Supplementary materials for KalignP

Nanjiang Shu[1] and Arne Elofsson[1]

[1]Department of Biochemistry and Biophysics, Stockholm Bioinformatics Center, Center for Biomembrane Research, Swedish e-science Research Center; Stockholm University; 106 91 Stockholm; Sweden

Contents

1 Implementation of KalignP

KalignP inherits the syntax of Kalign2 and extends its functionality so that externally supplied position specific gap penalty (ESPSGP) can be input into the alignment program to change the behavior of the alignment. The ESPSGP can be supplied in the Enhanced FASTA format as described in Section **5**. If ESPSGP is supplied, it will replace the default gap penalty at each residue position of the sequences to be aligned. KalignP uses the same progressive method as Kalign2. For sequence to sequence alignment, gap penalty at each residue position of the sequences will be set as the ESPSGP. That is, for a given sequence, the gap penalty at position i is defined as

$$GP(i) = ESPSGP(i) \tag{1}$$

After aligning two sequences A and B, a profile C will be generated. The preliminary gap penalty of profile C is calculated from the pairwise alignment of sequence A and B by

$$GP^C(m) = \begin{cases} GP^A(i) + GP^B(j) & \text{if aligned} \\ GP^A(i) & \text{if gap at B} \\ GP^B(j) & \text{if gap at A} \end{cases} \tag{2}$$

where $GP^C(m)$ is the gap penalty for profile C at position m, $GP^A(i)$ is the gap penalty for sequence A at position i and $GP^B(j)$ is the gap penalty for sequence B at position j.

When aligning a profile to a sequence, or a profile to a profile, the gap penalty of the profile and the sequence will be weighted in the same way as the original Kalign2 (Lassmann *et al.*, 2009) with the consideration that positions with no or few gaps should be punished harder to open a gap.

If the input file is a normal FASTA file or the ESPSGP are all set to 1, the alignment will be the same as that of Kalign2. With the implementation of ESPSGP, KalignP itself takes about 50% more CPU time than the original Kalign2 to do a common alignment. The whole KalignP package, including automatic secondary structure prediction and ESPSGP calculation, is about 15 times slower than Kalign2. Nevertheless, this speed is still comparable to many other MSA programs such as ClustalW (Thompson *et al.*, 1994) and Muscle (Edgar, 2004) and with better performance.

KalignP inherits one of the best merits from Kalign2: low memory consumption. Since KalignP uses the same progressive method as Kalign2, it consumes almost the same amount of memory as Kalign2. KalignP does need a little more memory to store position specific gap penalties for each sequence but this is very small (at the level of 1/N, where N is the average sequence length) compared to the memory used in the progressive alignment. Further, we calculated the heap memory usage by "valgrind" and the the result are so that KalignP uses about 2.6% more memory than Kalign2 on average when tested on Balibase 3.0.

2 Estimation of ESPSGP from predicted secondary structure

The principle to estimate ESPSGP from the predicted secondary structure is simple: increase the gap penalty at helices and sheets while lowering the gap penalty at coils. ESPSGP for position i is defined by

$$ESPSGP(i) = GP_{default} * \big(1.0 + m(i)\big) \tag{3}$$

For helices and sheets, $m(i)$ is defined as

$$m(i) = weight_{HE} * weight_{pos} * \big(p(i) - shift_{HE}\big) \tag{4}$$

where $weight_{HE}$ is the weight for helices and sheets, $p(i)$ is the reliability of the prediction that ranges from 0 to 1, and $shift_{HE}$ is a parameter to adjust the prediction reliability for helices and sheets. $weight_{pos}$ is a parameter to adjust the effect of the residue position in helices and sheets, taking into account that the predictions at the center of helices and sheets are usually more accurate than the two ends (Zhou *et al.*, 2010). $weight_{pos}$ for position j in the helices or sheets is defined as:

$$weight_{pos}(j) = \begin{cases} 0.05 & \text{if } j = 0 \text{ or } N-1 \\ 1.0 - (j + 1 - mid)/mid & \text{for other } j \end{cases} \tag{5}$$

where j is the position of the residue at helices/sheets, N is the length of the segment, and $mid = N/2$.

For loops, $weight_{pos}$ is not applied and thus $m(i)$ is defined as

$$m(i) = weight_C * \big(p(i) - shift_C\big) \tag{6}$$

where $weight_C$ is the weight for coils and $shift_C$ is a parameter to adjust the prediction reliability for coils.

Note that in the Enhanced FASTA file only the scale $\big(1.0 + m(i)\big)$ is supplied. This scale will be multiplied by the default gap penalty $GP_{default}$ when input into the KalignP program, see Section 5 for details.

We used PSIPRED_single (version 3.2) (Jones, 1999) to predicted the secondary structure due to its high speed and relatively high accuracy.

3 Optimizing parameters for KalignP for the benchmark on Balibase 3.0

KalignP makes improvement over Kalign2 by using ESPSGP estimated from secondary structure predicted by PSIPRED_single (version 3.2) (Jones, 1999). To estimate ESPSGP from

the predicted secondary structure with the method described in Section 2, we need to optimize four parameters, $weight_{HE}$, $weight_C$, $shift_{HE}$ and $shift_C$. Often, over-optimistic results can be obtained by over training a method on a particular benchmark dataset. To avoid this problem caused by direct parameter optimization, we split the benchmark dataset randomly into three subsets and performed a standard three-fold cross-validation. KalignP was run on each subset with parameters optimized from the other two subsets by grid searching. The final parameters were taken as the average of the three. The results of each fold of the cross-validation can be found at **Supplementary Table 4**.

4 Benchmark data

Due to the size limitation, all benchmark data are not supplied in this document but are available for download at http://kalignp.cbr.su.se/download/benchmark

5 Enhanced FASTA

The enhanced FASTA format is the same as FASTA format except that gap penalty arrays can be appended below the sequence. Gap penalties are enclosed in curly brackets {}. The gap open array, gap extension array and the terminal gap extension array are recognized by the tags "gpo:", "gpe:" and "tgpe:" respectively. If there is a leading char '#' within the brackets {}, this gap penalty array is commented out. Gap penalties can be supplied in the following two formats, but can not be mixed in the same input file.

1. An array of real values enclosed in {} and delimited by white spaces. e.g. {gpo: 1 1 1.5 1.5 1.5 0.5 0.5}. In this case, the number of real values should be exactly the same as the number of residues in the sequence.

2. Index:value, e.g. 1-15:2.0 means the residues 1 to 15 are assigned with value 2.0. For unassigned residues, the default value is 1.

The value of each element is not the absolute gap penalty but the scale of the default gap penalty for KalignP, e.g. {gpo: 1:2.0} means that the gap open penalty of the first residue will be set to 2.0*10, if the default gap open penalty is 10. We use the scale instead of the absolute gap penalty since it is easier to understand how position specific gap penalty varies. Moreover, it avoids possible conflicts when the general gap penalty is reset by the "-gpo", "-gpe" or "-tgpe" option of KalignP.

An example is shown below in both formats. The sequence is `ASNLSKLFLSDSDA`. All three types of gap penalty arrays are supplied, but gap extension array and terminal gap penalty array are commented out by the leading char '#' and thus only the gap open array is effective.

5.1 Example in format 1:

Note that the size of the array should be exactly the same as the length of the sequence.

```
>seq1
ASNLSKLFLSDSDA
{gpo:  1.5 1.5 1.5 1.5 1.5 0.5 0.5 0.5 0.5 0.5 1 1 1 1}
{#gpe:  2 2 2 2 2 0.4 0.4 0.4 0.4 0.4 1 1 1 1}
{#tgpe:  2 2 2 2 2 2 2 2 2 2 1 1 1 1 }
```

5.2 Example in format 2:

```
>seq1
ASNLSKLFLSDSDA
{gpo:  1-5:1.5 6-10:0.5}
{#gpe:  1-5:2.0 6-10:0.4}
{#tgpe:  1-10:2.0}
```

6 How KalignP works

KalignP changes the behavior of the alignment intuitively with the externally supplied position specific gap penalties. Generally speaking, if the users want to force a gap after the first residue of a sequence, set the gap open penalty at the second residue position of that sequence to a small value. For example, for the following four sequences,

```
>seq1
ASNLSKLFLSDSDA
>seq2
ASNLDA
>seq3
ASNLKFFFDDDAA
>seq4
LLNFFSDAAAAA
```

The multiple sequence alignment (MSA) with all parameters set to default is as follows (in ClustalW format)

```
seq1 ASNLSKLFLSDSDA--
```

```
seq2 ASNLDA----------
seq3 ASNLKF-FFDDDAA--
seq4 LLN----FFSDAAAAA
```

The tree of the MSA is $(((seq1, seq2), seq3), seq4)$. If the users want to open a gap after the second residue in $seq1$, the users can set the gap open penalty at the third position to be a minus value (e.g. -5). An example setting of ESPSGP is as follows. To ensure that only one gap is opened after the second residue 'S' in $seq1$ so that we can see clearly the effect of gap open, we have set the gap open penalties for $seq2$ and the gap open penalty at the fourth position of $seq1$ to be a large positive value.

```
>seq1
ASNLSKLFLSDSDA
{gpo:  1 1 -5 10 1 1 1 1 1 1 1 1 1 }
>seq2
ASNLDA
{gpo:  10 10 10 10 10 10 }
>seq3
ASNLKFFFDDDAA
>seq4
LLNFFSDAAAAA
```

The alignment will become

```
seq1 AS-NLSKLFLSDSDA--
seq2 -ASNLDA----------
seq3 -ASNLKF-FFDDDAA--
seq4 ---LLN--FFSDAAAAA
```

If the users want again to open a gap after the 11^{th} residue 'D' in $seq3$, set the gap open penalty of the 11^{th} position in $seq3$ to a minus value (e.g. -5). An example ESPSGP setting is as follows. Again, to ensure that gap will only be opened after the 11^{th} residue in $seq3$, we have set the gap open penalty of $seq4$ and the neighboring residue positions in $seq3$ to large positive values.

```
>seq1
ASNLSKLFLSDSDA
{gpo:  1 1 -5 10 1 1 1 1 1 1 1 1 1 }
>seq2
ASNLDA
{gpo:  10 10 10 10 10 10 }
```

```
>seq3
ASNLKFFFDDDAA
{gpo:  1 1 1 1 1 1 1 1 1 10 -5 20 10 }
>seq4
LLNFFSDAAAAA
{gpo:  10 10 10 10 10 10 10 10 10 10 10 10}
```

The alignment will become

```
seq1 ---------AS-NLSKLFLSDSDA
seq2 ----------ASNLDA--------
seq3 ASNLKFFFDD----DAA-------
seq4 ------------LLNFFSDAAAAA
```

The alignment above might not be ideal since there are many gaps at the terminals. To reduce the number of terminal gaps, one can increase the terminal gap extension penalty as shown in the example below.

```
>seq1
ASNLSKLFLSDSDA
{gpo:  1 1 -5 10 1 1 1 1 1 1 1 1 1 1 }
{tgpe:  10 10 10 10 10 10 10 10 10 10 10 10 10 10}
>seq2
ASNLDA
{gpo:  10 10 10 10 10 10 }
{tgpe:10 10 10 10 10 10}
>seq3
ASNLKFFFDDDAA
{gpo:  1 1 1 1 1 1 1 1 1 10 -5 20 10}
{tgpe:  10 10 10 10 10 10 10 10 10 10 10 10 10}
>seq4
LLNFFSDAAAAA
{gpo:  10 10 10 10 10 10 10 10 10 10 10 10}
{tgpe:10 10 10 10 10 10 10 10 10 10 10 10}
```

The alignment will become

```
seq1 AS-NLSKLFLSDS-DA
seq2 -ASNLDA---------
seq3 AS-NLKFFFDD-D-AA
seq4 ---LLNFFSDAAAAA-
```

Sometimes, the gap will not be opened at the expected position if many customized gap penalties are set in multiple sequences. This is because KalignP forbid neighboring gap opens at two aligned sequences such as the following example

```
ALDDS-D-S
ALED-D-S-
```

7 Supplementary tables

Supplementary Table 1: TC (column) score of KalignP and other multiple sequence alignment methods on Balibase 3.0 (Thompson *et al.*, 2005). The TC score was calculated by the bali_score version 3.0 and the core blocks defined in the xml file were used. Note that the values shown in table have been multiplied by 100 for easy reading.

Method	CPU[a] (s)	RV11[1]	RV12[2]	RV20[3]	RV30[4]	RV40[5]	RV50[6]	All
Kalign2	57	40.6	79.2	40.8	49.3	50.4	42.1	52.2
KalignP	95 (974[b])	42.2	80.0	40.8	51.2	52.3	44.5	53.4
Mafft	223	29.2	75.6	38.0	47.8	48.0	48.5	48.5
ClustalW	1281	32.2	75.1	33.5	37.8	39.6	36.1	44.4
Muscle	1281	42.9	81.5	41.8	47.3	45.0	47.1	52.8
T_coffee	25293	51.4	85.9	48.7	56.4	54.2	60.2	60.5
Probcons	26085	52.4	86.3	50.3	59.7	53.2	59.0	61.4

[1]Reference 1: equi-distant sequences with <20% identity. [2]Reference 1: equi-distant sequences with 20-40% identity. [3]Reference 2: families aligned with highly divergent "orphan" sequences. [4]Reference 3: subgroups with <25% residue identity between groups. [5]Reference 4: sequences with N/C-terminal extensions. [6]Reference 5: sequences with internal insertions. [a]The CPU time refers to single threaded process running on a Linux box with Intel quad-core 2.50GHz CPU and 4GB memory. [b]KalignP running time including secondary structure prediction and ESPSGP calculation

Supplementary Table 2: The versions and commands of the programs used in the benchmark on Balibase 3.0 and pfamA_mem (see **Supplementary Table 3**).

Method	Version	Command
Kalign2	v 2.04	kalign2
KalignP	v 1.0	kalignP
Mafft	v 6.847b	mafft
ClustalW	2.0.10	clustalw
Muscle	v 3.8.31	muscle
T_coffee	v 8.97_101117	t_coffee
Probcons	v 1.12	probcons

Supplementary Table 3: List of PfamA seed alignments in the dataset PfamA-mem. It was derived from Pfam-A Finn *et al.* (2009) (version 24.0) seed alignments and filtered by matching the key word "transmembrane" in the DE (definition) record.

No.	AC number	No. of seq	Definition
1	PF00001.14	64	7 transmembrane receptor (rhodopsin family)
2	PF00002.17	32	7 transmembrane receptor (Secretin family)
3	PF00003.15	110	7 transmembrane sweet-taste receptor of 3 GCPR
4	PF01490.11	26	Transmembrane amino acid transporter protein
5	PF00664.16	70	ABC transporter transmembrane region
6	PF06472.8	26	ABC transporter transmembrane region 2
7	PF09608.3	49	Putative transmembrane protein (Alph_Pro_TM)
8	PF05232.5	181	Bacterial Transmembrane Pair family
9	PF01891.9	12	Cobalt uptake substrate-specific transmembrane region
10	PF04505.5	69	Interferon-induced transmembrane protein
11	PF04549.7	13	CD47 transmembrane region
12	PF05602.5	28	Cleft lip and palate transmembrane protein 1 (CLPTM1)
13	PF09125.3	2	Cytochrome C oxidase subunit II, transmembrane
14	PF02790.8	32	Cytochrome C oxidase subunit II, transmembrane domain
15	PF02683.8	10	Cytochrome C biogenesis protein transmembrane region
16	PF12077.1	14	Transmembrane protein of unknown function (DUF3556)
17	PF12089.1	24	Transmembrane domain of unknown function (DUF3566)
18	PF04148.6	19	Transmembrane adaptor Erv26
19	PF09721.3	102	Transmembrane exosortase (Exosortase_EpsH)
20	PF01794.12	215	Ferric reductase like transmembrane component
21	PF09163.4	40	Formate dehydrogenase N, transmembrane
22	PF03409.8	19	Transmembrane glycoprotein
23	PF12369.1	14	Gonadotropin hormone receptor transmembrane region
24	PF10192.2	23	Rhodopsin-like GPCR transmembrane domain
25	PF06814.6	15	Lung seven transmembrane receptor
26	PF09726.2	5	Transmembrane protein
27	PF09773.2	7	Meckelin (Transmembrane protein 67)
28	PF03821.9	3	Golgi 4-transmembrane spanning transporter
29	PF00939.12	11	Sodium:sulfate symporter transmembrane region
30	PF10716.2	24	NADH dehydrogenase transmembrane subunit
31	PF02932.9	50	Neurotransmitter-gated ion-channel transmembrane region
32	PF10670.2	108	Nickel uptake substrate-specific transmembrane region
33	PF01389.10	13	OmpA-like transmembrane domain
34	PF09777.2	10	Osteopetrosis-associated transmembrane protein 1 precursor
35	PF09656.3	29	Putative transmembrane protein (PGPGW)
36	PF01127.15	126	Succinate dehydrogenase/Fumarate reductase transmembrane subunit
37	PF08693.3	13	Transmembrane alpha-helix domain
38	PF09049.3	3	Stannin transmembrane
39	PF09772.2	10	Transmembrane protein 26
40	PF03647.6	66	Transmembrane proteins 14C
41	PF10268.2	8	Predicted transmembrane protein 161AB
42	PF10190.2	8	Putative transmembrane protein 170
43	PF10269.2	15	Transmembrane Fragile-X-F protein
44	PF09771.2	5	Transmembrane protein 188
45	PF09788.2	8	Transmembrane protein 55A
46	PF10267.2	15	Predicted transmembrane and coiled-coil 2 protein
47	PF11044.1	4	Plectrovirus spv1-c74 ORF 12 transmembrane protein
48	PF10271.2	11	Putative transmembrane protein
49	PF10272.2	10	Putative transmembrane protein precursor
50	PF09534.3	24	Tryptophan-associated transmembrane protein (Trp_oprn_chp)
51	PF02921.7	32	Ubiquinol cytochrome reductase transmembrane region

Supplementary Table 4: Results of KalignP at each fold of cross-validation on Balibase 3.0. SP and TC scores were calculated by the program bali_score version 3.0 and the core blocks defined in the xml file were used. The three subsets of the cross-validation were created by randomly splitting the alignments in each reference set into three equal parts. The parameters were optimized in the training set by maximizing the average of the sum of SP and TC scores at each fold. Since the optimized parameters $P1$ to $P4$ are the same within each fold, they are only shown in the line for record "All". Note that the values of SP and TC scores shown in table have been multiplied by 100 for easy reading.

		No. of alignments		Bali-scores				Parameters			
		Train	Test	Train		Test		P1[a]	P2[b]	P3[c]	P4[d]
				SP	TC	SP	TC				
Fold 1	RV11	51	25	63.9	36.9	69.9	52.7				
	RV12	59	29	92.2	81.0	90.6	77.1				
	RV20	55	27	91.7	39.3	93.5	43.1				
	RV30	40	20	84.9	52.6	79.2	45.9				
	RV40	33	16	89.2	54.3	90.4	48.1				
	RV50	21	10	84.4	46.8	81.9	38.9				
	All	259	127	84.4	52.9	84.7	53.5	0.50	0.52	0.50	0.50
Fold 2	RV11	51	25	65.8	43.8	64.6	37.6				
	RV12	59	29	92.1	80.6	91.3	79.0				
	RV20	55	27	92.2	42.4	92.9	36.7				
	RV30	40	20	83.9	54.1	83.2	47.9				
	RV40	33	16	90.3	52.5	89.1	51.0				
	RV50	21	10	85.1	46.5	82.4	41.1				
	All	259	127	84.9	54.8	84.1	50.5	0.50	0.51	0.60	0.45
Fold 3	RV11	50	26	67.2	45.1	61.8	35.3				
	RV12	58	30	91.2	78.7	93.1	82.9				
	RV20	54	28	93.2	40.4	90.7	39.2				
	RV30	40	20	82.6	50.5	86.2	55.9				
	RV40	32	17	90.2	49.5	89.5	57.5				
	RV50	20	11	82.9	40.7	86.8	52.0				
	All	254	132	84.8	52.8	84.4	54.3	0.50	0.52	0.60	0.50
Average parameters								0.50	0.52	0.57	0.48

[a] $shift_C$ [b] $shift_{HE}$ [c] $weight_C$ [d] $weight_{HE}$

References

Edgar, R. C. (2004). MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, **32**(5), 1792–1797.

Finn, R. D., Mistry, J., Tate, J., Coggill, P., Heger, A., Pollington, J. E., Gavin, O. L., Gunasekaran, P., Ceric, G., Forslund, K., Holm, L., Sonnhammer, E. L. L., Eddy, S. R., and Bateman, A. (2009). The pfam protein families database. *Nucleic Acids Research*, **38**(Database), D211–D222.

Jones, D. T. (1999). Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, **292**(2), 195–202.

Lassmann, T., Frings, O., and Sonnhammer, E. L. L. (2009). Kalign2: high-performance multiple alignment of protein and nucleotide sequences allowing external features. *Nucleic Acids Research*, **37**(3), 858 –865.

Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994). CLUSTAL w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, **22**(22), 4673–4680.

Thompson, J. D., Koehl, P., Ripp, R., and Poch, O. (2005). BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark. *Proteins*, **61**(1), 127–136.

Zhou, T., Shu, N., and Hovmller, S. (2010). A novel method for accurate one-dimensional protein structure prediction based on fragment matching. *Bioinformatics (Oxford, England)*, **26**(4), 470–477.