pairseqsim: Pairwise Sequence Similarity Simple

Witold Eryk Wolski

August 25, 2004

1 Introduction

The first fact of biological sequence analysis: In bio-molecular sequences (DNA, RNA, or amino acid sequences), high sequence similarity usually implies significant functional or structural similarity. The `pairseqsim` package can be used to compute pairwise sequence alignments of two amino acid sequences. Based on the optimal alignment of two sequences it computes also basic scores like sequence similarity, identity, and the alignment score. The optimal alignment can be visualized. A distance matrix storing all pairwise distances of all sequences in list can be computed. This matrix can be used to cluster the sequences using various clustering algorithms provided by **R**. To compute a biological meaningful alignment, a measure of amino acid similarity is needed. This information is provided by the families of BLOSUM or the PAM amino acid residue distance matrices. We imported many of this matrices from the EMBOSS software package (`http://www.hgmp.mrc.ac.uk/Software/EMBOSS/`. Gaps in an alignment are undesirable and thus penalized. The package implements an affine gap penalty:

$$\gamma(g) = d + (g-1)e \tag{1}$$

where d is the gap opening penalty and e is the gap extension penalty.

The package implements three types of alignment:

- Global Alignment.
- Local Alignment.
- Overlap Alignment.

Global alignment is applicable when we have two similar sequences that we want to align from end-to-end, e.g. two homologous genes from related species. To compute the global alignment the Needelman and Wunsch algorithm are used. For computing the *local alignment* the Smith Waterman algorithm are used. This type of alignment can be used if we would like to find the best match between subsequences of both. For example, we may want to find the position of a fragment of DNA in a genomic sequence. If we are given different fragments of genomic DNA that we would like to piece together, then we need an alignment method that does not penalize overhanging ends. Such an alignment can be computed by the *overlap alignment*.

2 Defining Sequences, Sequence Lists and Loading Data.

First you have to load the package with

```
> library(pairseqsim)
```

```
Creating a new generic function for "levels" in "pairseqsim"
Creating a new generic function for "frequency" in "pairseqsim"
Creating a new generic function for "summary" in "pairseqsim"
```

If we want to analyze more than one pair of sequences we need to define a list (`AASequencelist`) which stores our sequence objects.

```
> mySequlist <- new("AASequenceList", info = "my sequence list")
```

To the `AASequenceList` only an `AASequence` can be assigned. To define a new sequence we either use the `new` function or the function `AASequence`. The default constructor checks if the string is an amino acid sequence. This is done by comparing the letters in the sequence string to the amino acid alphabet which is defined by the class `AAAlphabet`. In the `AAAlphabet` class the characters which denote the amino acids are hard coded.

```
> new("AAAlphabet")
```

```
info :  AminoAcid
Alphabet :
 A R N D C Q E G H I L K M F P S T W Y V B Z X *
```

```
> pseq <- "MRTNPTTSNPEVSIREKKNLGRIAQIIGPVLDVAFPPGKMPNIYNALVVK"
> access <- "MyatpB"
> myseq1 <- new("AASequence", pseq, info = access)
> myseq1 <- AASequence(access, pseq)
```

There are many protein sequence database formats. In this package only a reader for the `FASTA` file format is implemented (`readFasta`). The `FASTA` file format looks in the following way.

```
>At1g01010 NAC domain protein, putative
MEDQVGFGFRPNDEELVGHYLRNKIEGNTSRDVEVAISEVNICSYDPWNLRFQSKYKSRD
AMWYFFSRRENNKGNRQSRTTVSGKWKLTGESVEVKDQWGFCSEGFRGKIGHKRVLVFLD
GRYPDKTKSDWVIHEFHYDLLPEHQRTYVICRLEYKGDDADILSAYAIDPTPAFVPNMTS
S*
```

Sequences in `FASTA` file format are preceded by a line starting with the symbol ">" as the first character. The rest of the line is the name and description of the sequence. The following lines contain the sequence data. To extract the "info" (id) string from the first line we define a function `infogrep` that extracts the appropriate substring from the line with the ">". We assign this function to the argument `grepinfo`.

```
> infogrep <- function(x) {
+     return(sub("^>([a-zA-Z0-9]+) .+", "\\1", x, perl = TRUE))
+ }
```

Additionally a function can be defined which pre-processes the amino acid sequence. In many sequence databases the end of the sequence is denoted by "*". The function `seqgrep` which we assign to the parameter `grepseq` will remove it from the end of the sequence string.

```
> seqgrep <- function(x) {
+     return(gsub("\\*", "", x))
+ }
```

The first argument to the function `readFasta` are a object of class `AASequenceList`, the second are the path to the file.

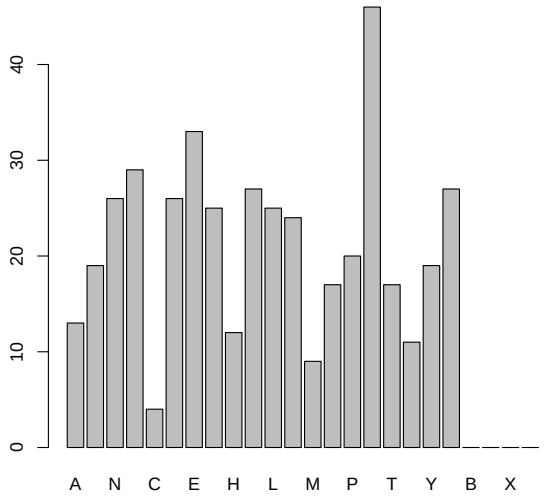
```
> mySequlist <- readFasta(mySequlist, "ex.fasta", grepinfo = infogrep,
+     grepseq = seqgrep)
> length(mySequlist)
```

```
[1] 66
```

3 Analysing the Protein sequences

To take a look at the amino acid composition of a protein sequence we use the function frequency.

```
> barplot(frequency(mySequlist[[1]]))
```



We also can compute a score if the sequence are aligned with itself. To compute the alignment we also need a similarity matrix to judge the similarity between two amino acid residues and to compute an optimal alignment of the sequence. In the data directory you will find many different similarity matrices from the Emboss package.

```
> data(EPAM110)
> selfalign(mySequlist[[2]], EPAM110)

  A
738
```

The class AASequence inherits from the R class character so you can use all the `R/base` functions defined for this class. E.g. to determine the length of the sequence we use the function `nchar`.

```
> nchar(mySequlist[[1]])

[1] 429
```

For pairwise comparison of sequences we use the function `salign`. The parameters which control the behavior of the function are:

- `delta` = Gap opening penalty, (default = -4)
- `gapext` = Gap extension penalty (default = delta)
- `alignment` = Type of the alignment, either "global", "local", "overlap" alignment.
- `scoring` = The return-type of the aligment. For two sequences it defaults to "AAAlignment".

The sequences in the sequence list can be accessed by their id. For two object of class `AASequence` the function `salign` returns by default an object of class "AAAlignment".

```
> str1 <- mySequlist[["At1g01580"]]
> str2 <- mySequlist[["At1g01590"]]
> as(str1, "character") <- substr(str1, 1, 90)
> as(str2, "character") <- substr(str2, 1, 140)
> res <- salign(str1, str2, EPAM110, delta = -4, gapext = -1, alignment = "global",
+     scoring = "AAAlignment")
> res

selfscore 1:  542 ; seq length 1 : 90
selfscore 2:  821 ; seq length 2 : 140
alig lenght:  153
score      :  188
FM         :  0.2818296
identity   :  45 / 90
similarity :  57 / 90
```

To see how the sequences where aligned use the function `summary`. The "|" denotes a identity of amino acid residues, the ":" denotes a similarity.

```
> summary(res)

selfscore 1:  542
selfscore 2:  821
alig lenght:  153
score      :  188
FM(score)  :  0.2818296
identity   :  45 / 90
similarity :  57 / 90
At1g01580 MEIEKSNNGGSNPSAGEEFKDMI-KGVTKFLMMVIFLGTIMLWIMMPTLTYRTKWLPHLRI
          |        |     ||  |::| | | ||||||| :|||::|||||| ||:  ||  :|
At1g01590 M--------G----VGEMNKEVIDK-VIKFLMMVILMGTIVIWIMMPTSTYKEIWLTSMRA

At1g01580 IKFGTSTYFGATGTTLFMYMFPMMVVA---C-----------------------------
           | | | |:|  |  | :|||||:::|   |
At1g01590 AKLGKSIYYGRPGVNLLVYMFPMILLAFLGCIYLHLKKSTTVNQFNSGVEKKRAKFGALRR

At1g01580 --------LGC----------------------
                  ||
At1g01590 RPMLVNGPLGIVTVTEVMFLTMFMALLLWSLAN
```

3.1 Scoring sequences

We use the function align to score a sequence against a list of sequences for example to find the most similar sequence. To rank the similarity of sequences we can use several scores.

- Based on the number of identities (matches of identical amino acid residues) in the optimal alignment we define the `identity` as:

$$\frac{\#\text{identity}}{\text{length of shorter sequence}} \tag{2}$$

- Based on the number of similarities (similar amino acid residues at the same position in the alignment) we define the `similarity` as:

$$\frac{\#\text{similarity}}{\text{length of shorter sequence}} \tag{3}$$

- The Smith Waterman, alignment `score` defined as:

$$S(A,B) = \sum_{k=1}^{N} m_{i_k j_k} - \text{gap penalties} \tag{4}$$

- The normalized Smith Waterman alignment score (`scoreN`) we define as the score of the alignment (if it is larger 0) divided by the smaller score of the alignment of the sequences with itself, zero otherwise.

$$\frac{\text{score}}{\min(\text{selfscore 1}, \text{selfscore 2})} \qquad \text{or 0 if score}<0 \tag{5}$$

- the `pozitive` score [1].

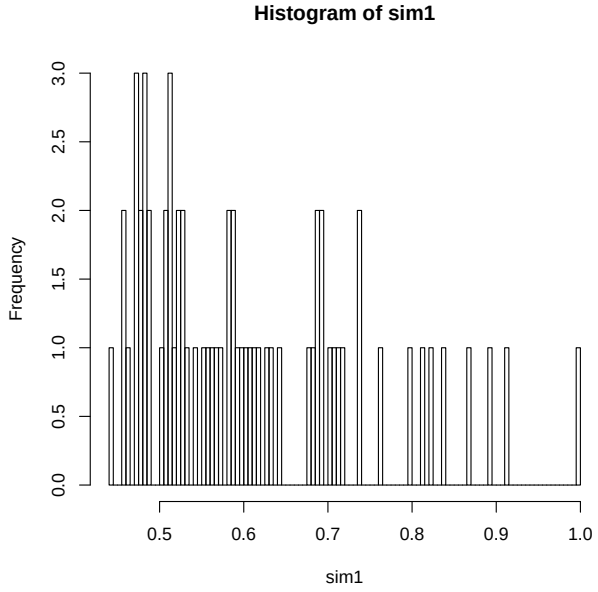
$$POZ(A,B) = \frac{S(A,B) - \mu_{perm}}{\sigma_{perm}} \tag{6}$$

To find a sequence in a list of sequences with the highest score to the bait sequence we use the function `salign` and `R/base` functions.

```
> sim1 <- salign(mySequlist, mySequlist[[1]], EPAM110, alignment = "global",
+     scoring = "similarity")
> which(sim1 == max(sim1))
```

```
At1g01010
        1
```

```
> hist(sim1, breaks = 100)
```



We also use the function `salign` to compute a distance matrix for all sequences in the list. The distance matrix can be used for example for clustering. To obtain the distance matrix we supply a `AASequenceList` as first parameter and `NULL` as second parameter to the function `saling`. In this case the scores defined above are converted in distances:

- 1-`similarity`, 1-`identity`, 1-`scoreN`
- Smith Waterman `score` first are converted into the Z score

$$Z(A,B) = \frac{S(A,B) - \text{mean}}{\text{standard deviation}} \tag{7}$$

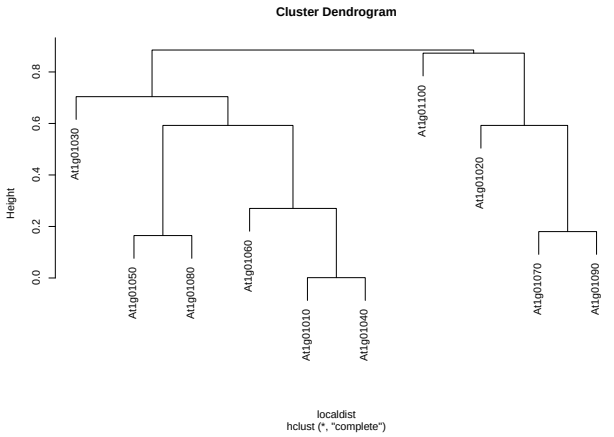
  Under the assumption that the Z scores for the sample set are normally distributed with *N(0,1)* [1] we compute the probability $P(X >= Z(A,B))$ that such or a higher score occur.

- The POZITIVE score is as well transformed in a p-value $P(X >= Z_{pozitive})$ with the assumption that X is normal distributed.

```
> globaldist <- salign(mySequlist[1:10], NULL, EPAM110, alignment = "global",
+     scoring = "score")
> localdist <- salign(mySequlist[1:10], NULL, EPAM110, alignment = "local",
+     scoring = "score")
```

The distance matrix returned by the function `salign` can be supplied to many clustering algorithms provided by `R` e.g. to the hierarchical clustering function `hclust` in the package `R/mva`.

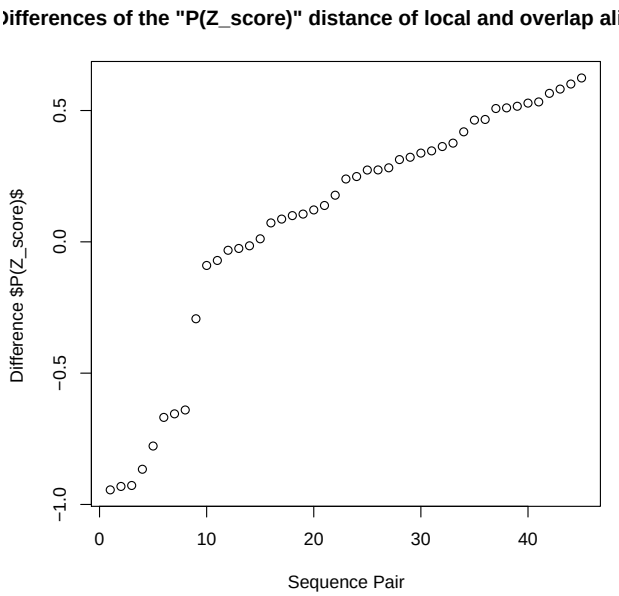
```
> plot(hclust(localdist))
```



It is also interesting to look at examples where the scores for different types of alignment highly differ. To do this the residues between the distance matrices can be computed and plotted

```
> plot(sort(as.numeric(localdist) - as.numeric(globaldist)), main = "Differences of the \"P(
+     xlab = "Sequence Pair", ylab = "Difference $P(Z_score)$ ")
```

[1] Its not[2]! We do it, not to derive a significance of the alignment, but to transform the SW-score (or the pozitive score) into a distance measure and to cluster the dataset.



We see that there are some sequence pairs which have different scores in the overlap and in the local alignment.

4 Summary

We are using the package `pairseqsim` to analyses results of Peptide Mass Fingerprint (PMF) identification experiments. By comparing the sequences of the highest scoring hits, we can draw conclusions about the significance of the identification result. Have fun and let me know if you find errors(!), have suggestions, your impressions, or for what you found useful this tool.

References

[1] Booth, H and Maindonald, J and Nielsen, O and Wilson, S Normalizing sequence alignment scores for composition bias *Recomb 2003 - Berlin*

[2] Waterman, M. S. and Vingron, M. 1994. Sequence comparison significance and Poisson approximation. *Statistical Science*`9:367-381.`