Using SynExtend

Nicholas Cooley

2021-05-30

Contents

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 interogate 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 proximal 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 syntenic k-mers that they share, or alignment.

2 Package Structure

Currently `SynExtend` contains only one set of functions, but will be expanded in the future.

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 archaea genomes: Nitrosopumilus adriaticus, Nitrosopumilus piranensis, and a Candidatus Nitrosopumilus.

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

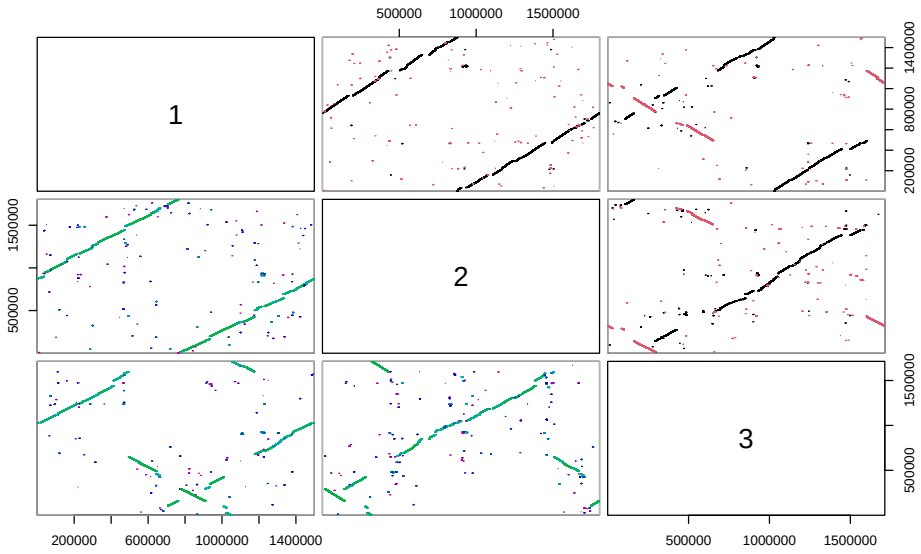
```
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
## The following objects 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
##
## Attaching package: 'IRanges'
## The following object is masked from 'package:grDevices':
##
##     windows
## Loading required package: XVector
## Loading required package: GenomeInfoDb
##
## Attaching package: 'Biostrings'
## The following object is masked from 'package:base':
##
##     strsplit
## Loading required package: RSQLite

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

Syn <- FindSynteny(dbFile = DBPATH)
## ================================================================================
##
## Time difference of 8.5 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 clear visual representations of the data inside. The genomes used in this example are all from the same genus, and should be expected to be somewhat closely related.

```
Syn
##            1          2        3
## 1      1 seq   43% hits 58% hits
## 2 177 blocks      1 seq 39% hits
## 3 133 blocks 207 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    18 932191 197117      0      0
## [2,]      1      1      0    14 932212 197138      0      0
## [3,]      1      1      0    45 932268 197194      3      1
## [4,]      1      1      0    17 932350 197276      0      0
## [5,]      1      1      0    20 932377 197303      0      0
## [6,]      1      1      0    24 932416 197342      1      2
print(head(Syn[[2, 1]]))
##      index1 index2 strand score start1  start2    end1    end2 first_hit
## [1,]      1      1      0 40740 932191  197117 1091978  359925         1
## [2,]      1      1      0 34439 495594 1507344  623248 1649448      1975
## [3,]      1      1      0 31996 771868   13735  918407  154427      3561
## [4,]      1      1      0 16601 328609 1298395  463732 1437247      5212
## [5,]      1      1      0 14806 705863 1746189  760642 1799647      6849
## [6,]      1      1      0 14003  40177  943557  155993 1070853      7482
##      last_hit
## [1,]     1974
## [2,]     3560
## [3,]     5211
## [4,]     6848
## [5,]     7481
```

```
## [6,]     8694
```

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]]`.

```
GeneCalls <- vector(mode = "list",
                    length = ncol(Syn))

GeneCalls[[1L]] <- gffToDataFrame(GFF = system.file("extdata",
                                                    "GCA_006740685.1_ASM674068v1_genomic.gff.gz",
                                                    package = "SynExtend"),
                                  Verbose = TRUE)
## ================================================================================
## Time difference of 23.0002 secs
GeneCalls[[2L]] <- gffToDataFrame(GFF = system.file("extdata",
                                                    "GCA_000956175.1_ASM95617v1_genomic.gff.gz",
                                                    package = "SynExtend"),
                                  Verbose = TRUE)
## ================================================================================
## Time difference of 27.46891 secs
GeneCalls[[3L]] <- gffToDataFrame(GFF = system.file("extdata",
                                                    "GCA_000875775.1_ASM87577v1_genomic.gff.gz",
                                                    package = "SynExtend"),
                                  Verbose = TRUE)
## ================================================================================
## Time difference of 27.68774 secs

names(GeneCalls) <- seq(length(GeneCalls))
```

`SynExtend`'s `gffToDataFrame` function will directly import gff files into a useable 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         1       307       621        gene gene-Nisw_00010
## 2         1         1       673      1182        gene gene-Nisw_00015
## 3         1         0      1271      1621        gene gene-Nisw_00020
## 4         1         1      1603      1914        gene gene-Nisw_00025
## 5         1         0      2013      2225        gene gene-Nisw_00030
## 6         1         1      2222      3313        gene gene-Nisw_00035
```

```
##           Range               Product    Coding Translation_Table      Contig
##   <IRangesList>           <character> <logical>       <character> <character>
## 1       307-621    DNA-binding protein      TRUE                11  CP035425.1
## 2      673-1182 DNA-directed RNA pol..      TRUE                11  CP035425.1
## 3     1271-1621   hypothetical protein      TRUE                11  CP035425.1
## 4     1603-1914 MarR family transcri..      TRUE                11  CP035425.1
## 5     2013-2225   hypothetical protein      TRUE                11  CP035425.1
## 6     2222-3313 deoxyhypusine synthase      TRUE                11  CP035425.1
```

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,
                           LimitIndex = FALSE,
                           Verbose = TRUE)
##
## Reconciling genecalls.
## ================================================================================
## Finding connected features.
## ================================================================================
## Time difference of 1.796854 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  1740 Pairs 1807 Pairs
## 2 14814 Kmers    1 Contig 1840 Pairs
## 3 16722 Kmers 15127 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 27.73458 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 Concensus TotalKmers MaxKmer p1FeatureLength
## 1 1_1_1 2_1_1080          1 0.9931940          1       1             315
## 2 1_1_1 2_1_1081        178 0.9878461          6      57             315
## 3 1_1_2 2_1_1082        237 0.9734190          8      60             510
## 4 1_1_3 2_1_1083        252 0.9612734          8      69             351
## 5 1_1_4 2_1_1083          9 0.9651535          1       9             312
## 6 1_1_4 2_1_1084        134 1.0000000          6      30             312
##   p2FeatureLength Adjacent    TetDist PIDType PredictedPID
## 1             417        1 0.07524152      AA    0.4152832
## 2             309        1 0.06476061      AA    0.7832442
## 3             543        2 0.04352695      AA    0.8178053
## 4             387        2 0.05609274      AA    0.8442963
## 5             387        2 0.09458148      AA    0.4622422
## 6             312        2 0.07030926      AA    0.7950427
```

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 0.04687595 secs
##
## Assigning final root:
##


==================================================================================
## Time difference of 0.03125 secs
```

```
##
## Assigning single linkage clusters.
## ================================================================================
## Time difference of 0.4374158 secs
```

```
SeqsByCluster <- ExtractBy(x = LinkedPairs1,
                           y = SingleLinkageClusters,
                           Method = "clusters",
                           DBPATH = DBPATH,
                           Verbose = TRUE)
##
## Preparing overhead data.
## Overhead complete.
## ================================================================================
## Time difference of 2.698197 mins

head(SeqsByCluster)
## [[1]]
## AAStringSet object of length 6:
##     width seq                                                 names
## [1]   105 MLMDETREPHGQEQTKKSDETIA...DTDAVAGIGRMTSTIEIVLVKI* 1_1_1
## [2]   139 MLLPAEIESKTLIPALRAILAKK...NIDEQVCKECENMLLKGPGSVY* 1_1_1794
## [3]   139 MLLPAEIESKTLIPALRAILAKK...NIDEQVCKECENMLLKGPGSVY* 2_1_1080
## [4]   103 MEETTEPYGQEQTEKSEGTIIHI...DTAEAPGIGSMTSTIEIILNKI* 2_1_1081
## [5]   139 MLLPAEIESKTLIPALRAILAKK...DIDEQVCKECENMLLKGPGSVY* 3_1_1268
## [6]   103 MEETSEPYGQEQTKKSDEAIIHI...DTENAAGIGRMTSTIEIILIKN* 3_1_1269
##
## [[2]]
## AAStringSet object of length 3:
##     width seq                                                 names
## [1]   170 MSDANNTEVVEVAEDEIPATEEI...ITIRRVLPNGDYQNIPIDYFEK* 1_1_2
## [2]   181 MIRLYNVLRKFCYLSDVKEAPLV...ITIRRVLPNGDFQNIPIDYFEK* 2_1_1082
## [3]   183 MIRLYNVLRKNRYLSDANKTKVV...ITIRRVLPNGDYQNIPIDYFEK* 3_1_1270
##
## [[3]]
## AAStringSet object of length 6:
##     width seq                                                 names
## [1]   117 MEIRGKAKICCDLKRHLSPRTVG...DNIDALKDVKSGDVLCIYEETA* 1_1_3
## [2]   104 MGGAKKPTVAKKDTSSGSKESKK...EKGIVKRVGGYSGHHLYQAVSS* 1_1_4
## [3]   129 MSTSSVSRKQLILEIRGKAKISC...GDIDALKNVKSGDVLCIYEETA* 2_1_1083
## [4]   104 MGGAKKPTAANKDKSAGSKDTKK...IKGTVKRVGGYSGHHLYQAVSS* 2_1_1084
## [5]   129 MSTASVSRKQLILEIKGKAKISC...ENIDALKDVKSGDVLCIYEETA* 3_1_1271
## [6]   104 MGGAKKPTAAKKDTSSNTKDSKK...EKGIVKKVGGYSGHHLYQAVSS* 3_1_1272
##
## [[4]]
## AAStringSet object of length 6:
##     width seq                                                 names
## [1]    71 MGVVSKGAKCNVDGCDNDGARSL...YKEYKKESKDDRDLERARFDKF* 1_1_5
## [2]   364 VDPHKFHGKDIPHIKLDPKMTIE...YEKLSNDYFKNPVNKKRSKKKN* 1_1_6
## [3]    71 MGIVSKGAKCNVDGCDQDGARSL...YKEWKKESKDDRDLERARFDKF* 2_1_1085
## [4]   354 VDPHEFHGKDIPHIKLDPNMTIE...KRLYKKLDKLYEKLREDYSKNP* 2_1_1086
```

```
## [5]    71 MGVVSKGAKCNVDGCDKDGARSL...YKEYKKESKDDRELERARYDRF* 3_1_1273
## [6]   366 MVVDPHKFHGKDIPHIKLDPKMT...YEKLSKDYFKNPVKKKRVKKKN* 3_1_1274
##
## [[5]]
## AAStringSet object of length 3:
##     width seq                                               names
## [1]   112 MDIIDLHDPQRVNRSPDAVEILL...TSNLGISGLILRSVISLRGKLE* 1_1_7
## [2]   112 MKIIDLHDPQRVDKSPDDVEILM...ISNLGISALILRSIICLREKLD* 2_1_1087
## [3]   113 MMDIVDLHDPQRVNRIPDKTKVL...TSNLGISGLILRSVISLRGKTS* 3_1_1275
##
## [[6]]
## AAStringSet object of length 3:
##     width seq                                               names
## [1]   623 MRILQLHCDSIEYTPTKKEIKSA...SGKPFTGLNQSSHLSKRPQLMV* 1_1_12
## [2]   621 MRILQLHCDSIEYTPTKKEIKSA...KEKPFTGLNLPKYLSKRPQLMV* 2_1_1094
## [3]   622 MRILQLHCDSIEYTPTKKEIKSA...SGKPFTGLNQAMHLSKRPQLMV* 3_1_1276
```

Session Info:

```
sessionInfo()
## R version 4.1.0 (2021-05-18)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows Server 2012 R2 x64 (build 9600)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=C
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] SynExtend_1.4.1     DECIPHER_2.20.0     RSQLite_2.2.7
##  [4] Biostrings_2.60.0   GenomeInfoDb_1.28.0 XVector_0.32.0
##  [7] IRanges_2.26.0      S4Vectors_0.30.0    BiocGenerics_0.38.0
## [10] BiocStyle_2.20.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.6             knitr_1.33             magrittr_2.0.1
##  [4] zlibbioc_1.38.0        bit_4.0.4              rlang_0.4.11
##  [7] fastmap_1.1.0          blob_1.2.1             stringr_1.4.0
## [10] tools_4.1.0            xfun_0.23              DBI_1.1.1
## [13] htmltools_0.5.1.1      bit64_4.0.5            yaml_2.2.1
## [16] digest_0.6.27          crayon_1.4.1           bookdown_0.22
## [19] GenomeInfoDbData_1.2.6 BiocManager_1.30.15    vctrs_0.3.8
```

```
## [22] bitops_1.0-7          RCurl_1.98-1.3        cachem_1.0.5
## [25] memoise_2.0.0         evaluate_0.14         rmarkdown_2.8
## [28] stringi_1.6.2         compiler_4.1.0        pkgconfig_2.0.3
```