HW4: Pairwise Alignment Solutions

Author: Your Name

Last update: 22 April, 2022

Source code downloads: [.Rmd] [pairwiseAlign_Fct.R]

Rendering Instructions

To render this R Markdown document, one needs to download the following files to the same directory.

- HW4_key.Rmd: Rmd source file for this document
- pairwiseAlign_Fct.R: R script defining pairwise alignment functions
- bibtex.bib: references cited in the text in BibTeX format

Next, one can render the report to HTML, PDF and other formats following the instructions below. Both the HTML and PDF versions are linked here:

- HTML: this report in HTML format
- PDF: corresponding PDF version

The HTML report can be rendered with `rmarkdown::render()` as follows.

```
rmarkdown::render('HW4_key.Rmd') # From R
Rscript -e "rmarkdown::render('HW4_key.Rmd')" # From command-line
```

To render a PDF file instead of HTML, one can instruct the rendering function to do so like this: `rmarkdown::render('HW4_key.Rmd', c('pdf_document'))`. To render to several formats with a single command, one can concatenate the formatting values with `c('html_document', 'pdf_document')`.

A. Choice of Sequence Type

Task 1: Which sequence type - amino acid or nucleotide - is more appropriate to search databases for remotely related sequences? Provide at least three reasons for your decision.

Answer: When coding sequences are expected to have weak similarities then one should use protein sequences rather than DNA sequences for database searching, because of (1) their higher information content (20 versus 4 letter alphabet), as well as (2) the better scoring and (3) functional classification systems available for amino acids.

B. Dynamic Programming for Pairwise Alignments

Task 2: Create manually (or write an R script for it) one global and one local alignment for the following two protein sequences using the Needleman-Wusch and Smith-Waterman algorithms, respectively (Smith and Waterman 1981; Needleman and Wunsch 1970).

```
O15528: PFGFGKRSCMGRRLA
P98187: FIPFSAGPRNCIGQK
```

Source functions

All alignment functions used in the following sections are defined in the downloaded R script file that is named `pairwiseAlign_Fct.R`. These functions are loaded with the `source()` command below.

```
source("pairwiseAlign_Fct.R")
```

Input sequences

Define within R or import them (here former).

```
S1 <- "PFGFGKRSCMGRRLA"
S2 <- "FIPFSAGPRNCIGQK"
```

Additional test sequences

```
# S1 <- "HEAGAWGHEE"
# S2 <- "PAWHEAE"
```

Global alignment

The alignment type choice is passed on to all following functions.

```
align_type <- "global"
# align_type <- "local"
```

Dynamic programming matrices

```
dynMA <- dynProgMatrix(S1, S2, align_method=align_type, gap_penalty=8, substitutionMA="BLOSUM50")
```

The matrices are stored in a list and returned below. The path is indicated by three numbers in the `glob_ma_path` matrix. Their meaning is:

- 1: diagonal
- 2: vertical (up)
- 3: horizontal (left)

```
dynMA
```

```
## $glob_ma
##      gp    P   F   G   F   G   K   R   S   C   M   G   R    R    L    A
## gp    0   -8 -16 -24 -32 -40 -48 -56 -64 -72 -80 -88 -96 -104 -112 -120
## F    -8   -4   0  -8 -16 -24 -32 -40 -48 -56 -64 -72 -80  -88  -96 -104
## I   -16  -11  -4  -4  -8 -16 -24 -32 -40 -48 -54 -62 -70  -78  -86  -94
## P   -24   -6 -12  -6  -8 -10 -17 -25 -33 -41 -49 -56 -64  -72  -80  -87
## F   -32  -14   2  -6   2  -6 -14 -20 -28 -35 -41 -49 -57  -65  -71  -79
## S   -40  -22  -6   2  -6   2  -6 -14 -15 -23 -31 -39 -47  -55  -63  -70
## A   -48  -30 -14  -6  -1  -6   1  -7 -13 -16 -24 -31 -39  -47  -55  -58
## G   -56  -38 -22  -6  -9   7  -1  -2  -7 -15 -19 -16 -24  -32  -40  -48
## P   -64  -46 -30 -14 -10  -1   6  -2  -3 -11 -18 -21 -19  -27  -35  -41
## R   -72  -54 -38 -22 -17  -9   2  13   5  -3 -11 -19 -14  -12  -20  -28
## N   -80  -62 -46 -30 -25 -17  -6   5  14   6  -2 -10 -18  -15  -16  -21
## C   -88  -70 -54 -38 -32 -25 -14  -3   6  27  19  11   3   -5  -13  -17
## I   -96  -78 -62 -46 -38 -33 -22 -11  -2  19  29  21  13    5   -3  -11
## G  -104  -86 -70 -54 -46 -30 -30 -19 -10  11  21  37  29   21   13    5
## Q  -112  -94 -78 -62 -54 -38 -28 -27 -18   3  13  29  38   30   22   14
## K  -120 -102 -86 -70 -62 -46 -32 -25 -26  -5   5  21  32   41   33   25
##
```

```
## $glob_ma_path
##    gp P F G F G K R S C M G R R L A
## gp  0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## F   2 1 1 3 1 3 3 3 3 3 3 3 3 3 3 3
## I   2 1 1 1 1 3 3 3 3 3 1 3 3 3 1 3
## P   2 1 2 1 1 1 1 3 1 3 3 1 3 3 3 1
## F   2 2 1 3 1 3 1 1 1 1 1 3 3 3 1 3
## S   2 2 2 1 2 1 1 3 1 3 3 3 3 3 3 1
## A   2 2 2 1 1 1 1 3 1 1 1 1 3 3 3 1
## G   2 2 2 1 2 1 3 1 1 3 1 1 3 3 3 3
## P   2 1 2 2 1 2 1 3 1 1 1 1 1 1 3 1
## R   2 2 2 2 1 2 1 1 3 3 3 3 1 1 3 3
## N   2 2 2 2 2 1 2 2 1 3 3 3 3 1 1 1
## C   2 2 2 2 1 2 2 2 2 1 3 3 3 3 3 1
## I   2 2 2 2 1 2 2 2 2 2 1 3 3 3 1 3
## G   2 2 2 1 2 1 2 2 2 2 2 1 3 3 3 3
## Q   2 2 2 2 2 2 1 2 2 2 2 2 1 1 3 3
## K   2 2 2 2 2 2 1 1 2 2 2 2 1 1 3 3
```

Compute alignment

The following `alignList` stores all relevant results in a list, including dynamic programming matrices, as well as the coordinates (named `path_coor`) to highlight path in dynamic progamming matrix (see below).

```
alignList <- alignmentTraceback(ma=dynMA[[1]], ma_path=dynMA[[2]], align_method=align_type)
names(alignList)
```

```
## [1] "ma"        "ma_path"   "path_coor" "as1"       "consensus" "as2"       "score"
```

```
# alignList$ma # dyn ma with scores
# alignList$ma_path # dyn ma with path
# alignList$path_coor # coordinates for path to auto highlight path in HTML/PDF table
```

Return results

Traceback in matrix

The following prints the fully populated dynamic programming matrix where the traceback path is highlighted in color.

```
printColMa(alignList)
```

Alignment and score

```
printAlign(x=alignList)
```

```
##
##  S1:  --PFGFGKRSCMGRRLA
##         ||  | | | |
##  S2:  FIPFSAGPRNCIGQK--
##
##  Score of alignment: 25
```

Local alignment

The alignment type choice is passed on to all following functions.

	gp	P	F	G	F	G	K	R	S	C	M	G	R	R	L	A
gp	0	-8	-16	-24	-32	-40	-48	-56	-64	-72	-80	-88	-96	-104	-112	-120
F	-8	-4	0	-8	-16	-24	-32	-40	-48	-56	-64	-72	-80	-88	-96	-104
I	-16	-11	-4	-4	-8	-16	-24	-32	-40	-48	-54	-62	-70	-78	-86	-94
P	-24	-6	-12	-6	-8	-10	-17	-25	-33	-41	-49	-56	-64	-72	-80	-87
F	-32	-14	2	-6	2	-6	-14	-20	-28	-35	-41	-49	-57	-65	-71	-79
S	-40	-22	-6	2	-6	2	-6	-14	-15	-23	-31	-39	-47	-55	-63	-70
A	-48	-30	-14	-6	-1	-6	1	-7	-13	-16	-24	-31	-39	-47	-55	-58
G	-56	-38	-22	-6	-9	7	-1	-2	-7	-15	-19	-16	-24	-32	-40	-48
P	-64	-46	-30	-14	-10	-1	6	-2	-3	-11	-18	-21	-19	-27	-35	-41
R	-72	-54	-38	-22	-17	-9	2	13	5	-3	-11	-19	-14	-12	-20	-28
N	-80	-62	-46	-30	-25	-17	-6	5	14	6	-2	-10	-18	-15	-16	-21
C	-88	-70	-54	-38	-32	-25	-14	-3	6	27	19	11	3	-5	-13	-17
I	-96	-78	-62	-46	-38	-33	-22	-11	-2	19	29	21	13	5	-3	-11
G	-104	-86	-70	-54	-46	-30	-30	-19	-10	11	21	37	29	21	13	5
Q	-112	-94	-78	-62	-54	-38	-28	-27	-18	3	13	29	38	30	22	14
K	-120	-102	-86	-70	-62	-46	-32	-25	-26	-5	5	21	32	41	33	25

```
# align_type <- "global"
align_type <- "local"
```

Dynamic programming matrices

```
dynMA <- dynProgMatrix(S1, S2, align_method=align_type, gap_penalty=8, substitutionMA="BLOSUM50")
```

The matrices are stored in a list and returned below.

```
dynMA
```

```
## $loc_ma
##    gp  P  F  G  F  G  K  R  S  C  M  G  R  R  L  A
## gp  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## F   0  0  8  0  8  0  0  0  0  0  0  0  0  0  1  0
## I   0  0  0  4  0  4  0  0  0  0  2  0  0  0  2  0
## P   0 10  2  0  0  0  3  0  0  0  0  0  0  0  0  1
## F   0  2 18 10  8  0  0  0  0  0  0  0  0  0  1  0
## S   0  0 10 18 10  8  0  0  5  0  0  0  0  0  0  2
## A   0  0  2 10 15 10  7  0  1  4  0  0  0  0  0  5
## G   0  0  0 10  7 23 15  7  0  0  1  8  0  0  0  0
## P   0 10  2  2  6 15 22 14  6  0  0  0  5  0  0  0
## R   0  2  7  0  0  7 18 29 21 13  5  0  7 12  4  0
## N   0  0  0  7  0  0 10 21 30 22 14  6  0  6  8  3
## C   0  0  0  0  5  0  2 13 22 43 35 27 19 11  4  7
## I   0  0  0  0  0  1  0  5 14 35 45 37 29 21 13  5
## G   0  0  0  8  0  8  0  0  6 27 37 53 45 37 29 21
## Q   0  0  0  0  4  0 10  2  0 19 29 45 54 46 38 30
## K   0  0  0  0  0  2  6 13  5 11 21 37 48 57 49 41
##
## $loc_ma_path
##    gp P F G F G K R S C M G R R L A
## gp  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## F   0 1 1 3 1 3 1 1 1 1 1 1 1 1 1 1
## I   0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
## P   0 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1
## F   0 2 1 3 1 3 1 1 1 1 1 1 1 1 1 1
## S   0 1 2 1 3 1 1 1 1 1 1 1 1 1 1 1
## A   0 1 2 1 1 1 1 3 1 1 1 1 1 1 1 1
## G   0 1 1 1 2 1 3 3 1 1 1 1 3 1 1 1
## P   0 1 3 2 1 2 1 3 1 3 1 2 1 1 1 1
## R   0 2 1 1 1 2 1 1 3 3 3 1 1 1 3 1
## N   0 1 2 1 3 1 2 2 1 3 3 3 1 1 1 1
## C   0 1 1 2 1 1 2 2 2 1 3 3 3 3 1 1
## I   0 1 1 1 1 1 1 2 2 2 1 3 3 3 1 3
## G   0 1 1 1 3 1 3 1 2 2 2 1 3 3 3 3
## Q   0 1 1 2 1 2 1 3 1 2 2 2 1 1 3 3
## K   0 1 1 1 1 1 1 1 1 3 2 2 2 1 1 3 3
```

Compute alignment

Note: `alignList` stores all relevant results in a list, including dynamic programming matrices, as well as the coordinates (named `path_coor`) to highlight the path in the dynamic progamming matrix. This way one can easily generate a single dynamic programming matrix with the traceback path highlighted by colors or arrows in an HTML or PDF document (see below).

```
alignList <- alignmentTraceback(ma=dynMA[[1]], ma_path=dynMA[[2]], align_method=align_type)
names(alignList)
```

```
## [1] "ma"        "ma_path"   "path_coor" "as1"       "consensus" "as2"       "score"
```

```
# alignList$ma # dyn ma with scores
# alignList$ma_path # dyn ma with path
# alignList$path_coor # coordinates for path to auto highlight path in HTML/PDF table
```

Return results

Traceback in matrix

The following prints the fully populated dynamic programming matrix where the traceback path is highlighted in color.

```
printColMa(alignList)
```

Alignment and score

```
printAlign(x=alignList)
```

```
##
##  S1:  PFGFGKRSCMGRR
##       ||  | | | |
##  S2:  PFSAGPRNCIGQK
##
##  Score of alignment: 57
```

C. Different Substitution Matrices

Task 1: Load the Biostrings package in R, import the following two cytochrome P450 sequences O15528 and P98187 from NCBI (save as myseq.fasta), and create a global alignment with the pairwiseAlignment function from Biostrings as follows.

	gp	P	F	G	F	G	K	R	S	C	M	G	R	R	L	A
gp	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	8	0	8	0	0	0	0	0	0	0	0	0	1	0
I	0	0	0	4	0	4	0	0	0	0	2	0	0	0	2	0
P	0	10	2	0	0	0	3	0	0	0	0	0	0	0	0	1
F	0	2	18	10	8	0	0	0	0	0	0	0	0	0	1	0
S	0	0	10	18	10	8	0	0	5	0	0	0	0	0	0	2
A	0	0	2	10	15	10	7	0	1	4	0	0	0	0	0	5
G	0	0	0	10	7	23	15	7	0	0	1	8	0	0	0	0
P	0	10	2	2	6	15	22	14	6	0	0	0	5	0	0	0
R	0	2	7	0	0	7	18	29	21	13	5	0	7	12	4	0
N	0	0	0	7	0	0	10	21	30	22	14	6	0	6	8	3
C	0	0	0	0	5	0	2	13	22	43	35	27	19	11	4	7
I	0	0	0	0	0	1	0	5	14	35	45	37	29	21	13	5
G	0	0	0	8	0	8	0	0	6	27	37	53	45	37	29	21
Q	0	0	0	0	4	0	10	2	0	19	29	45	54	46	38	30
K	0	0	0	0	0	2	6	13	5	11	21	37	48	57	49	41

```
library(Biostrings)
myseq <- readAAStringSet("myseq.fasta", "fasta")
(p <- pairwiseAlignment(myseq[[1]], myseq[[2]], type="global", substitutionMatrix="BLOSUM50"))
writePairwiseAlignments(p)
```

Your answers should address the following items:

Record the scores for the scoring matrices BLOSUM50, BLOSUM62 and BLOSUM80. How and why do the scores differ for the three scoring matrices?

Answer 1: The scores for the three BLOSUM substitutions matrices are:

- BLOSUM50: 227
- BLOSUM62: 54
- BLOSUM80: -52

Answer 2: Since the two sequences are relatively dissimilar (as determined by alignment view from `writePairwiseAlignments(p)`) it is expected that the BLOSUM matrices trained on more dissimilar sequences (*e.g.* BLOSUM50) result in higher scores than those trained on less similar sequences (*e.g.* BLOSUM80).

Session Info

```
sessionInfo()
```

```
## R version 4.1.3 (2022-03-10)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Debian GNU/Linux 10 (buster)
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.8.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.8.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8        LC_COLLATE=en_U
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATI
```

```
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] kableExtra_1.3.4    Biostrings_2.62.0   GenomeInfoDb_1.30.0 XVector_0.34.0      IRanges_2.28.0
##
## loaded via a namespace (and not attached):
##  [1] bslib_0.3.1           compiler_4.1.3        jquerylib_0.1.4       bitops_1.0-7          tool
## [10] evaluate_0.15         lifecycle_1.0.1       rlang_1.0.2           cli_3.1.0             rstu
## [19] stringr_1.4.0         httr_1.4.2            knitr_1.37            xml2_1.3.3            sys
## [28] R6_2.5.1              rmarkdown_2.13        magrittr_2.0.2        scales_1.1.1          html
## [37] munsell_0.5.0         crayon_1.4.2
```

References

Needleman, S B, and C D Wunsch. 1970. "A general method applicable to the search for similarities in the amino acid sequence of two proteins." *J. Mol. Biol.* 48 (3): 443–53. https://doi.org/10.1016/0022-2836(70)90057-4.

Smith, T F, and M S Waterman. 1981. "Identification of common molecular subsequences." *J. Mol. Biol.* 147 (1): 195–97. http://www.ncbi.nlm.nih.gov/pubmed/7265238.