Importing sequences from flat files

Charif, D. Lobry, J.R.

June 2, 2016

Contents

1 Importing raw sequence data from FASTA files

1.1 FASTA files examples

The FASTA format is very simple and widely used for simple import of biological sequences. It was used originally by the FASTA program [13]. It begins with a single-line description starting with a character `'>'`, followed by lines of sequence data of maximum 80 character each. Lines starting with a semi-colon character `';'` are comment lines. Examples of files in FASTA format are distributed with the `seqin`**R** package in the `sequences` directory:

```
list.files(path = system.file("sequences", package = "seqinr"), pattern = ".fasta")
```

```
 [1] "Anouk.fasta"        "bordetella.fasta"   "ct.fasta.gz"
 [4] "DarrenObbard.fasta" "ecolicgpe5.fasta"   "gopher.fasta"
 [7] "humanMito.fasta"    "legacy.fasta"       "louse.fasta"
[10] "malM.fasta"         "ortho.fasta"        "seqAA.fasta"
[13] "smallAA.fasta"      "smallAA.fasta.gz"
```

Here is an example of a FASTA file:

```
cat(readLines(system.file("sequences/seqAA.fasta", package = "seqinr")), sep = "\n")
```

```
>A06852                  183 residues
MPRLFSYLLGVWLLLSQLPREIPGQSTNDFIKACGRELVRLWVEICGSVSWGRTALSLEE
PQLETGPPAETMPSSITKDAEILKMMLEFVPNLPQELKATLSERQPSLRELQQSASKDSN
LNFEEFKKIILNRQNEAEDKSLLELKNLGLDKHSRKKRLFRMTLSEKCCQVGCIRKDIAR
LC*
```

Here is an example of a FASTA file with comment lines:

```
cat(readLines(system.file("sequences/legacy.fasta", package = "seqinr")), sep = "\n")
```

```
>LEGACY  921 bp
;
; Example of a FASTA file using comment lines starting with a semicolon
; as allowed in the original FASTA program:
;
;     if (line[0]!='>'&& line[0]!=';') {
;      for (i=l_offset; (n<maxs && rn < sstop)&&
;             ((ic=qascii[line[i]&AAMASK])<EL); i++)
;        if (ic<NA && ++rn > sstart) seq[n++]= ic;
;      if (ic == ES || rn > sstop) break;
;    }
;
; From file getseq.c in FASTA program version 35.2.5
;
ATGAAAATGAATAAAAGTCTCATCGTCCTCTGTTTATCAGCAGGGTTACTGGCAAGCGCG
CCTGGAATTAGCCTTGCCGATGTTAACTACGTACCGCAAAACACCAGCGACGCGCCAGCC
ATTCCATCTGCTGCGCTGCAACAACTCACCTGGACACCGGTCGATCAATCTAAAACCCAG
ACCACCCAACTGGCGACCGGCGGCCAACAACTGAACGTTCCCGGCATCAGTGGTCCGGTT
GCTGCGTACAGCGTCCCGGCAAACATTGGCGAACTGACCCTGACGCTGACCAGCGAAGTG
AACAAACAAACCAGCGTTTTTGCGCCGAACGTGCTGATTCTTGATCAGAACATGACCCCA
TCAGCCTTCTTCCCCAGCAGTTATTTCACCTACCAGGAACCAGGCGTGATGAGTGCAGAT
CGGCTGGAAGGCGTTATGCGCCTGACACCGGCGTTGGGGCAGCAAAAACTTTATGTTCTG
GTCTTTACCACGGAAAAAGATCTCCAGCAGACGACCCAACTGCTCGACCCGGCTAAAGCC
TATGCCAAGGGCGTCGGTAACTCGATCCCGGATATCCCCGATCCGGTTGCTCGTCATACC
ACCGATGGCTTACTGAAACTGAAAGTGAAAACGAACTCCAGCTCCAGCGTGTTGGTAGGA
CCCTTATTTGGTTCCTCCGCTCCAGCTCCGGTTACGGTAGGTAACACGGCGGCACCAGCT
GTGGCTGCACCCGCTCCGGCACCGGTGAAGAAAAGCGAGCCGATGCTCAACGACACGGAA
AGTTATTTTAATACCGCGATCAAAAACGCTGTCGCGAAAGGTGATGTTGATAAGGCGTTA
AAACTGCTTGATGAAGCTGAACGCTTGGGATCGACATCTGCCCGTTCCACCTTTATCAGC
AGTGTAAAAGGCAAGGGGTAA
```

1.2 The function `read.fasta()`

The function `read.fasta()` imports sequences from FASTA files into your workspace.

1.2.1 DNA file example

The example file looks like:

```
dnafile <- system.file("sequences/malM.fasta", package = "seqinr")
cat(readLines(dnafile), sep = "\n")
```

```
>XYLEECOM.MALM  921 bp ACCESSION  E00218, X04477
ATGAAAATGAATAAAAGTCTCATCGTCCTCTGTTTATCAGCAGGGTTACTGGCAAGCGCG
CCTGGAATTAGCCTTGCCGATGTTAACTACGTACCGCAAAACACCAGCGACGCGCCAGCC
ATTCCATCTGCTGCGCTGCAACAACTCACCTGGACACCGGTCGATCAATCTAAAACCCAG
ACCACCCAACTGGCGACCGGCGGCCAACAACTGAACGTTCCCGGCATCAGTGGTCCGGTT
GCTGCGTACAGCGTCCCGGCAAACATTGGCGAACTGACCCTGACGCTGACCAGCGAAGTG
AACAAACAAACCAGCGTTTTTGCGCCGAACGTGCTGATTCTTGATCAGAACATGACCCCA
TCAGCCTTCTTCCCCAGCAGTTATTTCACCTACCAGGAACCAGGCGTGATGAGTGCAGAT
CGGCTGGAAGGCGTTATGCGCCTGACACCGGCGTTGGGGCAGCAAAAACTTTATGTTCTG
GTCTTTACCACGGAAAAAGATCTCCAGCAGACGACCCAACTGCTCGACCCGGCTAAAGCC
TATGCCAAGGGCGTCGGTAACTCGATCCCGGATATCCCCGATCCGGTTGCTCGTCATACC
ACCGATGGCTTACTGAAACTGAAAGTGAAAACGAACTCCAGCTCCAGCGTGTTGGTAGGA
CCCTTATTTGGTTCCTCCGCTCCAGCTCCGGTTACGGTAGGTAACACGGCGGCACCAGCT
GTGGCTGCACCCGCTCCGGCACCGGTGAAGAAAAGCGAGCCGATGCTCAACGACACGGAA
AGTTATTTTAATACCGCGATCAAAAACGCTGTCGCGAAAGGTGATGTTGATAAGGCGTTA
AAACTGCTTGATGAAGCTGAACGCTTGGGATCGACATCTGCCCGTTCCACCTTTATCAGC
AGTGTAAAAGGCAAGGGGTAA
```

With default arguments the output looks like:

```
read.fasta(file = dnafile)
$XYLEECOM.MALM
  [1] "a" "t" "g" "a" "a" "a" "a" "t" "g" "a" "a" "t" "a" "a" "a" "a" "g" "t"
 [19] "c" "t" "c" "a" "t" "c" "g" "t" "c" "c" "t" "c" "t" "g" "t" "t" "t" "a"
 [37] "t" "c" "a" "g" "c" "a" "g" "g" "g" "t" "t" "a" "c" "t" "g" "g" "c" "a"
 [55] "a" "g" "c" "g" "c" "g" "c" "c" "t" "g" "g" "a" "a" "t" "t" "a" "g" "c"
 [73] "c" "t" "t" "g" "c" "c" "g" "a" "t" "g" "t" "t" "a" "a" "c" "t" "a" "c"
 [91] "g" "t" "a" "c" "c" "g" "c" "a" "a" "a" "a" "c" "a" "c" "c" "a" "g" "c"
[109] "g" "a" "c" "g" "c" "g" "c" "c" "a" "g" "c" "c" "a" "t" "t" "c" "c" "a"
[127] "t" "c" "t" "g" "c" "t" "g" "c" "g" "c" "t" "g" "c" "a" "a" "c" "a" "a"
[145] "c" "t" "c" "a" "c" "c" "t" "g" "g" "a" "c" "a" "c" "c" "g" "g" "t" "c"
[163] "g" "a" "t" "c" "a" "a" "t" "c" "t" "a" "a" "a" "a" "c" "c" "c" "a" "g"
[181] "a" "c" "c" "a" "c" "c" "c" "a" "a" "c" "t" "g" "g" "c" "g" "a" "c" "c"
[199] "g" "g" "c" "g" "g" "c" "c" "a" "a" "c" "a" "a" "c" "t" "g" "a" "a" "c"
[217] "g" "t" "t" "c" "c" "c" "g" "g" "c" "a" "t" "c" "a" "g" "t" "g" "g" "t"
[235] "c" "c" "g" "g" "t" "t" "g" "c" "t" "g" "c" "g" "t" "a" "c" "a" "g" "c"
[253] "g" "t" "c" "c" "c" "g" "g" "c" "a" "a" "a" "c" "a" "t" "t" "g" "g" "c"
[271] "g" "a" "a" "c" "t" "g" "a" "c" "c" "c" "t" "g" "a" "c" "g" "c" "t" "g"
[289] "a" "c" "c" "a" "g" "c" "g" "a" "a" "g" "t" "g" "a" "a" "c" "a" "a" "a"
[307] "c" "a" "a" "a" "c" "c" "a" "g" "c" "g" "t" "t" "t" "t" "t" "g" "c" "g"
[325] "c" "c" "g" "a" "a" "c" "g" "t" "g" "c" "t" "g" "a" "t" "t" "c" "t" "t"
[343] "g" "a" "t" "c" "a" "g" "a" "a" "c" "a" "t" "g" "a" "c" "c" "c" "c" "a"
[361] "t" "c" "a" "g" "c" "c" "t" "t" "c" "t" "t" "c" "c" "c" "c" "a" "g" "c"
[379] "a" "g" "t" "t" "a" "t" "t" "t" "c" "a" "c" "c" "t" "a" "c" "c" "a" "g"
[397] "g" "a" "a" "c" "c" "a" "g" "g" "c" "g" "t" "g" "a" "t" "g" "a" "g" "t"
[415] "g" "c" "a" "g" "a" "t" "c" "g" "g" "c" "t" "g" "g" "a" "a" "g" "g" "c"
[433] "g" "t" "t" "a" "t" "g" "c" "g" "c" "c" "t" "g" "a" "c" "a" "c" "c" "g"
[451] "g" "c" "g" "t" "t" "g" "g" "g" "g" "c" "a" "g" "c" "a" "a" "a" "a" "a"
[469] "c" "t" "t" "t" "a" "t" "g" "t" "t" "c" "t" "g" "g" "t" "c" "t" "t" "t"
[487] "a" "c" "c" "a" "c" "g" "g" "a" "a" "a" "a" "a" "g" "a" "t" "c" "t" "c"
[505] "c" "a" "g" "c" "a" "g" "a" "c" "g" "a" "c" "c" "c" "a" "a" "c" "t" "g"
[523] "c" "t" "c" "g" "a" "c" "c" "c" "g" "g" "c" "t" "a" "a" "a" "g" "c" "c"
[541] "t" "a" "t" "g" "c" "c" "a" "a" "g" "g" "g" "c" "g" "t" "c" "g" "g" "t"
[559] "a" "a" "c" "t" "c" "g" "a" "t" "c" "c" "c" "g" "g" "a" "t" "a" "t" "c"
[577] "c" "c" "c" "g" "a" "t" "c" "c" "g" "g" "t" "t" "g" "c" "t" "c" "g" "t"
[595] "c" "a" "t" "a" "c" "c" "a" "c" "c" "g" "a" "t" "g" "g" "c" "t" "t" "a"
[613] "c" "t" "g" "a" "a" "a" "c" "t" "g" "a" "a" "a" "g" "t" "g" "a" "a" "a"
[631] "a" "c" "g" "a" "a" "c" "t" "c" "c" "a" "g" "c" "t" "c" "c" "a" "g" "c"
[649] "g" "t" "g" "t" "t" "g" "g" "t" "a" "g" "g" "a" "c" "c" "c" "t" "t" "a"
[667] "t" "t" "t" "g" "g" "t" "t" "c" "c" "t" "c" "c" "g" "c" "t" "c" "c" "a"
[685] "g" "c" "t" "c" "c" "g" "g" "t" "t" "a" "c" "g" "g" "t" "a" "g" "g" "t"
[703] "a" "a" "c" "a" "c" "g" "g" "c" "g" "g" "c" "a" "c" "c" "a" "g" "c" "t"
[721] "g" "t" "g" "g" "c" "t" "g" "c" "a" "c" "c" "c" "g" "c" "t" "c" "c" "g"
[739] "g" "c" "a" "c" "c" "g" "g" "t" "g" "a" "a" "g" "a" "a" "a" "a" "g" "c"
[757] "g" "a" "g" "c" "c" "g" "a" "t" "g" "c" "t" "c" "a" "a" "c" "g" "a" "c"
[775] "a" "c" "g" "g" "a" "a" "a" "g" "t" "t" "a" "t" "t" "t" "t" "a" "a" "t"
[793] "a" "c" "c" "g" "c" "g" "a" "t" "c" "a" "a" "a" "a" "a" "c" "g" "c" "t"
```

```
[811] "g" "t" "c" "g" "c" "g" "a" "a" "a" "g" "g" "t" "g" "a" "t" "g" "t" "t"
[829] "g" "a" "t" "a" "a" "g" "g" "c" "g" "t" "t" "a" "a" "a" "a" "c" "t" "g"
[847] "c" "t" "t" "g" "a" "t" "g" "a" "a" "g" "c" "t" "g" "a" "a" "c" "g" "c"
[865] "t" "t" "g" "g" "g" "a" "t" "c" "g" "a" "c" "a" "t" "c" "t" "g" "c" "c"
[883] "c" "g" "t" "t" "c" "c" "a" "c" "c" "t" "t" "t" "a" "t" "c" "a" "g" "c"
[901] "a" "g" "t" "g" "t" "a" "a" "a" "a" "g" "g" "c" "a" "a" "g" "g" "g" "g"
[919] "t" "a" "a"
attr(,"name")
[1] "XYLEECOM.MALM"
attr(,"Annot")
[1] ">XYLEECOM.MALM  921 bp ACCESSION  E00218, X04477"
attr(,"class")
[1] "SeqFastadna"
```

As from **seqinR** 1.0-5 the automatic conversion of sequences into vector of single characters can be neutralized, for instance:

```
read.fasta(file = dnafile, as.string = TRUE)
```

```
$XYLEECOM.MALM
[1] "atgaaaatgaataaaagtctcatcgtcctctgtttatcagcagggttactggcaagcgcgcctggaattagccttgccgatgttaactacgtaccgcaaaacaccagc
attr(,"name")
[1] "XYLEECOM.MALM"
attr(,"Annot")
[1] ">XYLEECOM.MALM  921 bp ACCESSION  E00218, X04477"
attr(,"class")
[1] "SeqFastadna"
```

Forcing to lower case letters can be disabled this way:

```
read.fasta(file = dnafile, as.string = TRUE, forceDNAtolower = FALSE)
```

```
$XYLEECOM.MALM
[1] "ATGAAAATGAATAAAAGTCTCATCGTCCTCTGTTTATCAGCAGGGTTACTGGCAAGCGCGCCTGGAATTAGCCTTGCCGATGTTAACTACGTACCGCAAAACACCAGC
attr(,"name")
[1] "XYLEECOM.MALM"
attr(,"Annot")
[1] ">XYLEECOM.MALM  921 bp ACCESSION  E00218, X04477"
attr(,"class")
[1] "SeqFastadna"
```

1.2.2 Protein file example

The example file looks like:

```
aafile <-  system.file("sequences/seqAA.fasta", package = "seqinr")
cat(readLines(aafile), sep = "\n")
```

```
>A06852                  183 residues
MPRLFSYLLGVWLLLSQLPREIPGQSTNDFIKACGRELVRLWVEICGSVSWGRTALSLEE
PQLETGPPAETMPSSITKDAEILKMMLEFVPNLPQELKATLSERQPSLRELQQSASKDSN
LNFEEFKKIILNRQNEAEDKSLLELKNLGLDKHSRKKRLFRMTLSEKCCQVGCIRKDIAR
LC*
```

Read the protein sequence file, looks like:

```
read.fasta(aafile, seqtype = "AA")
```

```
$A06852
  [1] "M" "P" "R" "L" "F" "S" "Y" "L" "L" "G" "V" "W" "L" "L" "L" "S" "Q" "L"
 [19] "P" "R" "E" "I" "P" "G" "Q" "S" "T" "N" "D" "F" "I" "K" "A" "C" "G" "R"
 [37] "E" "L" "V" "R" "L" "W" "V" "E" "I" "C" "G" "S" "V" "S" "W" "G" "R" "T"
 [55] "A" "L" "S" "L" "E" "E" "P" "Q" "L" "E" "T" "G" "P" "P" "A" "E" "T" "M"
 [73] "P" "S" "S" "I" "T" "K" "D" "A" "E" "I" "L" "K" "M" "M" "L" "E" "F" "V"
 [91] "P" "N" "L" "P" "Q" "E" "L" "K" "A" "T" "L" "S" "E" "R" "Q" "P" "S" "L"
[109] "R" "E" "L" "Q" "Q" "S" "A" "S" "K" "D" "S" "N" "L" "N" "F" "E" "E" "F"
[127] "K" "K" "I" "I" "L" "N" "R" "Q" "N" "E" "A" "E" "D" "K" "S" "L" "L" "E"
```

```
[145] "L" "K" "N" "L" "G" "L" "D" "K" "H" "S" "R" "K" "K" "R" "L" "F" "R" "M"
[163] "T" "L" "S" "E" "K" "C" "C" "Q" "V" "G" "C" "I" "R" "K" "D" "I" "A" "R"
[181] "L" "C" "*"
attr(,"name")
[1] "A06852"
attr(,"Annot")
[1] ">A06852                  183 residues"
attr(,"class")
[1] "SeqFastaAA"
```

The same, but as string and without attributes setting, looks like:

```
read.fasta(aafile, seqtype = "AA", as.string = TRUE, set.attributes = FALSE)
```

```
$A06852
[1] "MPRLFSYLLGVWLLLSQLPREIPGQSTNDFIKACGRELVRLWVEICGSVSWGRTALSLEEPQLETGPPAETMPSSITKDAEILKMMLEFVPNLPQELKATLSERQPSL
```

1.2.3 Compressed file example

The original file before compression looks like:

```
uncompressed <- system.file("sequences/smallAA.fasta", package = "seqinr")
cat(readLines(uncompressed), sep = "\n")
```

```
>smallAA    A very small AA file in FASTA format
SEQINRSEQINRSEQINRSEQINR*
```

The compressed file example is full of mojibakes because of its binary nature, but the `readLines()` is still able to read it correctly:

```
compressed <- system.file("sequences/smallAA.fasta.gz", package = "seqinr")
readChar(compressed, nchar = 1000, useBytes = TRUE)
```

```
[1] "\037\x8b\b\b\xd4\024PW"
```

```
cat(readLines(compressed), sep = "\n")
```

```
>smallAA    A very small AA file in FASTA format
SEQINRSEQINRSEQINRSEQINR*
```

We can therefore import the sequences directly from a gzipped file:

```
res1 <- read.fasta(uncompressed)
res2 <- read.fasta(compressed)
identical(res1, res2)
```

```
[1] TRUE
```

This automatic conversion works well for local files but is no more active when you read the data from an URL, for instance:

```
myurl <- "ftp://ftp.ncbi.nlm.nih.gov/refseq/release/plasmid/plasmid.1.rna.fna.gz"
try.res <- try(read.fasta(myurl))
try.res
```

```
[1] "Error in read.fasta(myurl) : no line starting with a > character found\n"
attr(,"class")
[1] "try-error"
attr(,"condition")
<simpleError in read.fasta(myurl): no line starting with a > character found>
```

A simple workthrough is to encapsulate this into `gzcon()` :

```
myseq <- read.fasta(gzcon(url(myurl)))
getName(myseq)
```

```
[1] "gi|470467018|ref|NR_074151.1|" "gi|444303868|ref|NR_074290.1|"
[3] "gi|452192228|ref|NR_075742.1|" "gi|451991842|ref|NR_075394.1|"
[5] "gi|451991838|ref|NR_075390.1|" "gi|444303919|ref|NR_074342.1|"
[7] "gi|470486111|ref|NR_076736.1|" "gi|470480648|ref|NR_076426.1|"
[9] "gi|470478007|ref|NR_076423.1|"
```

1.3 The function `write.fasta()`

This function writes sequences to a file in FASTA format. Read 3 coding sequences sequences from a FASTA file:

```
ortho <- read.fasta(file = system.file("sequences/ortho.fasta", package = "seqinr"))
length(ortho)
[1] 3
ortho[[1]][1:12]
 [1] "a" "t" "g" "g" "c" "t" "c" "a" "g" "c" "g" "g"
```

Select only third codon positions:

```
ortho3 <- lapply(ortho, function(x) x[seq(from = 3, to = length(x), by = 3)])
ortho3[[1]][1:4]
[1] "g" "t" "g" "g"
```

Write the modified sequences to a file:

```
tmpf <- tempfile()
write.fasta(sequences = ortho3, names = names(ortho3), nbchar = 80, file.out = tmpf)
```

Read them again from the same file and check that sequences are preserved:

```
ortho3bis <- read.fasta(tmpf, set.attributes = FALSE)
identical(ortho3bis, ortho3)
[1] TRUE
```

1.4 Big room examples

1.4.1 Oriloc example (*Chlamydia trachomatis* complete genome)

A more consequent example is given in the fasta file `ct.fasta.gz` which contains the complete genome of *Chlamydia trachomatis* that was used in [2]. You should be able to reproduce figure 1b from this paper (*cf.* screenshot in figure 1) with the following code:

```
out <- oriloc(seq.fasta = system.file("sequences/ct.fasta.gz", package ="seqinr"),
      g2.coord = system.file("sequences/ct.predict", package = "seqinr"),
     oldoriloc = TRUE)
plot(out$st, out$sk/1000, type="l", xlab = "Map position in Kb",
         ylab = "Cumulated composite skew in Kb",
         main = expression(italic(Chlamydia~~trachomatis)~~complete~~genome), las = 1)
abline(h = 0, lty = 2)
text(400, -4, "Terminus")
text(850, 9, "Origin")
```

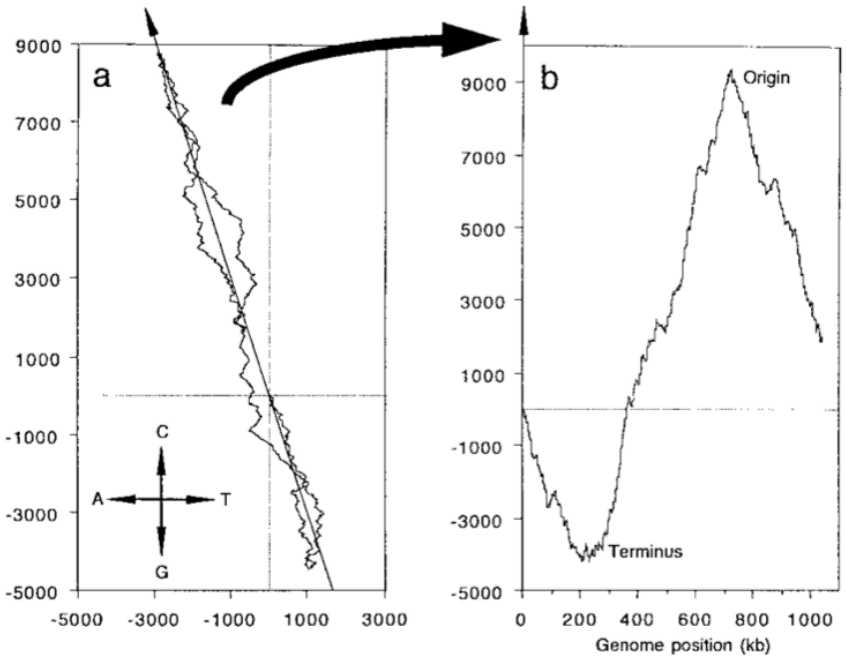
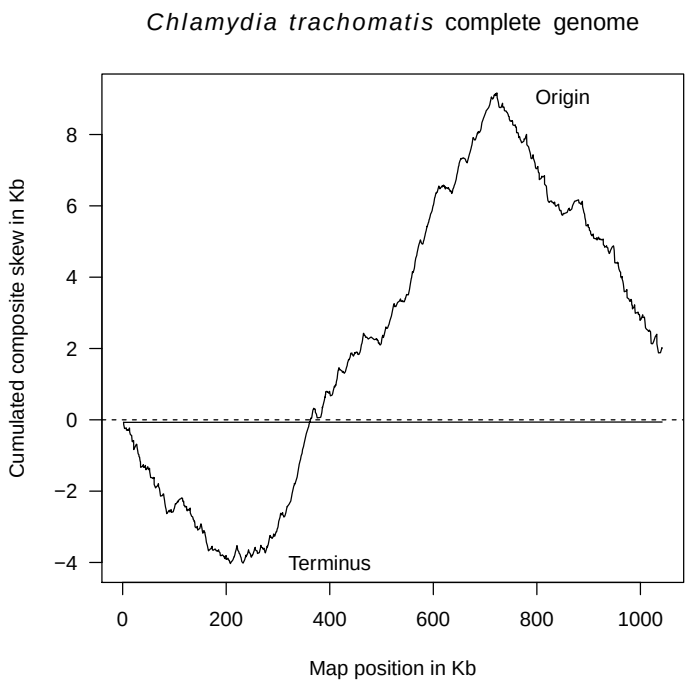
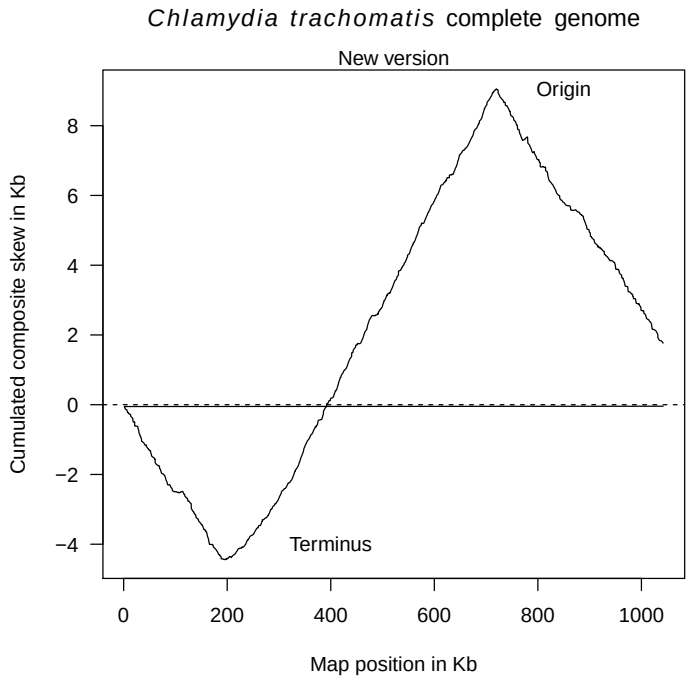


Figure 1: . Screenshot copy of figure 1 from [2]. The complete genome sequence of *Chlamydia trachomatis* (accession number: `AE001273`) was used to illustrate the method used by oriloc. (**a**) A DNA-walk is performed by reading the sequence in the third codon positions predicted by glimmer and walking into the plane according to the four directions defined by the four bases as indicated on the bottom left of the figure. The resulting DNA-walk is then summarized by projection onto the orthogonal regression line pointing out at about 11 o'clock in the figure. (**b**) The projected values are used as a composite skew index plotted versus map position on the chromosome. The origin is predicted at the maximum skew value while the terminus is predicted at the minimum.



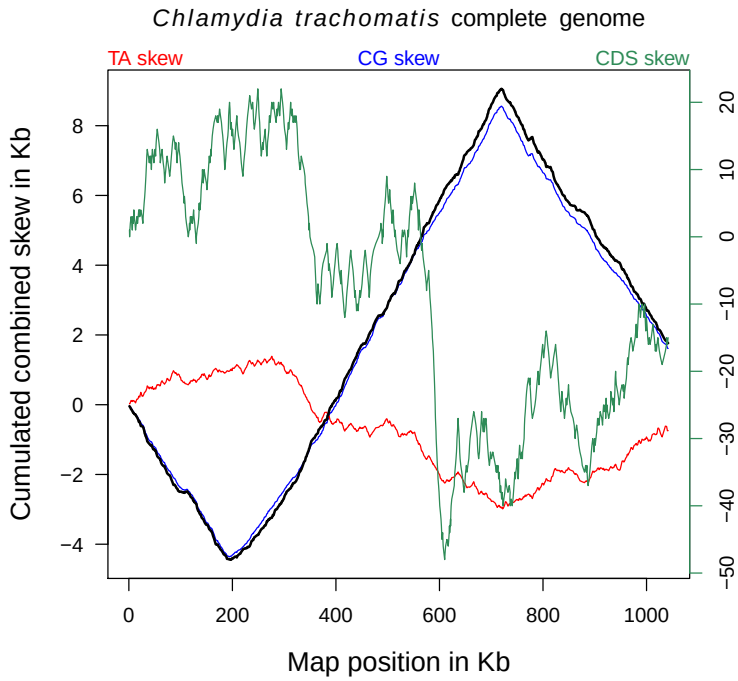
Note that the algorithm has been improved since then and that it's more advisable to use the default option `oldoriloc = FALSE` if you are interested in the prediction of origins and terminus of replication from base composition biases (more on this at `http://pbil.univ-lyon1.fr/software/oriloc.html`). See also [11] for a review on this topic. Here is the improved version:

```
out <- oriloc()
plot(out$st, out$sk/1000, type="l", xlab = "Map position in Kb",
         ylab = "Cumulated composite skew in Kb",
         main = expression(italic(Chlamydia~~trachomatis)~~complete~~genome), las = 1)
mtext("New version")
abline(h = 0, lty = 2)
text(400, -4, "Terminus")
text(850, 9, "Origin")
```



You can also call the `draw.oriloc()` function for the simultaneous representation of the CDS, AT and GC skew along with the combined skew of the previous plots:

```
draw.oriloc(out,
  main = expression(italic(Chlamydia~~trachomatis)~~complete~~genome),
  ta.mtext = "TA skew", ta.col = "red",
  cg.mtext = "CG skew", cg.col = "blue",
  cds.mtext = "CDS skew", cds.col = "seagreen",
  add.grid = FALSE)
```



1.4.2 Example with 21,161 proteins from *Arabidobpsis thaliana*

As from `seqin`**R** 1.0-5 the automatic conversion of sequences into vector of single characters and the automatic attribute settings can be neutralized, for instance :


Arabidobpsis thaliana.
Source: wikipedia.

```
smallAA <- system.file("sequences/smallAA.fasta", package = "seqinr")
read.fasta(smallAA, seqtype = "AA", as.string = TRUE, set.attributes = FALSE)
```

```
$smallAA
[1] "SEQINRSEQINRSEQINRSEQINR*"
```

This is interesting to save time and space when reading large FASTA files. Let's give a practical example. In their paper [5], Matthew Hannah, Arnd Heyer and Dirk Hincha were working on *Arabidobpsis thaliana* genes in order to detect those involved in cold acclimation. They were interested by the detection of proteins called hydrophilins, that had a mean hydrophilicity of over 1 and glycine content of over 0.08 [4], because they are though to be important for freezing tolerance. The starting point was a FASTA file called `ATH1_pep_cm_20040228` downloaded from the Arabidopsis Information Ressource (TAIR at `http://www.arabidopsis.org/`) which contains the sequences of 21,161 proteins.

```
athfile <- "ATH1_pep_cm_20040228.fasta"
download.file(paste("http://seqinr.r-forge.r-project.org", athfile, sep = "/"),
              athfile)
system.time(ath <- read.fasta(athfile, seqtype = "AA", as.string = TRUE,
                              set.attributes = FALSE))
```

```
   user  system elapsed
  3.827   0.036   3.863
```

It's about 10 seconds here to read 21,161 protein sequences. We save them in XDR binary format[1] to read them faster later at will:

```
save(ath, file = "ath.RData")
```

```
system.time(load("ath.RData"))
   user  system elapsed
  0.161   0.002   0.162
```

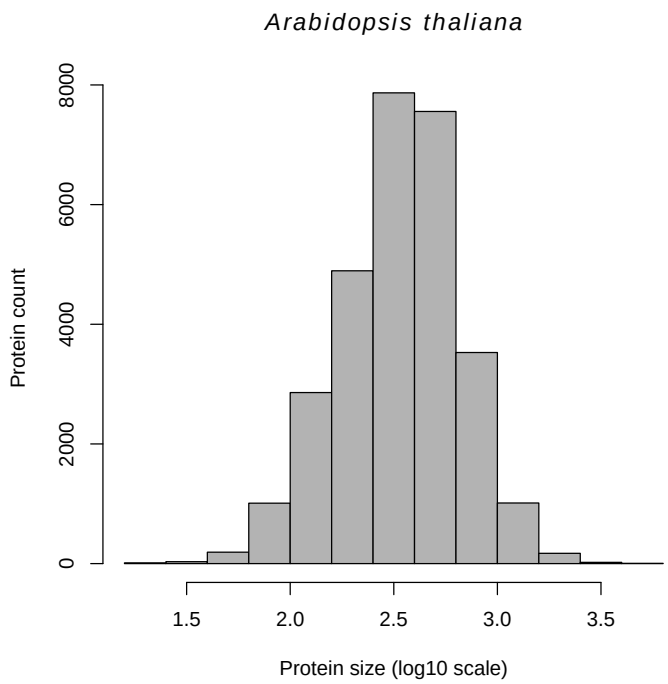
Now it's less than a second to load the whole data set thanks to the XDR format. The object size is about 15 Mo in RAM, that is something very close to the flat file size on disk:

```
object.size(ath)/2^20
16.2128143310547 bytes
file.info(athfile)$size/2^20
[1] 15.89863
```

Using strings for sequence storage is very comfortable when there is an efficient function to compute what you want. For instance, suppose that you are interested by the distribution of protein size in *Arabidopsis thaliana*. There is an efficient vectorized function called `nchar()` that will do the job, we just have to remove one unit because of the stop codon which is translated as a star (*) in this data set. This is a simple and direct task under R:

```
nres <- nchar(ath) - 1
hist(log10(nres), col = grey(0.7), xlab = "Protein size (log10 scale)",
ylab = "Protein count",
main = expression(italic(Arabidopsis~~thaliana)))
```

[1] this is a multi-platform compatible binary format: you can save data under unix and load them under Mac OS X, for instance, without problem.



However, sometimes it is more convenient to work with the single character vector representation of sequences. For instance, to count the number of glycine (G), we first play with one sequence, let's take the smallest one in the data set:

```
 which.min(nres)
At2g25990.1
       9523
 ath[[9523]]
[1] "MAGSQREKLKPRTKGSTRC*"
 s2c(ath[[9523]])
 [1] "M" "A" "G" "S" "Q" "R" "E" "K" "L" "K" "P" "R" "T" "K" "G" "S" "T" "R"
[19] "C" "*"
 s2c(ath[[9523]]) == "G"
 [1] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[13] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
 sum(s2c(ath[[9523]]) == "G")
[1] 2
```

We can now easily define a vectorised function to count the number of glycine:

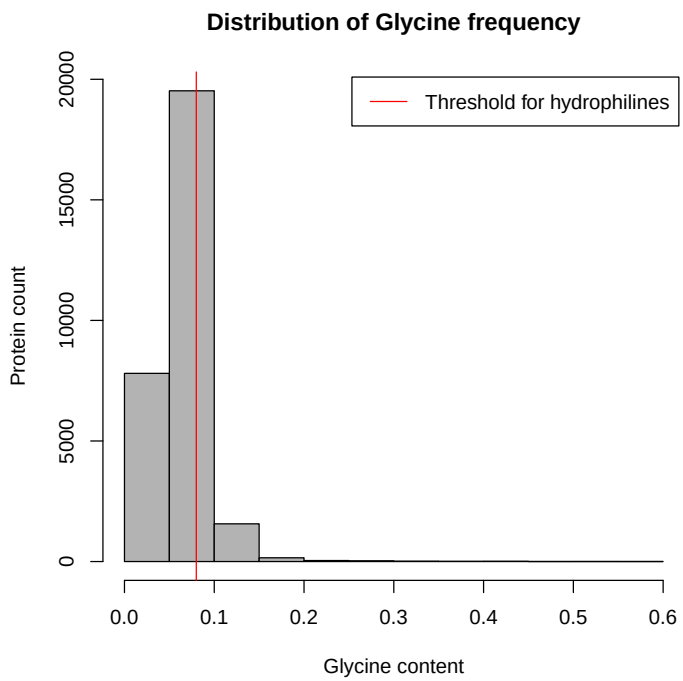
```
 ngly <- function(data){
  res <- sapply(data, function(x) sum(s2c(x) == "G"))
  names(res) <- NULL
  return(res)
 }
```

Now we can use `ngly()` in the same way that `nchar()` so that computing glycine frequencies is very simple:

```
ngly(ath[1:10])
[1]  25   5  29 128   8  27  27  26  21  18
fgly <- ngly(ath)/nres
```

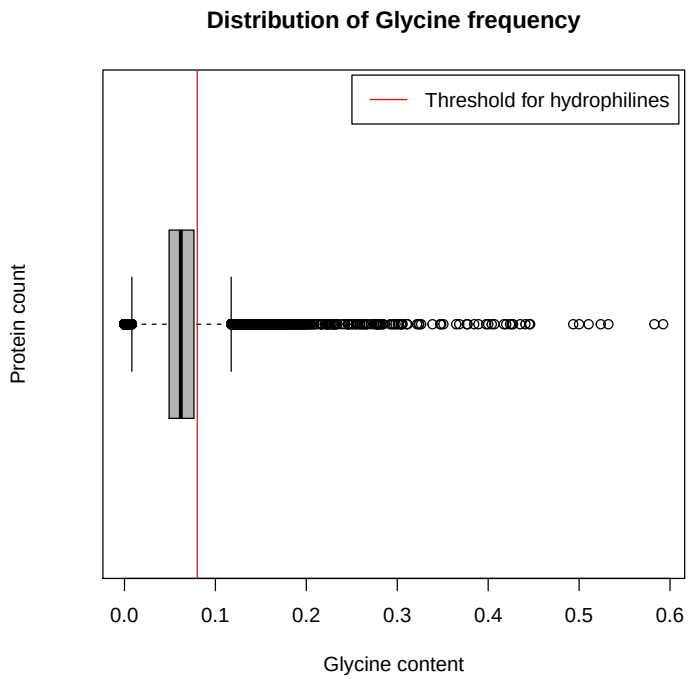
And we can have a look at the distribution:

```
hist(fgly, col = grey(0.7), main = "Distribution of Glycine frequency",
xlab = "Glycine content", ylab = "Protein count")
abline(v = 0.08, col = "red")
legend("topright",inset=0.01,lty=1,col="red",legend="Threshold for hydrophilines")
```



Let's use a boxplot instead:

```
boxplot(fgly, horizontal = TRUE, col = grey(0.7), main = "Distribution of Glycine frequency",
xlab = "Glycine content", ylab = "Protein count")
abline(v = 0.08, col = "red")
legend("topright",inset=0.01,lty=1,col="red",legend="Threshold for hydrophilines")
```



The threshold value for the glycine content in hydrophilines is therefore very close to the third quartile of the distribution:

```
summary(fgly)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00000 0.04907 0.06195 0.06475 0.07639 0.59240
```

We want now to compute something relatively more complex, we want the Kyte and Doolittle [9] hydropathy score of our proteins (aka GRAVY score). This is basically a linear form on amino acid frequencies:

$$s = \sum_{i=1}^{20} \alpha_i f_i$$

where α_i is the coefficient for amino acid number i and f_i the relative frequency of amino acid number i. The coefficients α_i are given in the `KD` component of the data set `EXP`:

```
data(EXP)
EXP$KD
 [1] -3.9 -3.5 -3.9 -3.5 -0.7 -0.7 -0.7 -0.7 -4.5 -0.8 -4.5 -0.8  4.5  4.5
[15]  1.9  4.5 -3.5 -3.2 -3.5 -3.2 -1.6 -1.6 -1.6 -1.6 -4.5 -4.5 -4.5 -4.5
[29]  3.8  3.8  3.8  3.8 -3.5 -3.5 -3.5 -3.5  1.8  1.8  1.8  1.8 -0.4 -0.4
[43] -0.4 -0.4  4.2  4.2  4.2  4.2  0.0 -1.3  0.0 -1.3 -0.8 -0.8 -0.8 -0.8
[57]  0.0  2.5 -0.9  2.5  3.8  2.8  3.8  2.8
```

This is for codons in lexical order, that is:

```
words()
 [1] "aaa" "aac" "aag" "aat" "aca" "acc" "acg" "act" "aga" "agc" "agg" "agt"
[13] "ata" "atc" "atg" "att" "caa" "cac" "cag" "cat" "cca" "ccc" "ccg" "cct"
[25] "cga" "cgc" "cgg" "cgt" "cta" "ctc" "ctg" "ctt" "gaa" "gac" "gag" "gat"
[37] "gca" "gcc" "gcg" "gct" "gga" "ggc" "ggg" "ggt" "gta" "gtc" "gtg" "gtt"
[49] "taa" "tac" "tag" "tat" "tca" "tcc" "tcg" "tct" "tga" "tgc" "tgg" "tgt"
[61] "tta" "ttc" "ttg" "ttt"
```

But since we are working with protein sequences here we name the coefficient according to their amino acid :

```
names(EXP$KD) <- sapply(words(),function(x) translate(s2c(x)))
```

We just need one value per amino acid, we sort them in the lexical order, and we reverse the scale so as to have positive values for hydrophilic proteins as in [5] :

```
kdc <- EXP$KD[unique(names(EXP$KD))]
kdc <- -kdc[order(names(kdc))]
kdc
  *    A    C    D    E    F    G    H    I    K    L    M    N    P    Q
0.0 -1.8 -2.5  3.5  3.5 -2.8  0.4  3.2 -4.5  3.9 -3.8 -1.9  3.5  1.6  3.5
  R    S    T    V    W    Y
4.5  0.8  0.7 -4.2  0.9  1.3
```

Now that we have the vector of coefficient α_i, we need the amino acid relative frequencies f_i, let's play with one protein first:

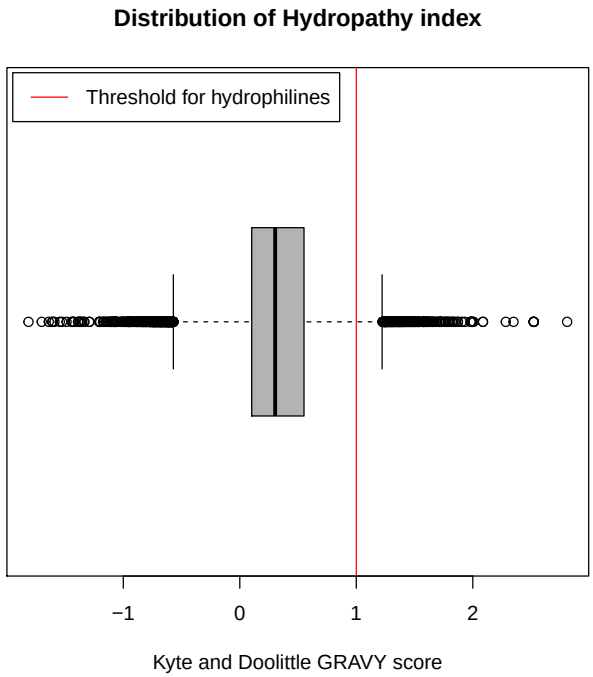
```
ath[[9523]]
[1] "MAGSQREKLKPRTKGSTRC*"
s2c(ath[[9523]])
 [1] "M" "A" "G" "S" "Q" "R" "E" "K" "L" "K" "P" "R" "T" "K" "G" "S" "T" "R"
[19] "C" "*"
table(s2c(ath[[9523]]))
* A C E G K L M P Q R S T
1 1 1 1 2 3 1 1 1 1 3 2 2
table(factor(s2c(ath[[9523]]), levels = names(kdc)))
* A C D E F G H I K L M N P Q R S T V W Y
1 1 1 0 1 0 2 0 0 3 1 1 0 1 1 3 2 2 0 0 0
```

Now that we know how to count amino acids it's relatively easy thanks to R's matrix operator **%*%** to define a vectorised function to compute a linear form on amino acid frequencies:

```
linform <- function(data, coef){
 f <- function(x){
   aaseq <- s2c(x)
   freq <- table(factor(aaseq, levels = names(coef)))/length(aaseq)
   return(coef %*% freq)
 }
 res <- sapply(data, f)
 names(res) <- NULL
 return(res)
}
kdath <- linform(ath,kdc)
```

Let's have a look at the distribution:

```
boxplot(kdath, horizontal = TRUE, col = grey(0.7),
main = "Distribution of Hydropathy index",
xlab = "Kyte and Doolittle GRAVY score")
abline(v = 1, col = "red")
legend("topleft",inset=0.01,lty=1,col="red",legend="Threshold for hydrophilines")
```



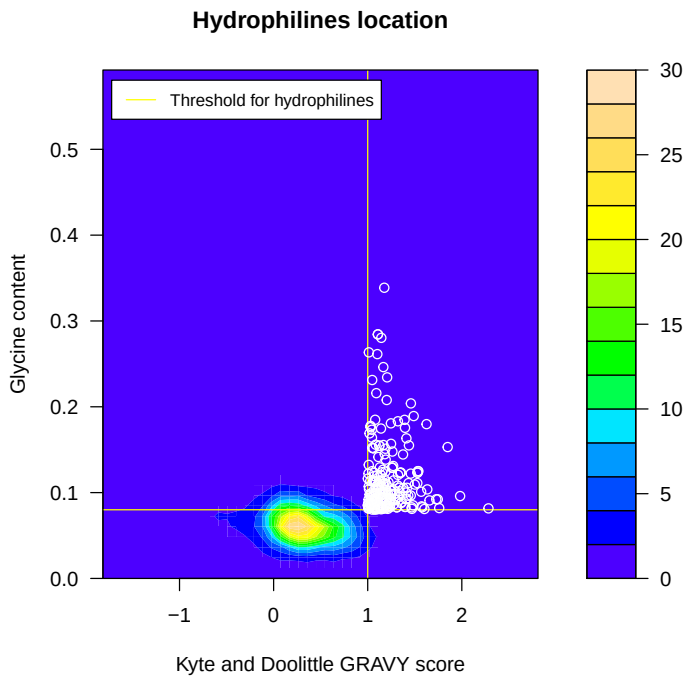
The threshold is therefore much more stringent here than the previous one on glycine content. Let's define a vector of logicals to select the hydrophilines:

```
hydrophilines <- fgly > 0.08 & kdath > 1
head(names(ath)[hydrophilines])
```

```
[1] "At1g02840.1" "At1g02840.2" "At1g02840.3" "At1g03320.1" "At1g03820.1"
[6] "At1g04450.1"
```

Check with a simple graph that there is no mistake here:

```
library(MASS)
dst <- kde2d(kdath,fgly, n = 50)
filled.contour(x = dst, color.palette = topo.colors,
plot.axes = {
  axis(1)
  axis(2)
  title(xlab="Kyte and Doolittle GRAVY score", ylab = "Glycine content",
    main = "Hydrophilines location")
  abline(v=1, col = "yellow")
  abline(h=0.08, col = "yellow")
  points(kdath[hydrophilines], fgly[hydrophilines], col = "white")
  legend("topleft",inset=0.02,lty=1,col="yellow", bg="white", legend="Threshold for hydrophilines", cex = 0.8)
  }
)
```



Everything seems to be OK, we can save the results in a data frame:

```
data.frame(list("name"=names(ath),
"KD"=kdath, "Gly"=fgly)) -> athres
head(athres)
```

```
                  name         KD        Gly
At1g01010.1 At1g01010.1  0.7297674 0.05827506
At1g01020.1 At1g01020.1 -0.1674419 0.03906250
At1g01030.1 At1g01030.1  0.8136490 0.08100559
At1g01040.1 At1g01040.1  0.4159686 0.06705081
At1g01050.1 At1g01050.1  0.4460094 0.03773585
At1g01060.1 At1g01060.1  0.7444272 0.04186047
```

We want to check now that the results are consistent with those reported previously. The following table is extracted from the file `pgen.0010026.st003.xls` provided as the supplementary material table S3 in [5] and available at `http://www.pubmedcentral.nih.gov/picrender.fcgi?artid=1189076&blobname=pgen.0010026.st003.xls`. Only the protein names, the hydrophilicity and the glycine content were extracted:

```
read.table(system.file("sequences/hannah.txt", package = "seqinr"), sep = "\t", header = TRUE)->hannah
head(hannah)
```

```
        AGI Hydrophilicity Glycine
1 At2g19570          -0.10    0.07
2 At2g45290          -0.25    0.09
3 At4g29570          -0.05    0.07
4 At4g29580          -0.10    0.06
5 At4g29600          -0.14    0.06
6 At5g28050          -0.11    0.08
```

The protein names are not exactly the same because they have no extension. As explained in [5], when multiple gene models were predicted only the first was one used. Then:

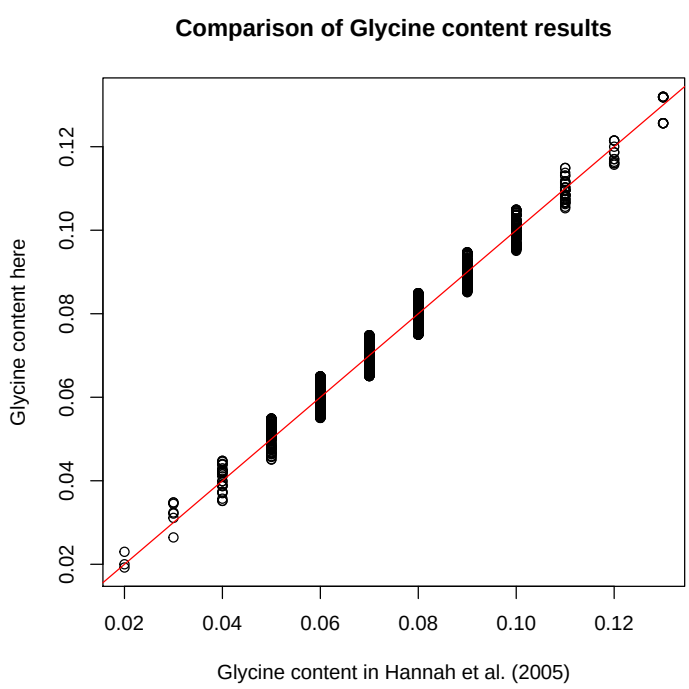
```
hannah$AGI <- paste(hannah$AGI, "1", sep = ".")
head(hannah)
          AGI Hydrophilicity Glycine
1 At2g19570.1          -0.10    0.07
2 At2g45290.1          -0.25    0.09
3 At4g29570.1          -0.05    0.07
4 At4g29580.1          -0.10    0.06
5 At4g29600.1          -0.14    0.06
6 At5g28050.1          -0.11    0.08
```

We join now the two data frames thanks to their common key:

```
join <- merge(hannah, athres, by.x = "AGI", by.y = "name")
head(join)
          AGI Hydrophilicity Glycine          KD        Gly
1 At1g01120.1          -0.10    0.06 0.106994329 0.05871212
2 At1g01390.1           0.02    0.06 0.009147609 0.06458333
3 At1g01390.1           0.02    0.06 0.009147609 0.06458333
4 At1g01420.1          -0.05    0.07 0.062033195 0.07276507
5 At1g01420.1          -0.05    0.07 0.062033195 0.07276507
6 At1g01480.1          -0.20    0.07 0.200804829 0.06653226
```

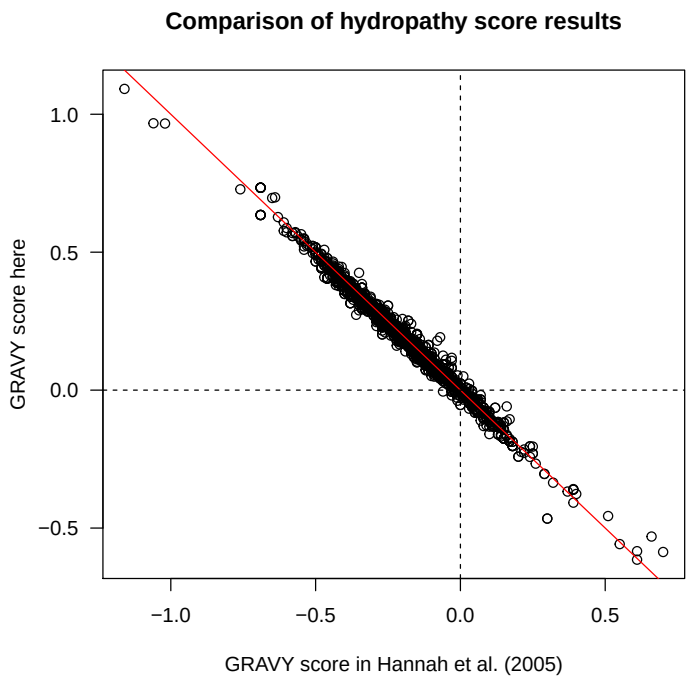
Let's compare the glycine content :

```
plot(join$Glycine, join$Gly, xlab = "Glycine content in Hannah et al. (2005)",
ylab = "Glycine content here", main = "Comparison of Glycine content results")
abline(c(0,1), col = "red")
```



The results are consistent, we have just lost some resolution because there are only two figures after the decimal point in the Excel[2] file. Let's have a look at the GRAVY score now:

```
plot(join$Hydrophilicity, join$KD, xlab = "GRAVY score in Hannah et al. (2005)",
ylab = "GRAVY score here", main = "Comparison of hydropathy score results", las = 1)
abline(c(0,-1), col = "red")
abline(v=0, lty=2)
abline(h=0, lty=2)
```



The results are consistent, it's hard to say whether the small differences are due to Excel rounding errors or because the method used to compute the GRAVY score was not exactly the same (in [5] they used the mean over a sliding window).

[2]this software is a real **pain** for the reproducibility of results. This is well documented, see `http://www.burns-stat.com/pages/Tutor/spreadsheet_addiction.html` and references therein.

2 Importing aligned sequence data

2.1 Aligned sequences files examples

2.1.1 mase

Mase format is a flatfile format use by the SeaView multiple alignment editor [3], developed by Manolo Gouy and available at `http://pbil.univ-lyon1.fr/software/seaview.html`. The mase format is used to store nucleotide or protein multiple alignments. The beginning of the file must contain a header containing at least one line (but the content of this header may be empty). The header lines must begin by ;;. The body of the file has the following structure: First, each entry must begin by one (or more) commentary line. Commentary lines begin by the character ;. Again, this commentary line may be empty. After the commentaries, the name of the sequence is written on a separate line. At last, the sequence itself is written on the following lines.

```
masef <- system.file("sequences/test.mase", package = "seqinr")
cat(readLines(masef), sep = "\n")
```

```
;;Aligned by clustal on Tue Jun 30 17:36:11 1998
;empty description
Langur
-KIFERCELARTLKKLGLDGYKGVSLANWVCLAKWESGYNTEATNYNPGDESTDYGIFQINSRYWCNNGKPGAVDACHISCSALLQNNIADAVACAKRVVSDQGIRAWVAWRN
;
Baboon
-KIFERCELARTLKRLGLDGYRGISLANWVCLAKWESDYNTQATNYNPGDQSTDYGIFQINSHYWCNDGKPGAVNACHISCNALLQDNITDAVACAKRVVSDQGIRAWVAWRN
;
Human
-KVFERCELARTLKRLGMDGYRGISLANWMCLAKWESGYNTRATNYNAGDRSTDYGIFQINSRYWCNDGKPGAVNACHLSCSALLQDNIADAVACAKRVVRDQGIRAWVAWRN
;
Rat
-KTYERCEFARTLKRNGMSGYYGVSLADWVCLAQHESNYNTQARNYDPGDQSTDYGIFQINSRYWCNDGKPRAKNACGIPCSALLQDDITQAIQCAKRVVRDQGIRAWVAWQR
;
Cow
-KVFERCELARTLKKLGLDGYKGVSLANWLCLTKWESSYNTKATNYNPSSESTDYGIFQINSKWWCNDGKPNAVDGCHVSCSELMENDIAKAVACAKKIVSEQGITAWVAWKS
;
Horse
-KVFSKCELAHKLKAQEMDGFGGYSLANWVCMAEYESNFNTRAFNGKNANGSSDYGLFQLNNKWWCKDNKRSSSNACNIMCSKLLDENIDDDISCAKRVVRDKGMSAWKAWVK
```

A screenshot copy of the same file as seen under SeaView is given in figure 2.

2.1.2 clustal

The CLUSTAL format (*.aln) is the format of the ClustalW multialignment tool output [6, 15]. It can be described as follows. The word CLUSTAL is on the first line of the file. The alignment is displayed in blocks of a fixed length, each line in the block corresponding to one sequence. Each line of each block starts with the sequence name (maximum of 10 characters), followed by at least one space character. The sequence is then displayed in upper or lower cases, '-' denotes gaps. The residue number may be displayed at the end of the first line of each block.

```
clustalf <-system.file("sequences/test.aln", package = "seqinr")
cat(readLines(clustalf), sep = "\n")
```

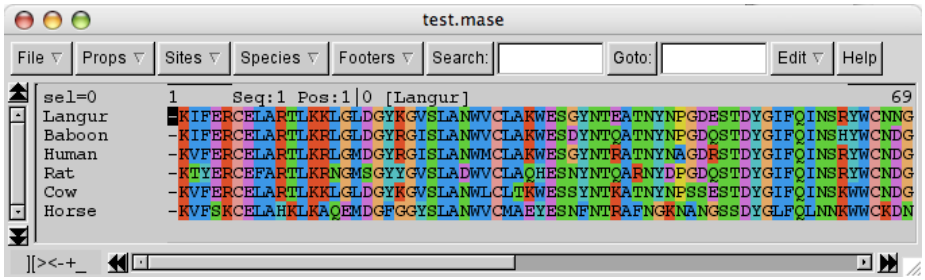


Figure 2: The file `test.mase` under SeaView. This is a graphical multiple sequence alignment editor developped by Manolo Gouy [3]. SeaView is able to read and write various alignment formats (NEXUS, MSF, CLUSTAL, FASTA, PHYLIP, MASE). It allows to manually edit the alignment, and also to run DOT-PLOT or CLUSTALW programs to locally improve the alignment.

```
CLUSTAL W (1.82) multiple sequence alignment


FOSB_MOUSE      MFQAFPGDYDSGSRCSSSPSAESQYLSSVDSFGSPPTAAASQECAGLGEMPGSFVPTVTA 60
FOSB_HUMAN      MFQAFPGDYDSGSRCSSSPSAESQYLSSVDSFGSPPTAAASQECAGLGEMPGSFVPTVTA 60
                ************************************************************

FOSB_MOUSE      ITTSQDLQWLVQPTLISSMAQSQGQPLASQPPAVDPYDMPGTSYSTPGLSAYSTGGASGS 120
FOSB_HUMAN      ITTSQDLQWLVQPTLISSMAQSQGQPLASQPPVVDPYDMPGTSYSTPGMSGYSSGGASGS 120
                *******************************.***************:*.**:******

FOSB_MOUSE      GGPSTSTTTSGPVSARPARARPRRPREETLTPEEEEKRRVRRERNKLAAAKCRNRRRELT 180
FOSB_HUMAN      GGPSTSGTTSGPGPARPARARPRRPREETLTPEEEEKRRVRRERNKLAAAKCRNRRRELT 180
                ****** ***** .**********************************************

FOSB_MOUSE      DRLQAETDQLEEEKAELESEIAELQKEKERLEFVLVAHKPGCKIPYEEGPGPGPLAEVRD 240
FOSB_HUMAN      DRLQAETDQLEEEKAELESEIAELQKEKERLEFVLVAHKPGCKIPYEEGPGPGPLAEVRD 240
                ************************************************************

FOSB_MOUSE      LPGSTSAKEDGFGWLLPPPPPPPLPFQSSRDAPPNLTASLFTHSEVQVLGDPFPVVSPSY 300
FOSB_HUMAN      LPGSAPAKEDGFSWLLPPPPPPPLPFQTSQDAPPNLTASLFTHSEVQVLGDPFPVVNPSY 300
                ****:.******.**************:*:**************************.***

FOSB_MOUSE      TSSFVLTCPEVSAFAGAQRTSGSEQPSDPLNSPSLLAL 338
FOSB_HUMAN      TSSFVLTCPEVSAFAGAQRTSGSDQPSDPLNSPSLLAL 338
                ***********************:**************
```

2.1.3 phylip

PHYLIP is a tree construction program [1]. The format is as follows: the number of sequences and their length (in characters) is on the first line of the file. The alignment is displayed in an interleaved or sequential format. The sequence names are limited to 10 characters and may contain blanks.

```
phylipf <- system.file("sequences/test.phylip", package = "seqinr")
cat(readLines(phylipf), sep = "\n")
  5    42
Turkey     AAGCTNGGGC ATTTCAGGGT
Salmo gairAAGCCTTGGC AGTGCAGGGT
H. SapiensACCGGTTGGC CGTTCAGGGT
Chimp      AAACCCTTGC CGTTACGCTT
```

```
Gorilla   AAACCCTTGC CGGTACGCTT

GAGCCCGGGC AATACAGGGT AT
GAGCCGTGGC CGGGCACGGT AT
ACAGGTTGGC CGTTCAGGGT AA
AAACCGAGGC CGGGACACTC AT
AAACCATTGC CGGTACGCTT AA
```

2.1.4 msf

MSF is the multiple sequence alignment format of the GCG sequence analysis package (`http://www.accelrys.com/products/gcg/index.html`). It begins with the line (all uppercase) !!NA_MULTIPLE_ALIGNMENT 1.0 for nucleic acid sequences or !!AA_MULTIPLE_ALIGNMENT 1.0 for amino acid sequences. Do not edit or delete the file type if its present (optional). A description line which contains informative text describing what is in the file. You can add this information to the top of the MSF file using a text editor (optional). A dividing line which contains the number of bases or residues in the sequence, when the file was created, and importantly, two dots (..) which act as a divider between the descriptive information and the following sequence information (required). msf files contain some other information: the Name/Weight, a Separating Line which must include two slashes (//) to divide the name/weight information from the sequence alignment (required) and the multiple sequence alignment.

```
msff <- system.file("sequences/test.msf", package = "seqinr")
cat(readLines(msff), sep = "\n")
```

```
PileUp of: @Pi3k.Fil

Symbol comparison table: GenRunData:Pileuppep.Cmp  CompCheck: 1254

                   GapWeight: 3.000
             GapLengthWeight: 0.100

 Pi3k.Msf  MSF: 377  Type: P  July 12, 1996 10:40  Check: 167 ..

 Name: Tor1_Yeast       Len:   377  Check: 7773  Weight:  1.00
 Name: Tor2_Yeast       Len:   377  Check: 8562  Weight:  1.00
 Name: Frap_Human       Len:   377  Check: 9129  Weight:  1.00
 Name: Esr1_Yeast       Len:   377  Check: 8114  Weight:  1.00
 Name: Tel1_Yeast       Len:   377  Check: 1564  Weight:  1.00
 Name: Pi4k_Human       Len:   377  Check: 8252  Weight:  1.00
 Name: Stt4_Yeast       Len:   377  Check: 9117  Weight:  1.00
 Name: Pik1_Yeast       Len:   377  Check: 3455  Weight:  1.00
 Name: P3k1_Soybn       Len:   377  Check: 4973  Weight:  1.00
 Name: P3k2_Soybn       Len:   377  Check: 4632  Weight:  1.00
 Name: Pi3k_Arath       Len:   377  Check: 3585  Weight:  1.00
 Name: Vp34_Yeast       Len:   377  Check: 5928  Weight:  1.00
 Name: P11a_Human       Len:   377  Check: 6597  Weight:  1.00
 Name: P11b_Human       Len:   377  Check: 8486  Weight:  1.00

//

            1                                                  50
Tor1_Yeast  .......GHE DIRQDSLVMQ LFGLVNTLLK NDSECFKRHL DIQQYPAIPL
Tor2_Yeast  .......GHE DIRQDSLVMQ LFGLVNTLLQ NDAECFRRHL DIQQYPAIPL
Frap_Human  .......GHE DLRQDERVMQ LFGLVNTLLA NDPTSLRKNL SIQRYAVIPL
Esr1_Yeast  .......KKE DVRQDNQYMQ FATTMDFLLS KDIASRKRSL GINIYSVLSL
Tel1_Yeast  .KALMKGSND DLRQDAIMEQ VFQQVNKVLQ NDKVLRNLDL GIRTYKVVPL
Pi4k_Human  ..AAIFKVGD DCRQDMLALQ IIDLFKNIFQ LV....GLDL FVFPYRVVAT
Stt4_Yeast  ..AAIFKVGD DCRQDVLALQ LISLFRTIWS SI....GLDV YVFPYRVTAT
Pik1_Yeast  ...VIAKTGD DLRQEAFAYQ MIQAMANIWV KE....KVDV WVKRMKILIT
P3k1_Soybn  TCKIIFKKGD DLRQDQLVVQ MVSLMDRLLK LE....NLDL HLTPYKVLAT
```

```
P3k2_Soybn  ....IFKKGD DIRQDQLVVQ MVSLMDRLLK LE....NLDL HLTPYKVLAT
Pi3k_Arath  ..KLIFKKGD DLRQDQLVVQ MVWLMDRLLK LE....NLDL CLTPYKVLAT
Vp34_Yeast  .YHLMFKVGD DLRQDQLVVQ IISLMNELLK NE....NVDL KLTPYKILAT
P11a_Human  ...IIFKNGD DLRQDMLTLQ IIRIMENIWQ NQ....GLDL RMLPYGCLSI
P11b_Human  ...VIFKNGD DLRQDMLTLQ MLRLMDLLWK EA....GLDL RMLPYGCLAT

            51                                                 100
Tor1_Yeast  SPKSGLLGWV PNSDTFHVLI REHRDAKKIP LNIEHWVMLQ MAPDYENLTL
Tor2_Yeast  SPKSGLLGWV PNSDTFHVLI REHREAKKIP LNIEHWVMLQ MAPDYDNLTL
Frap_Human  STNSGLIGWV PHCDTLHALI RDYREKKKIL LNIEHRIMLR MAPDYDHLTL
Esr1_Yeast  REDCGILEMV PNVVTLRSIL STKYESLKIK Y.....SLKS LHDRWQHTAV
Tel1_Yeast  GPKAGIIEFV ANSTSLHQIL SKLHTNDKIT FDQARKGMKA VQTKSN....
Pi4k_Human  APGCGVIECI PDCTS..... .......... RDQLGRQTDF GMYDYFTRQY
Stt4_Yeast  APGCGVIDVL PNSVS..... .......... RDMLGREAVN GLYEYFTSKF
Pik1_Yeast  SANTGLVETI TNAMSVHSIK KALTKKMIED AELDDKGGIA SLNDHFLRAF
P3k1_Soybn  GQDEGMLEFI P.SRSLAQI. .......... ..LSENRSII SYLQ......
P3k2_Soybn  GQDEGMLEFI P.SRSLAQI. .......... ..LSENRSII SYLQ......
Pi3k_Arath  GHDEGMLEFI P.SRSLAQI. .......... ..LSEHRSIT SYLQ......
Vp34_Yeast  GPQEGAIEFI P.NDTLASI. .......... ..LSKYHGIL GYLK......
P11a_Human  GDCVGLIEVV RNSHTIMQI. .......... ..Q.CKGGLK GALQFNSHTL
P11b_Human  GDRSGLIEVV STSETIADI. .......... ..QLNSSNVA AAAAFNKDAL
```

2.1.5 FASTA

Sequence in fasta format begins with a single-line description (distinguished by a greater-than (>) symbol), followed by sequence data on the next line.

```
fastaf <- system.file("sequences/Anouk.fasta", package = "seqinr")
cat(readLines(fastaf), sep = "\n")
```

```
>LmjF01.0030
ATGATGTCGGCCGAGCCGCCGTCGTCGCAGCCGTACATCAGCGACGTGCTGCGGCGGTAC
CAGCTGGAGCGCTTTCAGTGTGCCTTTGCATCGAGCATGACCATCAAGGACCTCCTCGCC
CTGCAGCCAGAGGACTTCAACCGCTACGGCGTCGTAGAGGCGATGGACATTTTGCGGCTG
CGTGACGCCATCGAGTACATCAAGGCTAATCCGCTCCCCGCCTCGCGCTCTGGCAGTGAC
GTGCTCGACAACGACGGCGACGGCGACGGCGACGACAGTACGCCGGAGGGGAAGGAGGGG
TGCTCGACGGAGCGCCGGCGGCAGTACACAGCACGCGGAACCACAGTCCTTTGCCGGTCG
ACCGACACCGCCGAGGAGGTGAAGCGCAAGAGCCGCATCCTCGTCGCCATTCGCAAGCGT
CCGCTCAGCGCCGGGGAGCAGACGAACGGCTTCACGGACATCATGGACGCCGACAACAGC
GGCGAGATTGTGCTGAAGGAGCCAAAGGTGAAGGTCGACCTCCGCAAGTACACCCACGTG
CACCGCTTCTTCTTCGACGAGGTTTTCGACGAGGCCTGCGACAACGTCGACGTGTACAAC
CGCGCTGCCCGCGCGCTGATCGACACCGTCTTCGACGGCGGCTGCGCGACATGCTTCGCC
TATGGACAGACAGGGAGCGGCAAGACACACACGATGCTGGGCAAGGGCCCCGAGCCGGGC
CTCTACGCACTCGCCGCCAAAGACATGTTTGACCGCCTCACGAGCGACACGCGCATCGTC
GTTTCCTTTTACGAGATCTACAGCGGGAAGCTCTTTGACTTGCTGAACGGCCGGCGACCC
CTGCGAGCCCTCGAGGACGACAAGGGCCGGGTGAACATCCGCGGCCTCACCGAACACTGC
TCTACCAGCGTGGAGGACCTCATGACGATCATCGACCAGGGCAGCGGTGTTCGCAGCTGC
GGCTCCACCGGCGCCAATGACACAAGCTCCCGCTCCCACGCCATTCTCGAGATCAAGCTC
AAGGCGAAACGGACGTCGAAGCAGAGCGGCAAGTTCACGTTCATCGACCTCGCTGGAAGC
GAGCGCGGCGCTGACACGGTGGACTGCGCGCGACAGACACGCCTCGAAGGGGCGGAGATC
AACAAGAGCCTACTCGCGCTGAAGGAGTGCATTCGTTTTTTAGATCAGAACAGGAAGCAC
GTCCCGTTCCGCGGCTCGAAGCTGACTGAGGTGCTCCGCGACTCGTTTATCGGCAACTGC
CGCACGGTGATGATCGGCGCCGTCTCTCCGTCGAACAACAATGCCGAGCACACGCTGAAC
ACGCTGCGCTACGCCGATCGTGTCAAGGAGCTGAAGCGCAACGCCACGGAGCGGCGCACT
GTGTGCATGCCCGACGACCAGGAAGAGGCCTTCTTTGACACGACCGAGAGCAGGCCACCG
TCGCGGAGGACGACAACTCGCCTTTCTACGGCCGCCCCGCTTTTCTCCGGCTCTTCGACG
GCTGCGCCAGCACTTAGAAGCACGCTACTCAGCAGCCGCTCCGTCAACACACTCTCGCCG
TCGTCGCAGGCCAAGTCGACTCTCGTCACCCCGAAGCCGCCGTCGCGCGATCGGACTCCG
GACATGGTGTGCACTAAGCGGCCCCGCGACTCAGACAGAAGCGGCGAGGACGAAGTGGTA
GCGCGGCCGAGTGGGCGCCCAAGCTTCAAGCGCTTCGAGAGCGGCGCCGAGCTTGTCGCG
GCCCAGCGCAGTCGCGTCATTGACCAATACAACGCCTACCTCGAGACGGACATGAACTGT
ATCAAGGAGGAGTACCAGGTGAAGTACGACGCAGAGCAGATGAACGCCAACACGCGCAGC
TTTGTGGAGCGCGCACGTCTGCTGGTGAGCGAGAAACGGCGCGCGATGGAGTCCTTCCTA
ACGCAGCTGGAGGAGCTCGACAAGATCGCGCAGCAGGTCGCCGACATCACCGCCTTTCAG
CAGCACCTGCCGCCAACG
>LinJ01.0030
ATGATGTCGGCCGAGCCGCCGTCGTCGCAGCCGTACATCAGCGACGTGCTGCGGCGGTAC
CAGCTGGAGCGCTTTCAGAGTTCCTTTGCATCGAGCATGACCATCAAGGACCTCCTCGCC
CTGCAGCCGGAGGACTTCAACCGCTACGGCGTCGTAGAGGCAATGGACATTTTGCGGCTG
CGCGACGCCATCGAGTACATCAAGGCCAACCCGCTCCCCGCCTCGCGCTCCGGCAGTGAC
GTGCTCGACAACGACGGCGACGGCGACGGCGACGACAGTACGCCGGAGGGGAAGGAGGGG
TGCTCGACGGAGCGCCGACGGCAGTACACAGCACGCGGAACCACCGTCCTTTGCGGGTCG
```

```
ACCGACACCGCCGAGGAGGTGAAGCGCAAGAGCCGCATCATCGTCGCCATTCGCAAGCGT
CCGCTCAGCGCCGGGGAGCAGACGAACGGCTTCACGGACATCATGGACGCCGACAACAAC
GGCGAGATTGTGCTGAAGGAGCCAAAGGTGAAGGTCGACCTCCGCAAGTACACCCACGTG
CACCGCTTCTTCTTCGACGAGGTTTTCGACGAGGCGTGCGACAACGTCGACGTGTACAAC
CGCGCTGCCCGCGCGCTGATCGACACCGTCTTCGACGGCGGCTGCGCGACATGCTTCGCC
TATGGGCAGACAGGGAGCGGCAAGACACACACGATGCTCGGCAAGGGCCCCGAGCCGGGC
CTGTACGCACTCGCCGCCAAAGACATGTTTGACCGCCTCACGAGCGACACGCGCATCGTT
GTTTCCTTTTACGAGATCTACAGCGGGAAGCTCTTTGACTTGCTGAACGGCCGGCGACCA
CTGCGAGCCCTCGAGGACGACAAGGGGAGGGTGAACATCCGCGGCCTCACCGAACACTGC
TCTACCAGCGTGGAGGACCTCATGACGATCATCGACCAGGGCAGCGGCGTTCGCAGCTGC
GGCTCCACCGGCGCCAACGACACGAGCTCCCGCTCCCACGCCATTCTCGAGATCAAGCTC
AAGGCGAAACGGACGTCGAAGCAGAGCGGCAAGTTCACATTCATCGACCTCGCTGGAAGC
GAGCGCGGCGCCGACACGGTGGATTGCGCGCGACAGACACGCCTCGAAGGGGCGGAGATT
AACAAGAGCCTACTCGCTCTGAAGGAGTGCATTCGTTTTTTAGATCAGAACAGGAAGCAC
GTCCCGTTCCGCGGCTCGAAGCTGACTGAGGTGCTCCGCGACTCGTTTATCGGCAACTGC
CGCACGGTGATGATCGGCGCCGTCTCTCCGTCCAACAACAATGCCGAGCACACGCTGAAC
ACGTTGCGCTACGCCGATCGCGTCAAGGAGCTGAAGCGCAACGCCACGGAGCGGCGCACC
GTGTGCGTGCCCAACGACCAGGAAGAGGCCTTCTTTGACACGACCGAGAGCAGGCCACCG
TCGCGGAGGACGACAACTCGGCTTTCTGCGGCCGCCCCGCTTTTCTCCGGCACTTCGACG
GCTGCCCCAGCATGTAAAAGCACGTTGCTCAGCAGCCGCTCCGTCAACACACTCTCGCCG
TCGTCGCAGGGCAAGTCGACTCTCGTCACCCCGAAGCCACTGTCGCGCGATCGGACTCCG
GACATGGTGTGCGCTAAGCGGCCCCGCGACTCAGACCGAAGCGGCGAAGACGAAGTGGTG
GCGCGGCCGAGTGGGCGCCCAAGCTTCAAGCGCTTCGAGGGCGGCGCCGAGCTCGTGGCG
GCCCAGCGCAGTCGTGTCATTGACCAATACAACGCCTACCTCGAGACGGACATGAACTGT
ATCAAGGAGGAGTACCAGGTGAAGTACGACGCAGAGCAGATGAACGCCAACACGCGCACC
TTTGTCGAGCGCGCACGCCTGCTGGTGAGCGAGAAGCGGCGCGCGATGGAGTCCTTCCTA
ACGCAGCTGGACGAGCTCGATAAGATCGCGCAGCAGGTCGCCAGCATCACCGCCTTTCAG
CAGCACCTGCCGCCAACG
```

2.2 The function `read.alignment()`

Aligned sequence data are very important in evolutionary studies, in this representation all vertically aligned positions are supposed to be homologous, that is sharing a common ancestor. This is a mandatory starting point for comparative studies. There is a function in `seqin`**R** called `read.alignment()` to read aligned sequences data from various formats (`mase`, `clustal`, `phylip`, `fasta` or `msf`) produced by common external programs for multiple sequence alignment.

```
example(read.alignment)
rd.lgn mase.res   <- read.alignment(file = system.file("sequences/test.mase", package = "seqinr"),
rd.lgn  format = "mase")

rd.lgn clustal.res <- read.alignment(file = system.file("sequences/test.aln", package = "seqinr"),
rd.lgn  format="clustal")

rd.lgn phylip.res  <- read.alignment(file = system.file("sequences/test.phylip", package = "seqinr"),
rd.lgn  format = "phylip")

rd.lgn msf.res      <- read.alignment(file = system.file("sequences/test.msf", package = "seqinr"),
rd.lgn  format = "msf")

rd.lgn fasta.res    <- read.alignment(file = system.file("sequences/Anouk.fasta", package = "seqinr"),
rd.lgn  format = "fasta")

rd.lgn #
rd.lgn # Quality control routine sanity checks:
rd.lgn #
rd.lgn
rd.lgn data(mase); stopifnot(identical(mase, mase.res))

rd.lgn data(clustal); stopifnot(identical(clustal, clustal.res))

rd.lgn data(phylip); stopifnot(identical(phylip, phylip.res))

rd.lgn data(msf); stopifnot(identical(msf, msf.res))

rd.lgn data(fasta); stopifnot(identical(fasta, fasta.res))
```

2.3 A simple example with the louse-gopher data

Let's give an example. The gene coding for the mitochondrial cytochrome oxidase I is essential and therefore often used in phylogenetic studies because of its ubiquitous nature. The following two sample tests of aligned sequences of this gene (extracted from ParaFit [10]), are distributed along with the **seqinR** package:

```
louse <- read.alignment(system.file("sequences/louse.fasta", package = "seqinr"), format = "fasta")
louse$nam
```

```
[1] "gi|548117|gb|L32667.1|GYDCYTOXIB Geomydoecus chapini mitochondrial cytochrome oxidase I gene, partial cds"
[2] "gi|548119|gb|L32668.1|GYDCYTOXIC Geomydoecus cherriei mitochondrial cytochrome oxidase I gene, partial cds"
[3] "gi|548121|gb|L32669.1|GYDCYTOXID Geomydoecus costaricensis mitochondrial cytochrome oxidase I gene, partial
[4] "gi|548125|gb|L32671.1|GYDCYTOXIF Geomydoecus ewingi mitochondrial cytochrome oxidase I gene, partial cds"
[5] "gi|548127|gb|L32672.1|GYDCYTOXIG Geomydoecus geomydis mitochondrial cytochrome oxidase I gene, partial cds"
[6] "gi|548131|gb|L32675.1|GYDCYTOXII Geomydoecus oklahomensis mitochondrial cytochrome oxidase I gene, partial c
[7] "gi|548133|gb|L32676.1|GYDCYTOXIJ Geomydoecus panamensis mitochondrial cytochrome oxidase I gene, partial cds
[8] "gi|548137|gb|L32678.1|GYDCYTOXIL Geomydoecus setzeri mitochondrial cytochrome oxidase I gene, partial cds"
```

```
gopher <- read.alignment(system.file("sequences/gopher.fasta", package = "seqinr"), format = "fasta")
gopher$nam
```

```
[1] "gi|548223|gb|L32683.1|PPGCYTOXIA Geomys breviceps mitochondrial cytochrome oxidase I gene, partial cds"
[2] "gi|548197|gb|L32686.1|OGOCYTOXIA Orthogeomys cavator mitochondrial cytochrome oxidase I gene, partial cds"
[3] "gi|548199|gb|L32687.1|OGOCYTOXIB Orthogeomys cherriei mitochondrial cytochrome oxidase I gene, partial cds"
[4] "gi|548201|gb|L32691.1|OGOCYTOXIC Orthogeomys underwoodi mitochondrial cytochrome oxidase I gene, partial cds
[5] "gi|548203|gb|L32692.1|OGOCYTOXID Orthogeomys hispidus mitochondrial cytochrome oxidase I gene, partial cds"
[6] "gi|548229|gb|L32693.1|PPGCYTOXID Geomys bursarius mitochondrial cytochrome oxidase I gene, partial cds"
[7] "gi|548231|gb|L32694.1|PPGCYTOXIE Geomys bursarius mitochondrial cytochrome oxidase I gene, partial cds"
[8] "gi|548205|gb|L32696.1|OGOCYTOXIE Orthogeomys heterodus mitochondrial cytochrome oxidase I gene, partial cds"
```

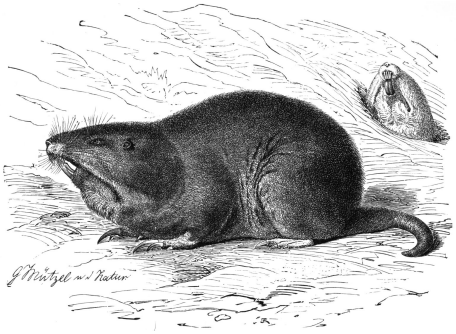


Figure 3: Louse (left) and gopher (right). Images are from the wikipedia (`http://www.wikipedia.org/`). The picture of the chewing louse *Damalinia limbata* found on Angora goats was taken by Fiorella Carnevali (ENEA, Italy). The gopher drawing is from Gustav Mützel, Brehms Tierleben, Small Edition 1927.

The aligned sequences are now imported in your R environment. The 8 genes of the first sample are from various species of louse (insects parasitics on warm-blooded animals) and the 8 genes of the second sample are from their corresponding gopher hosts (a subset of rodents), see figure 3 :

```
l.names <- readLines(system.file("sequences/louse.names", package = "seqinr"))
l.names
```

```
[1] "G.chapini "  "G.cherriei " "G.costaric " "G.ewingi  "  "G.geomydis "
[6] "G.oklahome " "G.panamens " "G.setzeri  "
```

```
g.names <- readLines(system.file("sequences/gopher.names", package = "seqinr"))
g.names
```

```
[1] "G.brevicep " "O.cavator "  "O.cherriei " "O.underwoo " "O.hispidus "
[6] "G.burs1 "    "G.burs2 "    "O.heterodu"
```

`Seqin`**R** has very few methods devoted to phylogenetic analyses but many are available in the `ape` package [12]. This allows for a very fine tuning of the graphical outputs of the analyses thanks to the power of the R facilities. For instance, a natural question here would be to compare the topology of the tree of the hosts and their parasites to see if we have congruence between host and parasite evolution. In other words, we want to display two phylogenetic trees face to face. This would be tedious with a program devoted to the display of a single phylogenetic tree at time, involving a lot of manual copy/paste operations, hard to reproduce, and then boring to maintain with data updates.

How does it looks under R? First, we need to *infer* the tree topologies from data. Let's try as an *illustration* the famous neighbor-joining tree estimation of Saitou and Nei [14] with Jukes and Cantor's correction [7] for multiple substitutions.

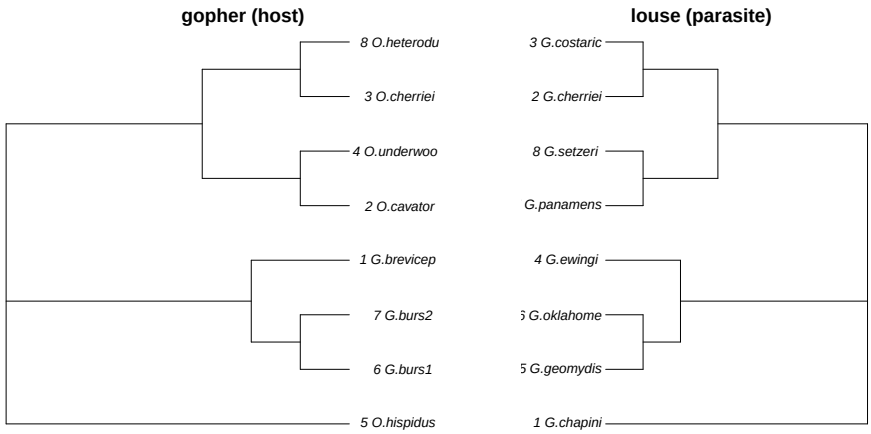
```
library(ape)
louse.JC <- dist.dna(as.DNAbin(louse), model = "JC69")
gopher.JC <- dist.dna(as.DNAbin(gopher), model = "JC69")
l <- nj(louse.JC)
g <- nj(gopher.JC)
```

Now we have an estimation for *illustrative* purposes of the tree topology for the parasite and their hosts. We want to plot the two trees face to face, and for this we must change R graphical parameters. The first thing to do is to save the current graphical parameter settings so as to be able to restore them later:

```
op <- par(no.readonly = TRUE)
```

The meaning of the `no.readonly = TRUE` option here is that graphical parameters are not all settable, we just want to save those we can change at will. Now, we can play with graphics :

```
g$tip.label <- paste(1:8, g.names)
l$tip.label <- paste(1:8, l.names)
layout(matrix(data = 1:2, nrow = 1, ncol = 2), width=c(1.4, 1))
par(mar=c(2,1,2,1))
plot(g, adj = 0.8, cex = 1.4, use.edge.length=FALSE,
  main = "gopher (host)", cex.main = 2)
plot(l,direction="l", use.edge.length=FALSE, cex = 1.4,
  main = "louse (parasite)", cex.main = 2)
```



We now restore the old graphical settings that were previously saved:

```
par(op)
```

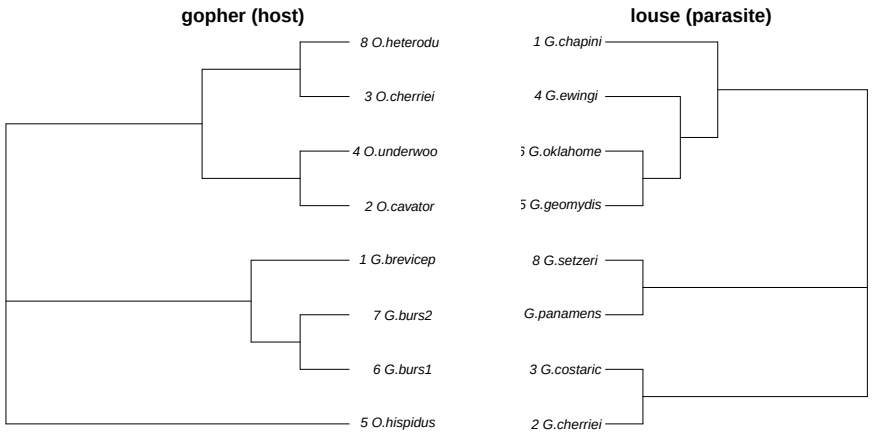
OK, this may look a little bit obscure if you are not fluent in programming, but please try the following experiment. In your current working directory, that is in the directory given by the `getwd()` command, create a text file called `essai.r` with your favourite text editor, and copy/paste the previous R commands, that is :

```
louse <- read.alignment(system.file("sequences/louse.fasta", package = "seqinr"), format = "fasta")
gopher <- read.alignment(system.file("sequences/gopher.fasta", package = "seqinr"), format = "fasta")
l.names <- readLines(system.file("sequences/louse.names", package = "seqinr"))
g.names <- readLines(system.file("sequences/gopher.names", package = "seqinr"))
library(ape)
louse.JC <- dist.dna(as.DNAbin(louse), model = "JC69")
gopher.JC <- dist.dna(as.DNAbin(gopher), model = "JC69")
l <- nj(louse.JC)
g <- nj(gopher.JC)
g$tip.label <- paste(1:8, g.names)
l$tip.label <- paste(1:8, l.names)
layout(matrix(data = 1:2, nrow = 1, ncol = 2), width=c(1.4, 1))
par(mar=c(2,1,2,1))
plot(g, adj = 0.8, cex = 1.4, use.edge.length=FALSE,
  main = "gopher (host)", cex.main = 2)
plot(l,direction="l", use.edge.length=FALSE, cex = 1.4,
  main = "louse (parasite)", cex.main = 2)
```

Make sure that your text has been saved and then go back to R console to enter the command :

```
source("essai.r")
```

This should reproduce the previous face-to-face phylogenetic trees in your R graphical device. Now, your boss is unhappy with working with the Jukes and Cantor's model [7] and wants you to use the Kimura's 2-parameters distance [8] instead. Go back to the text editor to change `model = "JC69"` by `model = "K80"`, save the file, and in the R console `source("essai.r")` again, you should obtain the following graph :



Now, something even worst, there was a error in the aligned sequence set: the first base in the first sequence in the file `louse.fasta` is not a C but a T. To locate the file on your system, enter the following command:

```
system.file("sequences/louse.fasta", package = "seqinr")
[1] "/Users/lobry/seqinr/pkg.Rcheck/seqinr/sequences/louse.fasta"
```

Open the `louse.fasta` file in your text editor, fix the error, go back to the R console to `source("essai.r")` again. That's all, your graph is now consistent with the updated dataset.

Session Informations

This part was compiled under the following R environment:

- R version 3.2.4 (2016-03-10), `x86_64-apple-darwin13.4.0`
- Locale: `fr_FR.UTF-8/fr_FR.UTF-8/fr_FR.UTF-8/C/fr_FR.UTF-8/fr_FR.UTF-8`
- Base packages: base, datasets, graphics, grDevices, grid, methods, stats, utils
- Other packages: ade4 1.7-4, ape 3.5, grImport 0.9-0, MASS 7.3-45, seqinr 3.1-5, tseries 0.10-35, XML 3.98-1.4, xtable 1.8-2
- Loaded via a namespace (and not attached): lattice 0.20-33, nlme 3.1-125, quadprog 1.5-5, tools 3.2.4, zoo 1.7-12

There were two compilation steps:

- R compilation time was: Thu Jun 2 15:58:57 2016
- LaTeX compilation time was: June 2, 2016

References

[1] J. Felsenstein. PHYLIP-phylogeny inference package (version 3.2). *Cladistics*, 5:164–166, 1989.

[2] A.C. Frank and J.R. Lobry. Oriloc: prediction of replication boundaries in unannotated bacterial chromosomes. *Bioinformatics*, 16(6):560–561, 2000.

[3] N. Galtier, M. Gouy, and C. Gautier. SeaView and Phylo_win, two graphic tools for sequence alignment and molecular phylogeny. *Comput. Applic. Biosci.*, 12:543–548, 1996.

[4] A. Garay-Arroyo, J.M. Colmenero-Flores, A. Garciarrubio, and A.A. Covarrubias. Highly hydrophilic proteins in prokaryotes and eukaryotes are common during conditions of water deficit. *J. Biol. Chem.*, 275:5668–5674, 2000.

[5] M.A. Hannah, A.G. Heyer, and D.K. Hincha. A global survey of gene regulation during cold acclimation in *Arabidopsis thaliana*. *PLoS Genet.*, 1:e26, 2005.

[6] D. G. Higgins and P. M. Sharp. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 73:237–244, 1988.

[7] T.H. Jukes and C.R. Cantor. Evolution of protein molecules. In H.N. Munro, editor, *Mammalian Protein Metabolism*, pages 21–132, New York, 1969. Academic Press.

[8] M. Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.*, 16:111–120, 1980.

[9] J. Kyte and R.F. Doolittle. A simple method for displaying the hydropathic character of a protein. *Journal of Molecular Biology*, 157:105–132, 1982.

[10] P. Legendre, Y. Desdevises, and E. Bazin. A statistical test for host-parasite coevolution. *Syst. Biol.*, 51:217–234, 2002.

[11] P. Mackiewicz, J. Zakrzewska-Czerwińska, A. Zawilak, M.R. Dudek, and S. Cebrat. Where does bacterial replication start? rules for predicting the *oriC* region. *Nucleic Acids Research*, 32:3781–3791, 2004.

[12] E. Paradis, J. Claude, and K. Strimmer. Ape: analyses of phylogenetics and evolution in R language. *Bioinformatics*, 20:289–290, 2004.

[13] W.R. Pearson and D.J. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America*, 85:2444–2448, 1988.

[14] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1984.

[15] I.M. Wallace, G. Blackshields, and D.G. Higgins. Multiple sequence alignments. *Curr. Opin. Struct. Biol.*, 15:261–266, 2005.