PRO  LIFVFFGLGSALKWP...SA...LPTILQISIAFGLAIGTL
          .. *   .**                               .   ** .

```

The minimal contents of an `.ALN` file are shown below; this is fully sufficient. Many sequence alignment programs can produce such an output. Have a look at `seqpup` by DON GILBERT if you need a comprehensive conversion program³.

```

AQP1.PRO MAS.....EIKKKLFWRAVVAEFLAM
AQP2.PRO MW.....ELRSIAFSRAVLAEFLAT
AQP3.PRO M.....NRCG....EMLHIRYR.....LLRQALAECLGT
AQP4.PRO MSDGAAARRWGKCGPPCSRESIMVAFKGVWTQAFWKAVTAEFLAM
AQP5.PRO MK.....KEVCSLAFFKAVFAEFLAT

AQP1.PRO TLFVFISIGSALGFNYPLERNQTLVQDNVKVSLAFGLSIATL
AQP2.PRO LLFVFFGLGSALQWA...SS...PPSVLQIAVAFGLGIGIL
AQP3.PRO LILVMFGCGSVAQVVLRSRGTGGF...LTINLAFGFAVTLA
AQP4.PRO LIFVLLSVGSTINWG...GSENPLPVDMLISLCFGLSIATM
AQP5.PRO LIFVFFGLGSALKWP...SA...LPTILQISIAFGLAIGTL

```

3 Use of a `TeXshade` parameter file

Using predefined parameter files for repeatedly occurring 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 me⁴ 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 `TeXshade` distribution to make them available for all interested users.

No special file format is required for parameter files. `TeXshade` 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 `TeXshade` called `texshade.def`. This file can be changed freely and can be used as a template for the creation of personal parameter files.

Five steps are executed by `TeXshade` when processing the `texshade` environment:

³Sorry, `seqpup` is much more!

⁴`eric.beitz@uni-tuebingen.de`

```
\begin{texshade}[\langle parameterfile \rangle]{\langle alignmentfile \rangle}
```

1. Analysis of the $\langle alignmentfile \rangle$; determination of the number of sequences and sequence names
2. Setting parameters to default
3. Setting parameters to the definitions of the $\langle parameterfile \rangle$, if existent
4. Execution of further `TeXshade` commands within the environment, if existent

```
\end{texshade}
```

5. Loading and setting the alignment on a line by line basis

4 texshade user commands

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

```
\usepackage{texshade}
```

Then, the `texshade` environment is ready to use as described in 1.3. See also section 3 for a description of the optional parameter file. All other commands provided by `TeXshade` (except `\molweight` and `\charge`, see 4.8) must be used within the `texshade` environment.

4.1 Using predefined shading modes

If no `\shadingmode` command is given in the `texshade` environment the default shading mode (*identical*, see 1.4.1) is active. For the selection of one of the other predefined shading the following command is provided.

```
\shadingmode[\langle option \rangle]{\langle mode \rangle}
```

You can choose from four shading modes and declare one option which is dependent from the selected mode.

1. `\shadingmode[\langle allmatchspecial \rangle]{\code{identical}}`

There is not much to explain here (see 1.4.1). Use the option `allmatchspecial` to shade positions in a special color where all

residues are identical. `\allmatchspecial` can also be used as a command. As both, option or command `allmatchspecial` is only active in the *identical* and *similar* shading modes.

One can choose from five predefined shading color schemes with the command `\shadingcolors{<scheme>}`. The sets are named ‘blues’ (used in the example, 1.4.1), ‘reds’, ‘greens’, ‘grays’ and ‘black’. Default is `\shadingcolors{blues}`. Further, the colors for the non matching, the conserved and all matching residues can be set individually plus the letter case (lower or upper) or any character can be chosen:

```
\nomatchresidues{<res.col.>}{<shad.col.>}{<case>}{<style>}
\conservedresidues{<res.col.>}{<shad.col.>}{<case>}{<style>}
\allmatchresidues{<res.col.>}{<shad.col.>}{<case>}{<style>}
```

How to handle colors for the foreground `<res.col.>` and the background `<shad.col.>` see section 5. The third parameter `<case>` tells `TeXshade` to print the corresponding residue as a lowercase or an uppercase letter or even to print any other character. Finally, the `<style>` parameter tells `TeXshade` which shape to use for the letters. Use one of the following styles for `<style>`.

<code><style></code>	<i>effect</i>
<code>bf</code>	bold face series
<code>md</code>	normal series
<code>up</code>	upright shape (normal shape)
<code>it</code>	italics shape
<code>sl</code>	slanted shape
<code>rm</code>	modern roman family
<code>sf</code>	sans serif family
<code>tt</code>	typewriter family

In order to change only some of the parameters it is sufficient to declare these and use empty braces for the others. Examples:

`\conservedresidues{White}{Blue}{upper}{bf}`: the conserved residues are printed as bold face white uppercase letters on blue.

`\nomatchresidues{}{}{ \bullet }`: instead of the non matching residues a ‘•’ is printed. The colors and style are not changed. Mind the double curly braces which makes `TeXshade` interpreting this complex symbol description as one single character.

2. `\shadingmode[allmatchspecial]{similar}`

See 1.4.2 for an example output and the explanation of the shading. In addition to the described commands for changing shading colors this shading mode provides the command `\similarresidues`. Use it in analogy to the commands above.

How does `TeXshade` know which residues are considered to be similar? Well, this is set by two command couples, i. e. `\pepsims`,`\pepgroups` for peptides and `\DNAsims`,`\DNAgroups` for nucleotides. With `\pepsims` and `\DNAsims` residues are defined which are similar to the consensus residue. Examples:

`\pepsims{S}{TA}` If a serine is the consensus residue then all threonins and alanines at this position are shaded in the color for similars. This definition does *not* imply that threonine and alanine are similar to each other! This becomes obvious when inspecting the next definition.

`\pepsims{T}{S}` Serine but not alanine is declared to be similar to threonine.

What happens if there is no consensus residue? How does `TeXshade` decide if a group of similars is greater than the threshold? That is easy:

`\pepgroups{FYW,ILVM,RK,DE,GA,ST,NQ}` This command allows one to set up to nine groups of similars, separated by commas. Each residue can belong to only one group. If one residue is assigned to several groups only the last assignment is carried out.

`\DNAgroups{GAR,CTY}` This command is used in analogy to the amino acid groups. Here, two ambiguity codes (‘R’ for purine base and ‘Y’ for pyrimidine base) are assigned in addition.

Residues which do not appear in any of the four commands are considered not to belong to a group. Here, the default settings for similars are listed:

```

\pepgroups{FYW,ILVM,RK,DE,GA,ST,NQ}

\pepsims{F}{YW} % Y and W are similar to F
\pepsims{Y}{WF} % W and F are similar to Y
\pepsims{W}{YF} % Y and F are similar to W

\pepsims{I}{LVM} % L, V and M are similar to I
\pepsims{L}{VMI} % V, M and I are similar to L
\pepsims{V}{MIL} % Y, F and L are similar to V

\pepsims{R}{KH} % K and H are similar to R
\pepsims{K}{HR} % H and R are similar to K
\pepsims{H}{RK} % R and K are similar to H

\pepsims{A}{GS} % G and S are similar to A
\pepsims{G}{A} % A (but not S) is similar to G

\pepsims{S}{TA} % T and A are similar to S
\pepsims{T}{S} % S (but not A) is similar to T

\pepsims{D}{EN} % E and N (but not Q) are similar to D
\pepsims{E}{DQ} % D and Q (but not N) are similar to E
\pepsims{N}{QD} % Q and D (but not E) are similar to N
\pepsims{Q}{NE} % N and E (but not D) are similar to Q

\DNAGroups{GAR,CTY}

\DNAsims{A}{GR} % G and R are similar to A
\DNAsims{G}{AR} % A and R are similar to G
\DNAsims{R}{AG} % A and G are similar to R

\DNAsims{C}{TY} % T and Y are similar to C
\DNAsims{T}{CY} % C and Y are similar to T
\DNAsims{Y}{CT} % C and T are similar to Y

```

3. `\shadingmode[⟨seqnumber⟩]{diverse}`

1.4.3 depicts an example alignment. As option `⟨seqnumber⟩` the consensus sequence can be selected. If no `⟨seqnumber⟩` is declared the first sequence is set as consensus (`⟨seqnumber⟩ = 1`).

4. `\shadingmode[⟨type⟩]{functional}` There are six different functional shading modes available for peptide sequences; nu-

cleotide sequences can not be shaded due to functional aspects. Four of `TeXshade`'s functional modes correspond to the four 'alphabets' employed by KARLIN and GHANDOUR for peptide alignments [2]. Additional 'alphabets' to the standard 20-letter array of amino acids can highlight peptide similarities which were otherwise not visible. For the 'alphabet' definitions see below:

- $\langle type \rangle = \text{charge}$ Acidic (D, E) and basic (H, K, R).
- $\langle type \rangle = \text{hydropathy}$ Acidic and basic (as above), polar uncharged (C, G, N, Q, S, T, Y) and hydrophobic nonpolar (A, F, I, L, M, P, V, W), see also KYTE and DOOLITTLE [3].
- $\langle type \rangle = \text{structure}$ External (D, E, H, K, N, Q, R), internal (F, I, L, M, V) and ambivalent (A, C, G, P, S, T, W, Y).
- $\langle type \rangle = \text{chemical}$ Acidic (D, E), aliphatic (A, G, I, L, V), amide (N, Q), aromatic (F, W, Y), basic (H, K, R), hydroxyl (S, T), imino (P) and sulfur (C, M).

The modes below highlight sidechain sizes and hydrophobicity, respectively, according to ROSE *et al.* [4,5]. Standard area stands for the surface area of the residue in \AA^2 , i. e. it is a measure for the size of a residue's sidechain. The accessible area value (also in \AA^2) gives information about the size of the surface area which is accessible by solvent molecules within the folded protein. A very small area means that the residue is strongly buried and is thus very hydrophobic. Hydrophilic residues in turn possess large accessible areas due to their preferred location at the protein surface. Therefore, this kind of shading provides another method, in addition to `hydropathy` and `structure`, for the visualization of structural protein properties.

- $\langle type \rangle = \text{standard area}$ for the area values see legend of the alignment in 1.4.4
- $\langle type \rangle = \text{accessible area}$ for values see 1.4.4

If no $\langle type \rangle$ or an unknown $\langle type \rangle$ is designated as option all functional groups and shading colors are cleared. This is also achieved by the command `\clearfuncgroups`. With all groups

cleared one can start to build new shading modes from scrap. How to do this is explained in the next section.

In order to exchange the colors but to keep the group definitions and descriptions the command `\funcshadingstyle` can be employed. Usage:

```
\funcshadingstyle{<residue>}{<res.col.>}{<shad.col.>}
                                     {<case>}{<style>}
```

`<residue>` is one representative of the whole amino acid group. The colors which are declared by the next four parameters are used for all residues in this group. `<case>` and `<style>` are as described for example in `\nomatchresidues`.

4.2 Creating new functional shading modes

The grouping of amino acids due to other properties can make sense as suggested by KARLIN and GHANDOUR [2], e. g. physical properties (molecular weight, shape), kinetic properties (reaction velocity, Michaelis-Menton constant), or structure (α -helices, β -sheets, turns). New amino acid groups are defined with the `\funcgroup` command. This command needs six parameters:

```
\funcgroup{<descr>}{<residues>}{<res.col.>}{<shad.col.>}
                                     {<case>}{<style>}
```

`<descr>` contains descriptive text which is displayed in the legend. The second parameter `<residues>` holds the amino acids to be grouped. The colors for the foreground and background are set with the following two parameters, the case and style is declared by the last parameters. The example below defines a functional group named ‘acidic (–)’ containing the amino acids aspartic and glutamic acid with white letters on a red background:

```
\funcgroup{acidic ($-$)}{DE}{White}{Red}{upper}{up}
```

For the usage of colors see section 5. Up to nine individual groups can be defined. New groups are simply added to the already existing groups, i. e. if an extension of the group definitions of an existing shading mode is desired there is no need to clear these groups und re-define them again. Just add the new groups with the `\funcgroup` command. To create completely new modes use the command `\shadingmode{functional}` without an option *before* setting

the new groups. The new definitions are active only in the functional shading mode—so be sure to have it switched on before setting the new groups. Remember, `\shadingmode{functional}` without an optional parameter clears all groups defined before, see above. The following example shows the definitions needed to produce an output which is identical to the functional mode ‘charge’:

```
\begin{texshade}{\langle alignmentfile \rangle}
  \shadingmode{functional}
  \funcgroup{acidic ($-$)}{DE}{White}{Red}{upper}{up}
  \funcgroup{basic ($+$)}{HKR}{White}{Blue}{upper}{up}
\end{texshade}
```

4.3 Appearance of the consensus line

An important parameter for the calculation of the consensus is the threshold percentage. Default setting is 50%, i.e. to become the consensus residue more than half of the residues at this position must be identical or similar, depending on the shading mode. Any percentage between 0 and 100 is allowed and can be set with `\threshold{\langle percentage \rangle}`, e.g. `\threshold{50}`.

Another possibility is to set one sequence of the alignment as consensus and compare the other sequences to this one. Therefore, the command `\constosingleseq{\langle seqnumber \rangle}` is provided. The `\langle seqnumber \rangle` selects the sequence to be used as consensus (numbering according to the appearance in the alignment file; top sequence is number 1). Nevertheless, the threshold percentage is also taken into account, i.e. with a threshold of 50% half of the sequences must be identical or similar compared to the specified consensus sequence in order to be shaded. With `\constoallseqs` the consensus is calculated considering all sequences (the case described in the paragraph above).

Consensus lines are displayed either on the top or at the bottom of the alignment by calling `\showconsensus{\langle position \rangle}` with `\langle position \rangle` `top` or `bottom`. To hide the consensus use `\hideconsensus`. The consensus line is named ‘consensus’ in English texts or ‘Konsensus’ if the `german.sty` is used. With `\nameconsensus{\langle name \rangle}` any name can be set.

You can tell `TEXshade` which symbols or letters to use in the consensus line for different matching qualities by

```
\defconsensus{\langle symbol1 \rangle}{\langle symbol2 \rangle}{\langle symbol3 \rangle}.
```

The following parameters are allowed for symbols 1–3:

1. $\langle symbol1 \rangle$ = no match symbol (if below threshold)
 - any character or letter
 - $\{\}$ (empty braces) for blank space
2. $\langle symbol2 \rangle$ = conserved symbol (if threshold is exceeded)
 - **upper** (prints the consensus residue in uppercase)
 - **lower** (prints the consensus residue in lowercase)
 - any character or letter
 - $\{\}$ (empty braces) for blank space
3. $\langle symbol3 \rangle$ = all match symbol (if all residues match and `\allmatchspecial` is active)
 - see $\langle symbol2 \rangle$

Example: `\defconsensus{\{\}\{*\}\{upper\}` does not show non matching residues in the consensus line, marks conserved residues with ‘*’, and displays the uppercase letter of the consensus residue at positions where all residues match.

Finally, the colors of the above defined symbols are adjustable by the command:

```
\consensuscolors{\langle res.col.1 \rangle}{\langle shad.col.1 \rangle}
                {\langle res.col.2 \rangle}{\langle shad.col.2 \rangle}
                {\langle res.col.3 \rangle}{\langle shad.col.3 \rangle}
```

The color definitions are in the same order as in the `\defconsensus` command:

1. $\langle res.col.1 \rangle$ = no match residue color (if below threshold)
 $\langle shad.col.1 \rangle$ = no match background color
2. $\langle res.col.2 \rangle$ = conserved residue color (if threshold is exceeded)
 $\langle shad.col.2 \rangle$ = conserved background color

3. $\langle res.col.3 \rangle$ = all match residue color (if all residues match and `\allmatchspecial` is active)
 $\langle shad.col.3 \rangle$ = all match background color

For colors which are not to be changed empty braces can be used.
 Example:

```
\consensuscolors{}{}{Blue}{White}{Red}{Green}
```

Non matching symbol colors are not changed, conserved residues are displayed blue on white and where all residues match red symbols on green ground are displayed in the consensus line.

4.4 Appearance of the sequence lines

4.4.1 Names, numbers and gaps

Many parameters for influencing the appearance of the actual sequence lines can be changed for customization. Thus, the sequence names and numbering can be shown and placed either left or right by

```
\shownames{\langle position \rangle}
\shownumbering{\langle position \rangle}
```

with $\langle position \rangle$ set to `left` or `right`. Both, names and numbering can be displayed on the same side.

`TEXshade` uses the sequence names from the alignment input file. If it is desired to change these names it is not necessary to edit the alignment file. A simple command in the `texshade` environment does the job:

```
\nameseq{\langle seqnumber \rangle}{\langle name \rangle}
```

$\langle seqnumber \rangle$ selects the sequence whose name is to be changed. The basis for the $\langle seqnumber \rangle$ is the appearance in the alignment input file with `top sequence = 1`. The colors are set by `\namescolor{\langle color \rangle}` and `\numberingcolor{\langle color \rangle}`, respectively.

In order to change the colors only of some sequence names or numbers the commands `\namecolor{\langle seq1 \rangle, \dots, \langle seq n \rangle}{\langle color \rangle}` and `\numbercolor{\langle seq1 \rangle, \dots, \langle seq n \rangle}{\langle color \rangle}` are provided.

In order to hide all names or the numbering use the command `\hidenames` or `\hidenumbering`. If only the names or numbers of some sequences should be hidden apply `\hidename{\langle seq1 \rangle, \dots, \langle seq n \rangle}` or

`\hidenumber{⟨seq1⟩, ... ,⟨seq n⟩}`, respectively.

In some situations, e. g. when only sections of sequences are displayed, one may not want to have the residue numbering start out with number 1. The command `\startnumber{⟨seqnumber⟩}{⟨first residue number⟩}` allows one to set the starting number of any sequence to any value incl. negative values but except ‘0’ which is not used in sequence numbering (the transition from negative to positive values is like this: ... -2, -1, 1, 2 ...).

TEXshade needs to know the correct lengths of the sequences to be able to break sequences right after the last residue. If `.MSF` files are used as input the length is already given (the calculation is sometimes wrong—the gaps might be included, so check this). Otherwise set the correct length by `\seqlength{⟨seqnumber⟩}{⟨length⟩}`.

Example: `\seqlength{1}{346}` means that sequence no. 1 is 346 residues long.

TEXshade can display a section of the complete alignment without the need to edit the alignment input file or even to re-calculate the entire alignment. This allows one to use one single alignment of the full length proteins or open reading frames for multiple visualizations of different sections in a document as done in this manual. Thus, the file `AQPpro.MSF` contains the full-length multiple protein alignment of five aquaporins but only sections are displayed as sample shading in 1.4.1 through 1.4.4. The definition of a section is done by

`\setends{⟨seqnumber⟩}{⟨startnumber⟩..⟨stopnumber⟩}`.

Again, `⟨seqnumber⟩` is the sequence number based on the appearance in the alignment file; further, in order to use the consensus as a measure for the sequence section the string ‘consensus’ as `⟨seqnumber⟩` is accepted. The specified sequence is truncated at positions `⟨startnumber⟩` and `⟨stopnumber⟩`. All other sequences are cut accordingly. If the number of the first residue in the sequence is set to a new value with the `\startnumber` command (s. a.) this is taken into account. Some examples:

a) `\setends{1}{20..100}`

b) `\startnumber{1}{15} \setends{1}{35..115}`

Both commands select the same section from the alignment but the numbering for sequence 1 starts at position 20 in the first example and at position 35 in the latter.

c) `\setends{consensus}{20..100}`

This may describe a completely different section of the multiple sequence alignment.

Another possibility to label sequence positions is to switch on a ruler on the top or at the bottom of the sequence block using `\showruler{<position>}{<seqnumber>}`. The residue ruler of one sequence `<seqnumber>` or the consensus (declare ‘consensus’ as `<seqnumber>`) can be displayed at `<position>` **top** or **bottom**. The ruler is hidden with `\hideruler`. The steps between two numbers are set by `\rulersteps{<number>}`. In order to change the color which is used for the ruler write `\rulercolor{<color>}`.

Further, the symbol which is displayed in sequence gaps is freely selectable with `\gapchar{<symbol>}`. `<symbol>` can be any character or symbol. If math symbols are to be used math mode must be activated by \$ characters, i. e. `\gapchar{${\triangle$}}`. Note the double curly braces in the last command. Everytime a ‘complex’ character is used, i. e. a character definition consisting of more than one letter, it must be braced in order to be interpreted as one character. One exception is `\gapchar{rule}`; with this parameter lines are drawn in the sequence gaps with a certain thickness defined by `\gaprule{<length>}`. The colors of the gaps and gaps symbols are set by `\gapcolors{<symbol color>}{<background color>}`.

4.4.2 Hiding, killing, separating and ordering

If one or more sequences from the alignment input file should be used for the calculation of the consensus but it is desired not to display these sequences in the final output use the command `\hideseq{<seq1>,<seq2>,...,<seq n>}`. This allows one for example to hide the sequence which has been defined as the consensus sequence with `\constosingleseq`. In order to completely exclude sequences the command `\killseq{<seq1>,<seq2>,...,<seq n>}` is provided. The designated sequences are neither displayed nor considered for the calculation of the consensus. This is another possibility to comment out sequences in addition to the use of an exclamation point in front of the **Name**: definition in an **MSF**-file (see figure on page 18).

The command `\donotshade{<seq1>,<seq2>...,<seq n>}` makes one or more sequences appear unshaded in black letters on white background. This does not influence any other sequences or the consensus calculation.

If a very graphical output of the sequences is desired, the residue symbols or letters can be blanked out by `\hideresidues`. Now, only

the shaded boxes are printed. In combination with `\gapchar{rule}` one obtains alignments in a style à la Mondrian. The residues reappear with `\showresidues`.

If an alignment contains members of several subgroups of a protein or a gene family it may be rather helpful to visualize the group divisions by a separation line. Therefore, the command `\separationline{<seqnumber>}` is applicable. This command inserts vertical space after the sequence which is referred to by `<seqnumber>`. How much space is inserted is defined by one of the following commands: `\smallsep`, `\medsep` (default) or `\bigsep`. These lengths correspond to the known `\small-`, `\med-` and `\bigskip` commands. With `\vsepspace{<length>}` any length with any T_EX unit can be assigned, e. g. `\vsepspace{2mm}`.

The sequence order given by the alignment input file is easily reorganized by `\orderseqs{<seq1>,<seq2>,...,<seq n>}` without the need for editing the alignment input file (which would be a big copy'n'paste job). Make sure that all sequence numbers are assigned in this command. If there are more sequences present than numbers in the command an error message will occur. Re-ordering sequences only changes the output; all commands using the parameter `<seqnumber>` are not influenced, because `<seqnumber>` always corresponds to the appearance in the alignment file. Thus, to completely reverse the order of a five sequence alignment simply type `\orderseqs{5,4,3,2,1}`.

4.4.3 Residues per line and further settings

By default T_EX`shade` puts the highest possible by five divisible amount of residues in one line depending on the `\textwidth`. With `\residuesperline{<number>}` a new value can be set. If this value exceeds the highest possible number of residues per line it is ignored; lower values are accepted of course. But also in the latter case the number of residues printed per line is rounded such to be divisible by five. To force T_EX`shade` to set lines with exactly the desired number of residues use the asterisk-extended command `\residuesperline*{<number>}`. Expect multiple *overfull hbox* errors after this command, because in this mode T_EX`shade` does not check the length of the lines any more.

T_EX`shade` calculates the dimensions of a shaded box from the width and height of the uppercase letter ‘M’ and the depth of the lowercase ‘g’. Depending on the font used for the sequence residues the box dimensions might not be fully satisfactory. With

`\charstretch{⟨factor⟩}` and `\linestretch{⟨factor⟩}` the width and height/depth, respectively, of the boxes can be multiplied individually by a *⟨factor⟩* to stretch (> 1) or shrink (< 1) the dimensions.

The reserved space for the sequence numbering is set by the command `\numberingwidth{⟨n digits⟩}`. Here, the default setting is four-digit numbering, i.e. -999 through 9999 . If this range is to be changed assign the desired number as parameter *⟨n digits⟩*, e.g. `\numberingwidth{111111}` reserves space for 6 digit numbering.

The vertical space between the sequence blocks can be controlled by the commands `\smallblockskip`, `\medblockskip` (default setting), `\bigblockskip` or `\noblockskip`. Further, the command `\vblockspace{⟨length⟩}` allows one to set a defined space length using any TeX unit, e.g. `\vblockspace{0.4in}`.

The position of the output can be aligned left, right or centered on the page by `\alignment{⟨position⟩}` with the *⟨position⟩* parameter `left`, `center` or `right`.

4.4.4 Fingerprinting

An easy way to gain an overview on complete alignments is provided by displaying a so called alignment ‘fingerprint’. In this style the whole sequence can be shown in one line. Due to the lacking space the residue names are hidden and the shaded boxes are reduced to thin vertical colored lines. The command `\fingerprint{⟨res. per line⟩}` takes one argument stating the desired number of residues per line, e.g. `\fingerprint{1000}`. All TeXshade commands are compatible with `\fingerprint`, i.e. all shading modes are applicable for displaying overviews on similarity or every functional aspect. Also, all kinds of labeling—as described in the following—work with this command.

4.5 Individual shading and labeling of sequence stretches

Computer calculated alignment shading is informative—but even more information can be visualized by manual labeling of positions and regions of interest with different colors, text styles or graphical marks and descriptive text. All this is provided by easy to handle TeXshade commands.

4.5.1 Manual shading of regions and blocks

Besides the shading calculated by `TeXshade` any region can be shaded manually in a color specified by the user. This is very useful to highlight secondary protein modification sites such as phosphorylation or glycosylation sites or longer motifs for example protein/protein interaction sites. All this is done by use of the following command:

```
\shaderegion{⟨seqnumber⟩}{⟨start1⟩..⟨stop1⟩,⟨start2⟩..⟨stop2⟩,  
... ,⟨start n⟩..⟨stop n⟩}{⟨res.col.⟩}{⟨shad.col.⟩}
```

Example: in order to shade the residue number 13 and the region 20–30 of sequence number 1 in red letters on green ground type the following command:

```
\shaderegion{1}{13..13,20..30}{Red}{Green}
```

If the consensus is to be shaded use `consensus` as `⟨seqnumber⟩`.

In analogy to `\shaderegion` which is restricted to one single sequence `\shadeblock` shades the corresponding region in all other sequences as well except the consensus. If also the consensus is to be shaded define the region using `consensus` as `⟨seqnumber⟩`.

```
\shadeblock{⟨seqnumber⟩}{⟨start1⟩..⟨stop1⟩,⟨start2⟩..⟨stop2⟩,  
... ,⟨start n⟩..⟨stop n⟩}{⟨res.col.⟩}{⟨shad.col.⟩}
```

4.5.2 Emphasizing regions and blocks

If it is preferred to keep the calculated shading colors but distinct regions or blocks are yet to be emphasized use the following commands to change the font style of such regions:

```
\emphregion{⟨seqnumber⟩}{⟨start1⟩..⟨stop1⟩,⟨start2⟩..⟨stop2⟩,  
... ,⟨start n⟩..⟨stop n⟩}
```

and

```
\emphblock{⟨seqnumber⟩}{⟨start1⟩..⟨stop1⟩,⟨start2⟩..⟨stop2⟩,  
... ,⟨start n⟩..⟨stop n⟩}
```

Which style `TeXshade` uses for emphasizing regions is defined by `\emphdefault{⟨style⟩}`. Default setting is the *italics* font shape (set by `\emphdefault{it}`). In order to change this setting choose one of the styles `bf`, `md`, `up`, `it`, `sl`, `rm`, `sf`, `tt`.

Example: `\emphdefault{bf}`

4.5.3 Graphical labeling of sequence features

The `\feature` command is designed to fulfill most needs for the graphical labeling of sequence stretches and the setting of descriptive text. It needs five parameters:

```
\feature{<position>}{<seqnumber>}{<start1>..<stop1>},
        <start2>..<stop2>,...,<start n>..<stop n>}{<labelstyle>}{<text>}
```

In the following paragraphs all possible parameter settings of this rather complex but mighty command are discussed in detail. The parameter `<position>` tells `TeXshade` where to display the feature label, i. e. on the top of the alignment (`top`), or at the bottom (`bottom`). Further, there can be a feature line on the top of the top feature line (`ttop`) or below the bottom feature line (`bbottom`). Thus, up to four features overlapping in four different lines may be displayed. The argument `<seqnumber>` and the third parameter containing the definitions of the specified regions are identical to the ones described before in several commands, e. g. `\ruler` (4.4.1) or `\shaderegion` (4.5.1). New is the fourth parameter for the definition of the label style. There are many possibilities like braces, boxes, arrows, bars, any fill character or even translations of the specified regions. In order to display an over- or underbrace as a label use the parameter `{brace}`. Depending on the `<position>` (`ttop`, `top`, `bottom` or `bbottom`) the respective brace is displayed. A region can be filled with any character for labeling purposes using the parameter `{fill:<symbol>}`. The `<symbol>` is freely selectable; the usage is like in `\gapchar` (4.4.1). Do not use spaces before or after the expression `<symbol>`; this will shift the symbols to the respective direction. The standard color of the fill symbol is black. It can be changed by an optional parameter directly after the definition of the symbol, e. g. `{fill:\bullet[Red]}`. Boxed text is printed using the parameter `{box:<text>}`. By default black letters in a white framed box are displayed. In order to change these colors optional parameters can be included in the argument: `{box[<boxcolor>]:<text>[<textcolor>]}`.

Example: `{box[Blue]:α~helix[Yellow]}`

For displaying bars and arrows a simple selection scheme consisting of three consecutive characters is used as the `<labelstyle>` parameter. Each bar or arrow is defined by its left end, the middle part, and the right end. The following table gives some examples for the construction of arrows and bars.

middle	left end ↓ right end
---	plain bar
===	double bar
-->	right arrow
'->	right arrow with up hook
<-	left <i>maps to</i> arrow
<=>	double arrow, two heads
,-,	plain bar with down hooks
=	double bar with vertical ends

All combinations of the left-end-characters (`--<`, `|`), the middle-characters (`--`), and the right-end-characters (`-->`, `'`, `|`) are allowed and produce the desired arrow or bar. The color is changed as described above. For plain bars the thickness can be set by `\featurerule{<length>}` with any T_EX measure as `<length>`, e.g. `\featurerule{3pt}`.

With the option `{translate}`, sequence stretches can be translated from nucleotide to peptide sequences and even backtranslations from peptide to nucleotide sequences are possible. Default setting for the translations is the standard genetic code. Of course, the codons can be re-defined by the user. The command `\codon{<amino acid>}{<triplet1, ..., triplet n>}` has been implemented for this issue. The usage is simple. Replace `<amino acid>` by the single letter code of the amino acid to be defined and add a list of triplets for this residue. Example definition for the amino acid *alanine*:

```
\codon{A}{GCA,GCG,GCC,GCT,GCU,GCN}
```

Note the last triplet in the list. It contains an ambiguity code N which stands for *any* nucleotide. This triplet has been added at the last position because the last triplet is used for the generation of the backtranslated nucleotide sequence from a peptide. Two files are included in the T_EXshad^e distribution as examples (`standard.cod`, `ciliate.cod`). If you want to define a new genetic code store your commands in a file like the examples. Such files with the suffix `.cod` can be loaded in the T_EXshad^e environment by `\geneticcode{<filename>}`, e.g. `\geneticcode{ciliate}`. Do not designate the suffix `.cod` in `<filename>`. Please note, when expecting the example files, that only the exchanges to the standard code must be defined in a new genetic code file.

When DNA sequences are translated to protein the resulting amino acids are printed atop of the second nucleotide of each triplet. It

is more difficult to produce a satisfactory display of backtranslated nucleotide sequences due to the lack of space. You need thrice as much space than the original peptide sequence, because single letter amino acid code is translated to a triplet code. Therefore, the user can choose from five display styles for backtranslations depending on personal preferences:

`\backtranslabel[$\langle size \rangle$]{ $\langle style \rangle$ }`, with

```

{ $\langle style \rangle$ } = {horizontal}
                = {alternating}
                = {zigzag}
                = {oblique}
                = {vertical}

```

$\langle size \rangle$ can be any TeX size from `tiny` up to `Huge`, well `tiny` is most recommended (and default setting). Translations can be colored as all other labels, see above.

If no graphical label is wanted the fourth parameter of `\feature` can be empty braces.

Finally, the fifth parameter of the `\feature` command contains the descriptive text for the labeled region. Type whatever you want incl. symbols and math chars. The text field can also contain sequence translations. In this case just set $\langle text \rangle = \{\text{translate}\}$. There is a command for setting the size and style of backtranslated sequences in the feature $\langle text \rangle$ which corresponds to the one described above:

`\backtranstext[$\langle size \rangle$]{ $\langle style \rangle$ }`, with

Again, the color can be set by an optional parameter appended to the text.

Examples are given in the overview section (1), see:

similarity mode (1.4.2): fill-character; here, only one position is labeled. It is also possible to label a longer stretch, then, the character is printed several times to fill the specified region.

```

\feature{top}{1}{93..93}{fill:$\downarrow$}{first...}
\feature{bottom}{1}{98..98}{fill:$\uparrow$}{second...}

```

diversity mode (1.4.3): no graphical label, text only

```

\feature{top}{1}{77..109}{{AQP2 species variants}}

```

functional mode (1.4.4): box, arrow, translation and brace

```

\feature{top}{1}{138..157}

```

```

        {box[Red]:$\alpha$~helix[Yellow]}
        {transmembrane domain 4}
\feature{top}{1}{164..170}{,->[Red]}{trans. dom. 5}
\feature{top}{1}{158..163}{translate[Blue]}{}
\backtranslabel{oblique}
\feature{bottom}{1}{158..163}{brace[Blue]}{loop D[Blue]}

```

4.5.4 Including secondary structure information

The DSSP [7], STRIDE [8] and PHD [9] algorithms produce very reliable secondary protein structure predictions. PHD files contain both, secondary structure information and topology data. This information can be displayed in an alignment by one of the commands:

```

\includeDSSP      sec. structure calculated by the DSSP software
\includeSTRIDE    sec. structure calculated by STRIDE
\includePHDsec    sec. structure calculated by PHD
\includePHDtopo   topology data calculated by PHD

```

The syntax is `\includeDSSP{<seqnum>}{<filename>}`, with `seqnum` indicating the sequence for which the secondary structure data is calculated and `filename` designating the corresponding structure file to be included.

Several types of secondary structures are predicted by these programs; in order to designate them in `TeXshade` use the names from the right column:

secondary structure	designation
<i>DSSP and STRIDE</i>	
4-helix (α -helix)	alpha
isolated β -bridge	bridge
extended strand (β -strand)	beta
3-helix (3_{10} -helix)	3-10
5-helix (π -helix)	pi
H-bonded turn	turn
<i>PHDsec</i>	
helix	alpha
sheet	beta
<i>PHDtopo</i>	
internal region	internal
external region	external
transmembrane domain	TM

By default all three types of helices and the strands are displayed whereas turns and bridges are skipped. If it is desired to show them, too, call for example `\shownonDSSP{bridge,turn}`. In analogy to this example all types of secondary structure can be activated in DSSP, STRIDE, PHDsec and PHDtopo. In order to hide certain structure types use for example `\hideonDSSP{3-10,pi}`.

Now, some information about how `TEXshade` extracts and displays secondary structure features. In short, it is a two step process. First, `TEXshade` analyzes the secondary structure file and extracts all necessary data. This data is converted into a format which is readable and processable by `TEXshade`, the `feature` command (see 4.5.3). This command allows one to label sequence stretches graphically. For a detailed explanation see the indicated reference. A list of feature commands is saved in a file with the ending `.sec` for DSSP, STRIDE and PHDsec or `.top` for PHDtopo. Then, in a second step, this file is loaded again and executed. When `TEXshade` encounters this file a second time, i. e. in a second `TEX` run, it uses the already existing file for the output. The great advantage of this method is its flexibility. Due to the simple reason that the feature file can be edited in the meantime. Thus, the user has the ability to change the computer generated file according to his personal needs. On the other hand, one can force `TEXshade` to write a new file every time by the optional argument `[make new]` in the include command, e. g.

`\includePHDsec[make new]{1}{AQP.phd}`.

Finally, the appearance of the feature labels can be assigned by the command

`\appearance{<filetype>}{<type>}{<position>}{<labelstyle>}{<text>}`.

Here, `<filetype>` stands for one of the following secondary structure file types: DSSP, STRIDE, PHDsec or PHDtopo and `<type>` designates the secondary structure type as shown in the right column of the table above. The other arguments `<position>`, `<labelstyle>` and `<text>` are almost as described in 4.5.3. One further possibility is to include internal counters for each secondary structure type. Just add one of the following commands to the text in the feature description.

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

Examples:

```
\appearance{DSSP}{alpha}{ttop}{-->}{${\alpha$-helix~\Alphacount}
\appearance{PHDtopo}{TM}{bottom}{box[Blue]:TM\numcount[Yellow]}{}
```

4.6 Displaying and building legends

For each predefined shading mode `TeXshade` can print an appropriate legend to explain the used shading colors. The commands `\showlegend` and `\hidelegend` display or clear the legend at the end of the alignment. The language used for the descriptions is english by default; if the `\german.sty` package is active legend texts are in german. Yet, only english and german are implemented. With the commands `\germanlanguage` and `\englishlanguage` switching to german or english is made possible. For the addition of other languages contact me. Finally, the color of the describing legend texts can be set with the command `\legendcolor{<color>}`.

User defined legends are easily built with the following command `\shadebox{<color>}`. Use this command outside the `TeXshade` environment, e.g. in the text or in the caption. As `<color>` any color can be designated (see section 5) or one of the following parameters:

- `nomatch` = the color used for nonmatching residues

- `similar` = the color used for similar residues
- `conserved` = the color used for conserved residues
- `allmatch` = the color used for the case that all residues match (if `\allmatchspecial` is active)

The command simply prints a shaded box in the specified color then a describing text can be appended. Examples:

```
\shadebox{nomatch}---nonmatching residues
\shadebox{similar}: similar residues
\shadebox{conserved}~conserved residues
\shadebox{Yellow}\quad PKA phosphorylation sites
```

4.7 Font handling

4.7.1 Changing font styles

The font styles for the numbering, the sequence names, the sequence residues, the descriptive feature texts and the legends 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 `numbering`, `names`, `residues`, `features` and `legend`.

The style is set by the second parameter:

command	<i><2. parameter></i>	
<code>\setfamily</code>	<code>rm</code>	modern roman font family
	<code>sf</code>	sans serif font family
	<code>tt</code>	typewriter font family
<code>\setseries</code>	<code>bf</code>	bold face series
	<code>md</code>	normal series
<code>\setshape</code>	<code>it</code>	italics shape
	<code>sl</code>	slanted shape
	<code>sc</code>	small capitals shape
	<code>up</code>	upright shape
<code>\setsize</code>	<code>tiny</code>	the known T _E X sizes
	<code>scriptsize</code>	
	<code>footnotesize</code>	
	<code>small</code>	
	<code>normalsize</code>	
	<code>large</code>	
	<code>Large</code>	
	<code>LARGE</code>	
	<code>huge</code>	
	<code>Huge</code>	

Example: `\setfamily{features}{it} \setseries{features}{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{features}{rm}{it}{bf}{normalsize}`

Further, short commands are provided to change single font attributes quickly. The following commands set attributes of feature texts.

```

\featuresrm   \featurestiny
\featuresff   \featurescriptsize
\featurestt   \featuresfootnotesize
\featuresbf   \featuressmall
\featuresmd   \featuresnormalsize
\featuresit   \featureslarge
\featuressl   \featuresLarge
\featuressc   \featuresLARGE
\featuresup   \featureshuge
              \featuresHuge

```

Corresponding sets are provided for numbering (`\numberingrm` etc.), names (`\namesrm` etc.), residues (`\residuesrm` etc.) and legend texts (`\legendrm` etc.).

4.7.2 Using PostScript fonts

As already mentioned `TEXshade` makes intensive use of POSTSCRIPT for shading. Now, that POSTSCRIPT output is active anyway, including POSTSCRIPT fonts is very easy. Just declare in the document header

```
\usepackage{<PS-font>}
```

The typewriter font of `TEX` is always a topic of discussions. By including the package `\usepackage{courier}` `TEX`'s typewriter font is replaced by the widely accepted `COURIER`. Have a look into the directory `..texinputs:latex:psnfss`; there, some styles are located which exchange the common `TEX` fonts by POSTSCRIPT fonts, e.g. `avant.sty`, `bookman.sty`, `chancery.sty`, `courier.sty`, `helvet.sty` or `utopia.sty`. Depending on the style used the `\rmdefault`-, `\sfdefault`-, and `\ttdefault` fonts are substituted partly or completely. Thus, `courier.sty` for instance exchanges only the typewriter font, whereas `bookman.sty` sets `BOOKMAN` as `\rmdefault`, `AVANTGARDE` as `\sfdefault` and `COURIER` as `\ttdefault`.

For further information see TOMAS ROKICKI's `dvips` manual [10].

4.8 Goodies—molweight and charge

During the process of sequence setting `TEXshade` sums up the molecular weight and charge of the aligned proteins. This data can be accessed by the following two commands.

```
\molweight{<seqnum>}{<Da/kDa>}
\charge{<seqnum>}{<i/o/N/C>}
```

The first parameter `<seqnum>` selects the sequence due to the appearance in the alignment input file. The second parameter in the `\molweight` command allows one to switch the units between Dalton (Da) and kilo-Dalton (kDa). The `\charge` command needs the second parameter for the correct consideration of the charged protein termini. Thus, 'i' refers to internal sequences, 'o' to the overall charge, 'N' to N-terminal sequence parts, and 'C' to the C-terminal end of a protein.

Example: Charge: `\charge{1}{o}`; Weight: `\molweight{1}{Da}`

5 The DVIPS color selection scheme

POSTSCRIPT provides 64 standard colors. All these colors are pre-defined 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. `TEXshade` enhances this color scheme by gray scales in 5% steps. The following colors and grays can be used in `TEXshade` by simply declaring the name of the color in the respective command, e.g. `\consensuscolors`:

<i>name</i>	<i>CMYK</i>	<i>name</i>	<i>CMYK</i>
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}
```

6 Listing of the `texshade` default settings

6.1 Standard definitions

The file `texshade.def` mirrors all commands which are carried out at the beginning of the `texshade` environment. Short comments are also included, thus, it is referred to this file for further information.

6.2 Colors used in the different shading modes

Color scheme *blues*:

<i>res.color</i>	<i>shad.color</i>	<i>residues</i>
Black	White	no match
Black	Magenta	similar
White	RoyalBlue	identical
Goldenrod	RoyalPurple	all match

Color scheme *greens*:

<i>res.color</i>	<i>shad.color</i>	<i>residues</i>
Black	White	no match
Black	GreenYellow	similar
White	PineGreen	identical
YellowOrange	OliveGreen	all match

Color scheme *reds*:

<i>res.color</i>	<i>shad.color</i>	<i>residues</i>
Black	White	no match
Black	YellowOrange	similar
White	BrickRed	identical
YellowGreen	Mahagony	all match

Color scheme *grays*:

<i>res.color</i>	<i>shad.color</i>	<i>residues</i>
Black	White	no match
Black	LightGray	similar
White	DarkGray	identical
White	Black	all match

Color scheme *black*:

<i>res.color</i>	<i>shad.color</i>	<i>residues</i>
Black	White	no match
Black	White	similar
White	Black	identical
White	Black	all match

Functional mode *charge*:

<i>res.color</i>	<i>shad.color</i>	<i>residues</i>
Black	White	no match
White	Red	acidic
White	Blue	basic

Functional mode *hydropathy*:

<i>res.color</i>	<i>shad.color</i>	<i>residues</i>
Black	White	no match
White	Red	acidic
White	Blue	basic
Black	Yellow	polar uncharged
White	Green	hydrophobic nonpolar

Functional mode *chemical*:

<i>res.color</i>	<i>shad.color</i>	<i>residues</i>
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*:

<i>res.color</i>	<i>shad.color</i>	<i>residues</i>
Black	White	no match
Black	Orange	external
Black	Yellow	ambivalent
White	Green	internal

Functional mode *standard area*:

<i>res.color</i>	<i>shad.color</i>	<i>residues</i>
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*:

<i>res.color</i>	<i>shad.color</i>	<i>residues</i>
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

7 Quick Reference

The **T_EXshade** logo

```
\TeXshade
```

The **T_EXshade** environment (6 ff.)

```
\begin{texshade}[\langle parameterfile \rangle] {\langle alignmentfile \rangle}
  further TEXshade commands, if needed
\end{texshade}
```

Predefined shading modes

```
\seqtype{\langle type \rangle} (P – peptide, N – nucleotide) [19]
```

```
\shadingmode[\langle option \rangle]{\langle mode \rangle} [21]
```

<i>\langle mode \rangle</i>	<i>\langle option \rangle</i>
identical	allmatchspecial
similar	allmatchspecial
diverse	<i>\langle seqnumber \rangle</i>
functional	<i>\langle type \rangle</i> charge hydropathy structure chemical standard area accessible area

```
\shadeallresidues [10]
```

Shading colors (22 ff.)

```
\shadingcolors{\langle scheme \rangle} (blues, reds, greens, grays, black)
```

```
\nomatchresidues{\langle res.col. \rangle}{\langle shad.col. \rangle}{\langle case \rangle}{\langle style \rangle}
```

```
\similarresidues{\langle res.col. \rangle}{\langle shad.col. \rangle}{\langle case \rangle}{\langle style \rangle}
```

```
\conservedresidues{\langle res.col. \rangle}{\langle shad.col. \rangle}{\langle case \rangle}{\langle style \rangle}
```

```
\allmatchresidues{\langle res.col. \rangle}{\langle shad.col. \rangle}{\langle case \rangle}{\langle style \rangle}
```

```
\funcshadingstyle{\langle residue \rangle}{\langle res.col. \rangle}{\langle shad.color \rangle}
{\langle case \rangle}{\langle style \rangle} [26]
```

Residue grouping

<code>\pepsims{<residue>}{<similar>}</code>	[23]
<code>\pepgroups{<group1>,<group2>, ... , <groupn>}</code>	[23]
<code>\DNAsims{<residue>}{<similar>}</code>	[23]
<code>\DNAGroups{<group1>,<group2>, ... , <groupn>}</code>	[23]

Definition of new functional shading modes

<code>\clearfuncgroups</code>	[25]
<code>\funcgroup{<descr>}{<residues>}{<res.col.>}{<shad.col.>}</code> <code>{<case>}{<style>}</code>	[26]

Appearance of the consensus line

<code>\threshold{<percentage>}</code>	[27]
<code>\constosingleseq{<seqnumber>}</code>	[27]
<code>\showconsensus{<position>}</code>	[27]
<code>\hideconsensus</code>	[27]
<code>\nameconsensus{<name>}</code>	[27]
<code>\defconsensus{<symbol1>}{<symbol2>}{<symbol3>}</code>	[27]
<code>\consensuscolors{<res.col.1>}{<shad.col.1>}</code> <code>{<res.col.2>}{<shad.col.2>}</code> <code>{<res.col.3>}{<shad.col.3>}</code>	[28]

Appearance of the sequence lines

<code>\shownames{<position>}</code>	[29]
<code>\shownumbering{<position>}</code>	[29]
<code>\nameseq{<seqnumber>}{<name>}</code>	[29]
<code>\namescolor{<color>}</code>	[29]
<code>\namecolor{<seq1>, ... , <seq n>}{<color>}</code>	[29]
<code>\hidenames</code>	[29]
<code>\hidename{<seq1>, ... , <seq n>}</code>	[29]
<code>\numberingcolor{<color>}</code>	[29]
<code>\numbercolor{<seq1>, ... , <seq n>}{<color>}</code>	[29]
<code>\hidenumbering</code>	[29]
<code>\hidenumber{<seq1>, ... , <seq n>}</code>	[29]
<code>\hideresidues</code>	[31]
<code>\showresidues</code>	[31]
<code>\startnumber{<seqnumber>}{<first residue number>}</code>	[30]

<code>\seqlength{⟨seqnumber⟩}{⟨length⟩}</code>	[30]
<code>\setends{⟨seqnumber⟩}{⟨startnumber⟩}..⟨stopnumber⟩}</code>	[30]
<code>\showruler{⟨position⟩}{⟨seqnumber⟩}</code>	[31]
<code>\rulersteps{⟨number⟩}</code>	[31]
<code>\rulercolor{⟨color⟩}</code>	[31]
<code>\hideruler</code>	[31]
<code>\gapchar{⟨symbol⟩}</code> (incl. rule)	[31]
<code>\gapcolors{⟨symbol color⟩}{⟨background color⟩}</code>	[31]
<code>\fingerprint{⟨res. per line⟩}</code>	[33]

Hiding, killing, separating and ordering

<code>\hideseq{⟨seq1⟩,⟨seq2⟩,...,⟨seq n⟩}</code>	[31]
<code>\killseq{⟨seq1⟩,⟨seq2⟩,...,⟨seq n⟩}</code>	[31]
<code>\donotshade{⟨seq1⟩,⟨seq2⟩,...,⟨seq n⟩}</code>	[31]
<code>\separationline{⟨seqnumber⟩}</code>	[32]
<code>\smallsep</code>	[32]
<code>\medsep</code>	[32]
<code>\bigsep</code>	[32]
<code>\vsepspace{⟨length⟩}</code>	[32]
<code>\orderseqs{⟨seq1⟩,⟨seq2⟩,...,⟨seq n⟩}</code>	[32]

Residues per line and further settings

<code>\residuesperline{⟨number⟩}</code>	[32]
<code>\residuesperline*{⟨number⟩}</code>	[32]
<code>\charstretch{⟨factor⟩}</code>	[32]
<code>\linestretch{⟨factor⟩}</code>	[32]
<code>\numberingwidth{⟨n digits⟩}</code>	[33]
<code>\smallblockskip</code>	[33]
<code>\medblockskip</code>	[33]
<code>\bigblockskip</code>	[33]
<code>\noblockskip</code>	[33]
<code>\vblockspace{⟨length⟩}</code>	[33]
<code>\alignment{⟨position⟩}</code>	[33]

Individual shading and labeling of sequence stretches

<code>\shaderegion{⟨seqnumber⟩}{⟨start1⟩}..⟨stop1⟩,⟨start2⟩}..⟨stop2⟩, ... ,⟨start n⟩}..⟨stop n⟩}{⟨res.col.⟩}{⟨shad.col.⟩}</code>	[34]
<code>\shadeblock{⟨seqnumber⟩}{⟨start1⟩}..⟨stop1⟩,⟨start2⟩}..⟨stop2⟩, ... ,⟨start n⟩}..⟨stop n⟩}{⟨res.col.⟩}{⟨shad.col.⟩}</code>	[34]

`\emphregion{<seqnumber>}{<start1>..<stop1>,<start2>..<stop2>,
...,<start n>..<stop n>} [34]`
`\emphblock{<seqnumber>}{<start1>..<stop1>,<start2>..<stop2>,
...,<start n>..<stop n>} [34]`
`\emphdefault{<style>} [34]`
`\feature{<position>}{<seqnumber>}{<start1>..<stop1>,
<start2>..<stop2>,...,<start n>..<stop n>}{<labelstyle>}{<text>} [35]`
`{<labelstyle>} = {brace[<color>]}`
`= {fill:<symbol>[<textcolor>]}`
`= {box[<boxcolor>]:<text>[<textcolor>]}`
`= arrows and bars (-=<' ,|)(-)(-=>' ,|)`
`= {translate[<color>]}`

`\codon{<amino acid>}{<triplet1>,...,<triplet n>} [36]`
`\geneticcode{<filename>} [36]`
`\backtranslabel[<size>]{<style>} [37]`
`\backtranstext[<size>]{<style>} [37]`

`{<style>} = {horizontal}`
`= {alternating}`
`= {zigzag}`
`= {oblique}`
`= {vertical}`

`\featurerule{<length>} [36]`

Including secondary structure information

`\includeDSSP[make new]{<seqnum>}{<filename>} [38]`
`\includeSTRIDE[make new]{<seqnum>}{<filename>} [38]`
`\includePHDsec[make new]{<seqnum>}{<filename>} [38]`
`\includePHDtopo[make new]{<seqnum>}{<filename>} [38]`
`\showonDSSP{<structures>} [39]`
`\showonSTRIDE{<structures>} [39]`
`\showonPHDsec{<structures>} [39]`
`\showonPHDtopo{<structures>} [39]`
`\hideonDSSP{<structures>} [39]`
`\hideonSTRIDE{<structures>} [39]`
`\hideonPHDsec{<structures>} [39]`
`\hideonPHDtopo{<structures>} [39]`

<code>\appearance{⟨type⟩}{⟨position⟩}{⟨labelstyle⟩}{⟨text⟩}</code>	[40]
<code>\numcount</code>	[40]
<code>\alphacount</code>	[40]
<code>\Alphacount</code>	[40]
<code>\romancount</code>	[40]
<code>\Romancount</code>	[40]

Displaying and building legends

<code>\showlegend</code>	[40]
<code>\hidelegend</code>	[40]
<code>\germanlanguage, \englishlanguage</code>	[40]
<code>\legendcolor{⟨color⟩}</code>	[40]
<code>\shadebox{⟨color⟩}</code>	[40]

Font handling

<code>\setfamily{⟨text⟩}{⟨family⟩}</code>	[41]
<code>\setseries{⟨text⟩}{⟨series⟩}</code>	[41]
<code>\setshape{⟨text⟩}{⟨shape⟩}</code>	[41]
<code>\setsize{⟨text⟩}{⟨size⟩}</code>	[41]
<code>\setfont{⟨text⟩}{⟨family⟩}{⟨series⟩}{⟨shape⟩}{⟨size⟩}</code>	[42]
<code>\featuresrm</code> <code>\featurestiny</code>	[42]
<code>\featuresf</code> <code>\featurescriptsiz</code>	
<code>\featurestt</code> <code>\featuresfootnotesize</code>	
<code>\featuresbf</code> <code>\featuressmall</code>	
<code>\featuresmd</code> <code>\featuresnormalsize</code>	
<code>\featuresit</code> <code>\featureslarge</code>	
<code>\featuresl</code> <code>\featuresLarge</code>	
<code>\featuresc</code> <code>\featuresLARGE</code>	
<code>\featuresup</code> <code>\featureshuge</code>	
	<code>\featuresHuge</code>

Corresponding sets are provided for numbering (`\numberingrm` etc.), names (`\namesrm` etc.), residues (`\residuesrm` etc.) and legend texts (`\legendrm` etc.).

Goodies—molweight and charge

<code>\molweight{⟨seqnum⟩}{⟨Da/kDa⟩}</code>	[43]
<code>\charge{⟨seqnum⟩}{⟨i/o/N/C⟩}</code>	[43]

8 References

- [1] CARLISLE, D. The Standard L^AT_EX ‘Graphics Bundle’, `color.sty`.
- [2] 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.
- [3] KYTE, J.; DOOLITTLE, R. F. (1982) A simple method for displaying the hydropathic character of a protein. *J. Mol. Biol.*: **157**, 105–132.
- [4] 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.
- [5] LESSER, G. J.; ROSE, G. D. (1990) Hydrophobicity of amino acid subgroups in proteins. *Proteins: structure, function and genetics*: **8**, 6–13.
- [6] FRÖHLICH, K.-U. (1994) Sequence similarity presenter: a tool for the graphic display of similarities of long sequences for use in presentations. *Comput. Applic. Biosci.*: **10**, 179–183.
- [7] KABSCH, W.; SANDER, C. (1983) Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*: **22**, 2577–2637.
- [8] FRISHMAN, D.; ARGOS, P. (1995) Knowledge-based protein secondary structure assignment. *Proteins: structure, function and genetics*: **23**, 566–579.
- [9] ROST, B.; SANDER, C. (1994) Combining evolutionary information and neural networks to predict protein secondary structure. *Proteins: structure, function and genetics*: **19**, 55–72.
- [10] ROKICKI, T. DVIPS: A T_EX driver.