odseq package vignette

Mantainer: jose@jimenezluna.com

January 2016

odseq is a very simple technique for outlier detection in multiple sequence alignments. It depends on package msa for performing first the alignment, therefore you can choose a variety of algorithms for doing so: Clustal Omega or MUSCLE among others, check the msa documentation for more info.

Typically, one would like to discard those sequences that cause the alignment to have a poor overall performance. These we call outliers in a statistical sense: a sequence that does not share as much information as the others when aligned to the rest may be considered superfluous. The odseq function takes an object of class MsaAAMultipleAlignments and other arguments, and returns a logical vector indicating the outliers given a threshold. Depending on your problem, one may need to tune this parameter, but in our simulations, this parameter has been proven to be sufficiently robust.

Aligned sequences

Let's start with a simple example: Imagine that we have a bunch of sequences, but among these we know that some of them may not share the same pattern as the others. To replicate this, we get 211 sequences from a certain PFAM family PF09274 and we add 100 random sequences for noise that we would like the algorithm to detect.

Load the library and data, which is provided as an example dataset with the package:

```
set.seed(93)
library(odseq)
data("seqs")
```

Perform a multiple sequence alignment using Clustal Omega with default parameters, for example:

```
library(msa)
```

```
## Loading required package: Biostrings
## Loading required package: BiocGenerics
## Loading required package: parallel
##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
## The following objects are masked from 'package:stats':
##
##     IQR, mad, xtabs
## The following objects are masked from 'package:base':
##
##     Filter, Find, Map, Position, Reduce, anyDuplicated, append,
##     as.data.frame, cbind, colnames, do.call, duplicated, eval,
```

```
##     evalq, get, grep, grepl, intersect, is.unsorted, lapply,
##     lengths, mapply, match, mget, order, paste, pmax, pmax.int,
##     pmin, pmin.int, rank, rbind, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which, which.max,
##     which.min
```

```
## Loading required package: S4Vectors
```

```
## Loading required package: stats4
```

```
##
## Attaching package: 'S4Vectors'
```

```
## The following objects are masked from 'package:base':
##
##     colMeans, colSums, expand.grid, rowMeans, rowSums
```

```
## Loading required package: IRanges
```

```
## Loading required package: XVector
```

```
alig <- msa(seqs)
```

```
## use default substitution matrix
```

Once this is done, just call `odseq`, and specify parameters. Check the documentation for more information on these.

```
ground_values <- c(rep(FALSE, 211), rep(TRUE, 100))
outliers <- odseq(alig, distance_metric = "affine", B = 1000)
```

Well, time to check how our algorithm has performed:

```
mean(outliers == ground_values)
```

```
## [1] 1
```

Apparently, our algorithm makes perfect prediction of outliers values using these sequences.

Unaligned sequences

Package `odseq` also includes function `odseq_unaligned`, for situations where computing a multiple alignment among a large number of sequences is computationally expensive. However, it requires a similarity or distance matrix to be computed before performing outlier detection. This can also be computationally expensive although efficient algorithms exist to compute approximations of similarity matrices, but for the sake of examples, let's assume that this is not a problem. We can use package `kebabs` functionality to compute a similarity matrix using string kernels among sequences.

For example, let's use the well known spectrum kernel:

```
library(kebabs)
```

```
## Loading required package: kernlab
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:Biostrings':
##
##     type
```

```
sp <- spectrumKernel(k = 3)
mat <- getKernelMatrix(sp, seqs)
```

Then call `odseq_unaligned` on this matrix, and specify if it is a distance or similarity metric:

```
outliers_unaligned <- odseq_unaligned(mat, B = 1000, type = "similarity")
```

Let's check accuracy again:

```
mean(outliers_unaligned == ground_values)
```

```
## [1] 0.7749196
```

As you can see, precision is heavily influenced by the fact that the sequences were not aligned before. `odseq_unaligned` is also very dependent on the type of metric considered. The spectrum kernel is very simple and does not take into account the relative position of each letter. Therefore, using other types of string kernels (or distance matrices in general) may improve performance consistently.