Using SynExtend

Nicholas Cooley

2022-04-26

Contents

1 Introduction 2

2 Package Structure 2

2.1 Installation 2

3 Usage 2

1 Introduction

`SynExtend` is a package of tools for working with objects of class `Synteny` built from the package `DECIPHER`'s `FindSynteny()` function.

Synteny maps provide a powerful tool for quantifying and visualizing where pairs of genomes share order. Typically these maps are built from predictions of orthologous pairs, where groups of pairs that provide contiguous and sequential blocks in their respective genomes are deemed a 'syntenic block'. That designation of synteny can then used to further interrogate the predicted orthologs themselves, or query topics like genomic rearrangements or ancestor genome reconstruction.

`FindSynteny` takes a different approach, finding exactly matched shared k-mers and determining where shared k-mers, or blocks of proximate shared k-mers are significant. Combining the information generated by `FindSynteny` with locations of genomic features allows us to simply mark where features are linked by syntenic k-mers. These linked features represent potential orthologous pairs, and can be easily evaluated on the basis of the k-mers that they share, or alignment.

2 Package Structure

Currently `SynExtend` contains one set of functions for performig orthology predictions, as well as a rearrangement estimation function that is currently under construction.

2.1 Installation

1. Install the latest version of R using CRAN.
2. Install `SynExtend` in R by running the following commands:

```
if (!requireNamespace("BiocManager",
                      quietly = TRUE)) {
  install.packages("BiocManager")
}
BiocManager::install("SynExtend")
```

3 Usage

Using the `FindSynteny` function in `DECIPHER` build an object of class `Synteny`. In this tutorial, a prebuilt `DECIPHER` database is used. For database construction see `?Seqs2DB` in `DECIPHER`. This example starts with a database containing three endosymbiont genomes that were chosen to keep package data to a minimum.

```
library(SynExtend)
## Loading required package: DECIPHER
## Loading required package: Biostrings
## Loading required package: BiocGenerics
##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:stats':
##
```

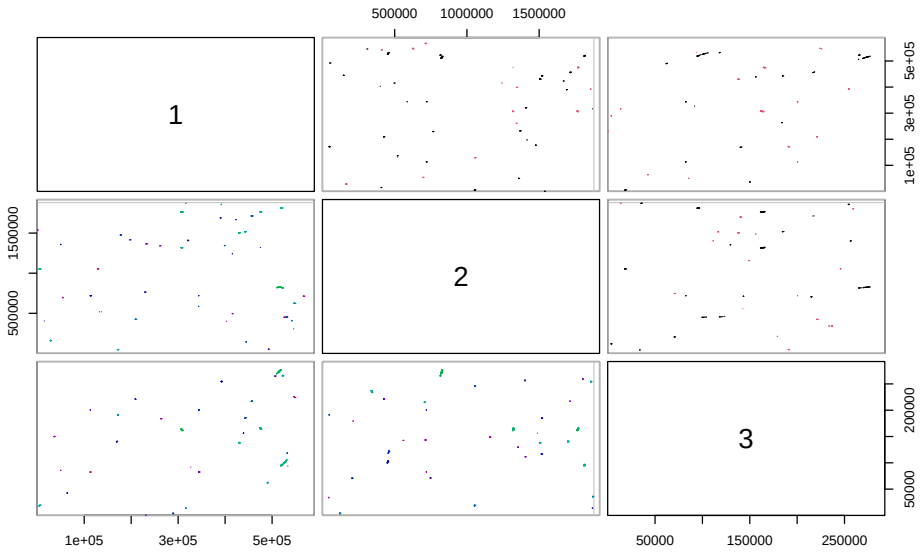
```
##     IQR, mad, sd, var, xtabs
## The following objects are masked from 'package:base':
##
##     Filter, Find, Map, Position, Reduce, anyDuplicated, append,
##     as.data.frame, basename, cbind, colnames, dirname, do.call,
##     duplicated, eval, evalq, get, grep, grepl, intersect, is.unsorted,
##     lapply, mapply, match, mget, order, paste, pmax, pmax.int, pmin,
##     pmin.int, rank, rbind, rownames, sapply, setdiff, sort, table,
##     tapply, union, unique, unsplit, which.max, which.min
## Loading required package: S4Vectors
## Loading required package: stats4
##
## Attaching package: 'S4Vectors'
## The following objects are masked from 'package:base':
##
##     I, expand.grid, unname
## Loading required package: IRanges
## Loading required package: XVector
## Loading required package: GenomeInfoDb
##
## Attaching package: 'Biostrings'
## The following object is masked from 'package:base':
##
##     strsplit
## Loading required package: RSQLite
## Loading required package: parallel

DBPATH <- system.file("extdata",
                      "Endosymbionts.sqlite",
                      package = "SynExtend")

Syn <- FindSynteny(dbFile = DBPATH)
## ================================================================================
##
## Time difference of 4.24 secs
```

Synteny maps represent where genomes share order. Simply printing a synteny object to the console displays a gross level view of the data inside. Objects of class `Synteny` can also be plotted to provide clear visual representations of the data inside. The genomes used in this example are distantly related and fairly dissimilar.

```
Syn
##           1         2         3
## 1     1 seq 1.4% hits   3% hits
## 2 46 blocks    2 seqs 3.7% hits
## 3 31 blocks 36 blocks     1 seq
pairs(Syn)
```



Data present inside objects of class `Synteny` can also be accessed relatively easily. The object itself is functionally a matrix of lists, with data describing exactly matched k-mers present in the upper triangle, and data describing blocks of chained k-mers in the lower triangle. For more information see `?FindSynteny` in the package `DECIPHER`.

```
print(head(Syn[[1, 2]]))
##      index1 index2 strand width start1 start2 frame1 frame2
## [1,]      1      1      0    62 510107 820531      1      3
## [2,]      1      1      0    27 510220 820644      1      3
## [3,]      1      1      0    27 511315 821517      1      3
## [4,]      1      1      0    33 511477 821676      1      3
## [5,]      1      1      0    30 511837 822042      1      3
## [6,]      1      1      0    27 512044 822252      1      3
print(head(Syn[[2, 1]]))
##      index1 index2 strand score start1  start2   end1    end2 first_hit
## [1,]      1      1      0   323 510107  820531 518022  827923         1
## [2,]      1      1      0   237 521728  815602 522899  816821        24
## [3,]      1      1      0   147 429274 1502877 430720 1504314        35
## [4,]      1      1      0   135 518406 1810735 521309 1813715        49
## [5,]      1      1      0   135   4142 1051916   6221 1054103        59
## [6,]      1      1      0   117 171167   46819 173077   49020        71
##      last_hit
## [1,]       23
## [2,]       34
## [3,]       48
## [4,]       58
## [5,]       70
## [6,]       78
```

The above printed objects show the data for the comparison between the first and second genome in our database.

To take advantage of these synteny maps, we can then overlay the gene calls for each genome present on top of our map.

Next, GFF annotations for the associated genomes are parsed to provide gene calls in a use-able format. GFFs are not the only possible source of appropriate gene calls, but they are the source that was used during package construction and testing. Parsed GFFs can be constructed with `gffToDataFrame`, for full functionality, or GFFs can be imported via `rtracklater::import()` for limited functionality. GeneCalls for both the `PairSummaries` and `NucleotideOverlap` functions must be named list, and those names must match `dimnames(Syn)[[1]]`.

```
# generating genecalls with local data:
GC <- gffToDataFrame(GFF = system.file("extdata",
                                       "GCF_021065005.1_ASM2106500v1_genomic.gff.gz",
                                       package = "SynExtend"),
                     Verbose = TRUE)
## ================================================================================
## Time difference of 6.029062 secs

# in an effort to be space conscious, not all original gffs are kept within this package
GeneCalls <- get(data("Endosymbionts_GeneCalls", package = "SynExtend"))
```

`SynExtend`'s `gffToDataFrame` function will directly import gff files into a usable format, and includes other extracted information.

```
print(head(GeneCalls[[1]]))
## DataFrame with 6 rows and 11 columns
##       Index    Strand     Start      Stop        Type                ID
##   <integer> <integer> <integer> <integer> <character>       <character>
## 1         1         0         1      1362        gene gene-Gromo_RS00005
## 2         1         0      1652      2740        gene gene-Gromo_RS00010
## 3         1         0      2745      3905        gene gene-Gromo_RS00015
## 4         1         0      4016      6400        gene gene-Gromo_RS00020
## 5         1         0      6496      8598        gene gene-Gromo_RS00025
## 6         1         0      8722      9822        gene gene-Gromo_RS00030
##           Range               Product    Coding Translation_Table
##   <IRangesList>           <character> <logical>       <character>
## 1        1-1362 chromosomal replicat..      TRUE                11
## 2     1652-2740 DNA polymerase III s..      TRUE                11
## 3     2745-3905      AAA family ATPase      TRUE                11
## 4     4016-6400 DNA topoisomerase (A..      TRUE                11
## 5     6496-8598   hypothetical protein      TRUE                11
## 6     8722-9822        MFS transporter      TRUE                11
##          Contig
##     <character>
## 1 NZ_CP069247.1
## 2 NZ_CP069247.1
## 3 NZ_CP069247.1
## 4 NZ_CP069247.1
## 5 NZ_CP069247.1
## 6 NZ_CP069247.1
```

Raw GFF imports are also acceptable, but prevent alignments in amino acid space with `PairSummaries()`.

```
X01 <- rtracklayer::import(system.file("extdata",
                                       "GCA_000875775.1_ASM87577v1_genomic.gff.gz",
                                       package = "SynExtend"))
class(X01)
print(X01)
```

`SynExtend`'s primary functions provide a way to identify where pairs of genes are explicitly linked by syntenic hits, and then summarize those links. The first step is just identifying those links.

```
Links <- NucleotideOverlap(SyntenyObject = Syn,
                           GeneCalls = GeneCalls,
                           Verbose = TRUE)
##
## Reconciling genecalls.
## ================================================================================
## Finding connected features.
## ================================================================================
## Time difference of 0.2624192 secs
```

The `Links` object generated by NucleotideOverlap is a raw representation of positions on the synteny map where shared k-mers link genes between paired genomes. As such, it is analagous in shape to objects of class `Synteny`. This raw object is unlikely to be useful to most users, but has been left exposed to ensure that this data remains accessible should a user desire to have access to it.

```
class(Links)
## [1] "LinkedPairs"
print(Links)
##           1         2        3
## 1  1 Contig  55 Pairs 49 Pairs
## 2 238 Kmers 2 Contigs 57 Pairs
## 3 296 Kmers 337 Kmers 1 Contig
```

This raw data can be processed to provide a straightforward summary of predicted pairs.

```
LinkedPairs1 <- PairSummaries(SyntenyLinks = Links,
                              DBPATH = DBPATH,
                              PIDs = FALSE,
                              Verbose = TRUE)
##
## Preparing overhead data.
## Overhead complete.
## Collecting pairs.
## ================================================================================
## Time difference of 1.657217 secs
```

The object `LinkedPairs1` is a data.frame where each row is populated by information about a predicted orthologous pair. By default `PairSummaries` uses a simple model to determine whether the k-mers that link a pair of genes are likely to provide an erroneous link. When set to `Model = "Global"`, is is simply a prediction of whether the involved nucleotides are likely

to describe a pair of genomic features whose alignment would result in a PID that falls within a random distribution. This model is effective if somewhat permissive, but is significantly faster than performing many pairwise alignments.

```
print(head(LinkedPairs1))
##       p1       p2 ExactMatch Consensus TotalKmers MaxKmer p1FeatureLength
## 1  1_1_1 2_1_1682         63 0.9867718          2      33            1362
## 2  1_1_4 2_1_1146        397 0.9918857         12      60            2385
## 3 1_1_13  2_1_426         29 0.9957898          1      29              82
## 4 1_1_23  2_1_156        330 0.9975095         10      45            1068
## 5 1_1_40 2_1_1492        131 0.9908460          4      41            2385
## 6 1_1_43  2_1_769         87 0.9881375          3      32            1275
##   p2FeatureLength Adjacent    TetDist PIDType PredictedPID
## 1            1344        0 0.04894316      AA    0.4081521
## 2            2484        0 0.03564127      AA    0.5693577
## 3              84        0 0.11731328      NT    0.5568290
## 4            1053        0 0.05201223      AA    0.6978996
## 5            2448        0 0.04585398      AA    0.3463909
## 6            1347        0 0.04718792      AA    0.4450279
```

PairSummaries includes arguments that allow for aligning all pairs that are predicted, via `PIDs = TRUE`, while `IgnoreDefaultStringSet = FALSE` indicates that alignments should be performed in nucleotide or amino acid space as is appropriate for the linked sequences. Setting `IgnoreDefaultStringSet = TRUE` will force all alignments into nucleotide space.

As of SynExtend v 1.3.13, the functions `ExtractBy` and `DisjointSet` have been added to provide users with direct tools to work with `PairSummaries` objects.

```
SingleLinkageClusters <- DisjointSet(Pairs = LinkedPairs1,
                                     Verbose = TRUE)
##
## Assigning initial root:
## ================================================================================
## Time difference of 0.002269983 secs
##
## Assigning final root:
##

==================================================================================
## Time difference of 0.001950026 secs
##
## Assigning single linkage clusters.
## Assignments complete.
##
## Time difference of 0.007353067 secs
```

```
# extract the first 10 clusters
Sets <- ExtractBy(x = LinkedPairs1,
                  y = DBPATH,
                  z = SingleLinkageClusters[1:10],
                  Verbose = TRUE)
##
## Extracting Sequences:
```

```
## ================================================================================
##
## Arranging Sequences:
## ================================================================================
##
## Time difference of 0.1853368 secs

head(Sets)
## [[1]]
## DNAStringSet object of length 2:
##     width seq                                               names
## [1]  1362 ATGAGTAGTGAAATTTCTAATGT...TTAAAGCAAAATTGATGAAATAA 1_1_1
## [2]  1344 ATGTTGACAAAAACCCCGCAAGA...CTAGACGCAATTTAGAAGCTTAA 2_1_1682
##
## [[2]]
## DNAStringSet object of length 3:
##     width seq                                               names
## [1]  2385 ATGGATTCAGACGATATTAAGAA...ATGTGAGAAATTTAGATGTATAA 1_1_4
## [2]  2484 ATGACTACAATGACCAAAGAAGA...CGGTTAAGAATTTGGATATTTAA 2_1_1146
## [3]  2385 ATGATTAATTATGATTCTTCGAA...CAAATTTTGACTTAGATTTTTAA 3_1_28
##
## [[3]]
## DNAStringSet object of length 2:
##     width seq                                               names
## [1]    82 GCCCAGGTGGCGGAATGGTAGAC...AGGTTCAAGTCCTCTTCTGGGTA 1_1_13
## [2]    84 GCGGTCATGGCGGAATTGGTAGA...ATGTTCGAGTCATCTTGACCGCA 2_1_426
##
## [[4]]
## DNAStringSet object of length 2:
##     width seq                                               names
## [1]  1068 TTGAATAAGAATCAAAATGATGA...AAGATGTACTTGAAGATGTATAA 1_1_23
## [2]  1053 ATGGCTCAACAAAATGATGCAGG...TAAAAGAAGAGGCCTTGGTTTAA 2_1_156
##
## [[5]]
## DNAStringSet object of length 3:
##     width seq                                               names
## [1]  2385 ATGTGCGCAAGAGACTTGGATGG...GCTTTAAATTGGAGAAGATATAA 1_1_40
## [2]  2448 TTGGCCAAGAATTTAAAAATAAA...CCTATTTAGACCAAGAGCTATAA 2_1_1492
## [3]  1500 ATGTCGTATAAAGAAAAAAAAAT...TTAAATCATATATTCAAGAATGA 3_1_100
##
## [[6]]
## DNAStringSet object of length 2:
##     width seq                                               names
## [1]  1275 ATGTCCGATGTATTGCTATCAAT...GTATGCGTGGAGAAATAACTTGA 1_1_43
## [2]  1347 ATGCGTAAACAACAAGTAAAAAC...GTCTTAAGAGTTTTAGCCATTAA 2_1_769
```

Session Info:

```
sessionInfo()
## R version 4.2.0 RC (2022-04-19 r82224)
## Platform: x86_64-pc-linux-gnu (64-bit)
```

```
## Running under: Ubuntu 20.04.4 LTS
##
## Matrix products: default
## BLAS:   /home/biocbuild/bbs-3.15-bioc/R/lib/libRblas.so
## LAPACK: /home/biocbuild/bbs-3.15-bioc/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_GB              LC_COLLATE=C
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] SynExtend_1.8.0     DECIPHER_2.24.0     RSQLite_2.2.12
##  [4] Biostrings_2.64.0   GenomeInfoDb_1.32.0 XVector_0.36.0
##  [7] IRanges_2.30.0      S4Vectors_0.34.0    BiocGenerics_0.42.0
## [10] BiocStyle_2.24.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.8.3           compiler_4.2.0         BiocManager_1.30.17
##  [4] bitops_1.0-7           tools_4.2.0            zlibbioc_1.42.0
##  [7] digest_0.6.29          bit_4.0.4              evaluate_0.15
## [10] memoise_2.0.1          pkgconfig_2.0.3        rlang_1.0.2
## [13] cli_3.3.0              DBI_1.1.2              yaml_2.3.5
## [16] xfun_0.30              fastmap_1.1.0          GenomeInfoDbData_1.2.8
## [19] stringr_1.4.0          knitr_1.38             vctrs_0.4.1
## [22] bit64_4.0.5            rmarkdown_2.14         bookdown_0.26
## [25] blob_1.2.3             magrittr_2.0.3         htmltools_0.5.2
## [28] stringi_1.7.6          RCurl_1.98-1.6         cachem_1.0.6
## [31] crayon_1.5.1
```