Package ‘SynExtend’

April 12, 2022

Type Package

Title Tools for Working With Synteny Objects

Version 1.6.0

biocViews Genetics, Clustering, ComparativeGenomics, DataImport

Description Shared order between genomic sequences provide a great deal of information. Synteny objects produced by the R package DECIPHER provides quantitative information about that shared order. SynExtend provides tools for extracting information from Synteny objects.

Depends R (>= 4.1.0), DECIPHER (>= 2.20.0)

Imports methods, Biostrings, S4Vectors, IRanges, utils, stats

Suggests BiocStyle, knitr, rtracklayer, igraph, markdown, rmarkdown

License GPL-3

Encoding UTF-8

NeedsCompilation no

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/SynExtend

git_branch RELEASE_3_14

git_last_commit 8d9293e

git_last_commit_date 2021-10-26

Date/Publication 2022-04-12

Author Nicholas Cooley [aut, cre] (<https://orcid.org/0000-0002-6029-304X>),
Adelle Fernando [ctb],
Erik Wright [aut]

Maintainer Nicholas Cooley <npc19@pitt.edu>

R topics documented:

DisjointSet — *Return single linkage clusters from* `PairSummaries` *objects.*

Description

Takes in a `PairSummaries` object and return a list of identifiers organized into single linkage clusters.

Usage

```
DisjointSet(Pairs,
            Verbose = FALSE)
```

Arguments

`Pairs`	A `PairSummaries` object.
`Verbose`	Logical indicating whether to print progress bars and messages. Defaults to `FALSE`.

Details

Takes in a `PairSummaries` object and return a list of identifiers organized into single linkage clusters.

Value

Returns a list of character vectors representing IDs of sequence features, typically genes.

Author(s)

Nicholas Cooley <`npc19@pitt.edu`>

See Also

`FindSynteny`, `Synteny-class`, `PairSummaries`, `FindSets`

Examples

```
DBPATH <- system.file("extdata",
                      "VignetteSeqs.sqlite",
                      package = "SynExtend")
Syn <- FindSynteny(dbFile = DBPATH)
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)
GeneCalls[[2L]] <- gffToDataFrame(GFF = system.file("extdata",
                                           "GCA_000956175.1_ASM95617v1_genomic.gff.gz",
                                                   package = "SynExtend"),
                                  Verbose = TRUE)
GeneCalls[[3L]] <- gffToDataFrame(GFF = system.file("extdata",
                                           "GCA_000875775.1_ASM87577v1_genomic.gff.gz",
                                                   package = "SynExtend"),
                                  Verbose = TRUE)
names(GeneCalls) <- seq(length(GeneCalls))
Links <- NucleotideOverlap(SyntenyObject = Syn,
                           GeneCalls = GeneCalls,
                           LimitIndex = FALSE,
                           Verbose = TRUE)
PredictedPairs <- PairSummaries(SyntenyLinks = Links,
                                DBPATH = DBPATH,
                                PIDs = FALSE,
                                AcceptContigNames = TRUE,
                                Verbose = TRUE)
PresentSeqs <- ExtractBy(x = PredictedPairs,
                         Method = "all",
                         DBPATH = DBPATH,
                         Verbose = TRUE)
Clusters <- DisjointSet(Pairs = PredictedPairs,
                        Verbose = TRUE)
SeqsByClusters <- ExtractBy(x = PredictedPairs,
                            y = Clusters,
                            Method = "clusters",
                            DBPATH = DBPATH,
                            Verbose = TRUE)

# Alternatively the same seqs can be accessed from the NCBI FTP site
# And gene calls can be accessed with the rtracklayer
## Not run:
DBPATH <- tempfile()
FNAs <- c("ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/006/740/685/GCA_006740685.1_ASM674068v1/GCA_006740685.1_A
       "ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/956/175/GCA_000956175.1_ASM95617v1/GCA_000956175.1_ASM95
       "ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/875/775/GCA_000875775.1_ASM87577v1/GCA_000875775.1_ASM87
for (m1 in seq_along(FNAs)) {
 X <- readDNAStringSet(filepath = FNAs[m1])
 X <- X[order(width(X),
              decreasing = TRUE)]

  Seqs2DB(seqs = X,
          type = "XStringSet",
          dbFile = DBPATH,
          identifier = as.character(m1),
          verbose = TRUE)
  }

GeneCalls <- vector(mode = "list",
                    length = ncol(Syn))
GeneCalls[[1L]] <- rtracklayer::import(system.file("extdata",
                                           "GCA_006740685.1_ASM674068v1_genomic.gff.gz",
                                                   package = "SynExtend"))
GeneCalls[[2L]] <- rtracklayer::import(system.file("extdata",
                                            "GCA_000956175.1_ASM95617v1_genomic.gff.gz",
                                                   package = "SynExtend"))
GeneCalls[[3L]] <- rtracklayer::import(system.file("extdata",
                                            "GCA_000875775.1_ASM87577v1_genomic.gff.gz",
                                                   package = "SynExtend"))
names(GeneCalls) <- seq(length(GeneCalls))
Links <- NucleotideOverlap(SyntenyObject = Syn,
                           GeneCalls = GeneCalls,
                           LimitIndex = FALSE,
                           Verbose = TRUE)
PredictedPairs <- PairSummaries(SyntenyLinks = Links,
                                DBPATH = DBPATH,
                                PIDs = FALSE,
                                AcceptContigNames = TRUE,
                                Verbose = TRUE)
PresentSeqs <- ExtractBy(x = PredictedPairs,
                         Method = "all",
                         DBPATH = DBPATH,
                         Verbose = TRUE)
Clusters <- DisjointSet(Pairs = PredictedPairs,
                        Verbose = TRUE)
SeqsByClusters <- ExtractBy(x = PredictedPairs,
                            y = Clusters,
                            Method = "clusters",
                            DBPATH = DBPATH,
                            Verbose = TRUE)

## End(Not run)
```

EstimateRearrangementScenarios

Estimate Genome Rearrangement Events with Double Cut and Join Operations

Description

Take in a Synteny object from FindSynteny and return predicted rearrangement events.

Usage

```
EstimateRearrangementScenarios(synt,
                               num_runs = -1,
                               verbose = TRUE,
                               mean = FALSE,
                               opp_value = 0,
                               min_unique_length = 10,
                               min_block_length = -1,
                               actual = c(),
                               test_run = 0)
```

Arguments

`synt`	Synteny object obtained from running the FindSynteny object. Expected input is unichromosomal sequences, though multichromosomal sequences are supported.
`num_runs`	Numeric; Number of times to simulate scenarios. The default value of -1 (and all non-positive values) runs each analysis for *2b* iterations, where *b* is the number of unique breakpoints.
`mean`	If TRUE, returns the mean number of inversions and transpositions found. If FALSE, returns the scenario corresponding to the minimum total number of operations across all runs. This parameter only affects the number of inversions and transpositions reported; the specific scenario returned is one of the runs that resulted in a minimum value.
`min_block_length`	Minimum size of syntenic blocks to use for analysis. The default value accepts all blocks. Set to a larger value to ignore sections of short mutations that could be the result of SNPs or other small-scale mutations.
`verbose`	Displays progress bars if verbose=TRUE.
`opp_value`	A parameter.
`min_unique_length`	A parameter.
`actual`	A parameter.
`test_run`	A parameter.

Details

`EstimateRearrangementScenarios` is an implementation of the Double Cut and Join (DCJ) method for analyzing large scale mutation events. This implementation incentivizes minimum number of total events rather than total number of DCJs. Multiple scenarios are computed, and results of estimation are returned to the user.

Value

An *NxN* matrix with the same shape as the input Synteny object.

The diagonal corresponds to total sequence length of the corresponding genome.

In the upper triangle, entry `[i,j]` corresponds to the percent hits between genome `i` and genome `j`. In the lower triangle, entry `[i,j]` contains a List object with 5 properties:

- `$Inversions` and `$Transpositions` contain the (mean/min) number of estimated inversions and transpositions (resp.) between genome `i` and genome `j`.
- `$pct_hits` contains percent hits between the genomes.
- `$Scenario` shows a rearrangement scenario resulting in a minimum number of events between the genomes, using permutation matrix notation. Each line shows the operation performed, the permutation vector corresponding to the current state relative to genome 1, and the number of blocks modified by the previous step.
- `$Key` displays a key for converting from permutation matrix to genomes. This matrix contains the start index for each block on each genome, the length of the block, the relative direction of the block on genome 2 compared to genome 1, and the permutation number for that block.

Note

A note.

Author(s)

Aidan Lakshman (<`ahl27@pitt.edu`>)

References

Friedberg, R., Darling, A. E., & Yancopoulos, S. (2008). Genome rearrangement by the double cut and join operation. *Bioinformatics*, 385-416.

See Also

`FindSynteny`

Examples

```
db <- system.file("extdata", "Influenza.sqlite", package="DECIPHER")
synteny <- FindSynteny(db)
synteny

rearrs <- EstimateRearrangementScenarios(synteny)

rearrs          # view whole object
rearrs[[2,1]]   # view details on Genomes 1 and 2
```

`ExtractBy`	*Extract and organize* `XStringSet`*s of sequences represented in a* `PairSummaries` *object.*

Description

Takes in a `PairSummaries` object and an optional vector of cluster representatives. Return an `XStringSet` of the sequences present in the `PairSummaries`, or when cluster representatives are provided, a list of `XStringSets` of the sequences that make up the provided clusters.

Usage

```
ExtractBy(x,
          y = NULL,
          DBPATH,
          Method = "all",
          DefaultTranslationTable = "11",
          Translate = TRUE,
          Storage = 1,
          Verbose = FALSE)
```

Arguments

`x`	A `PairSummaries` object.
`y`	An optional `list` containing the ids of sequences in the `PairSummaries` object.
`DBPATH`	A SQLite connection object or a character string specifying the path to the database file. Constructed from DECIPHER's `Seqs2DB` function.
`Method`	How to extract sequences from the `PairSummaries` object. Currently only the methods "all" and "clusters" are supported.
`Translate`	If `TRUE` return `AAStringSets` where possible.
`DefaultTranslationTable`	Currently Not Implemented! When implemented will allow for designation of a specific translation table if one is not indicated in the `GeneCalls` attribute of the `PairSummaries` object.
`Storage`	Numeric indicating the approximate size a user wishes to allow for holding `StringSets` in memory to extract gene sequences, in "Gigabytes". The lower `Storage` is set, the more likely that `ExtractBy` will need to reaccess `StringSets` when extracting gene sequences. The higher `Storage` is set, the more sequences `ExtractBy` will attempt to hold in memory, avoiding the need to re-access the source database many times. Set to 1 by default, indicating that `ExtractBy` can store a "Gigabyte" of sequences in memory at a time.
`Verbose`	Logical indicating whether to print progress bars and messages. Defaults to `FALSE`.

Details

Takes in a PairSummaries object and an optional vector of cluster representatives. Return an XStringSet of the sequences present in the PairSummaries, or when cluster representatives are provided, a list of XStringSets of the sequences that make up the provided clusters.

Value

Returns either a XStringSet or a list of XStringSets.

Author(s)

Nicholas Cooley <npc19@pitt.edu>

See Also

FindSynteny, Synteny-class, PairSummaries, DisjointSet

Examples

```
DBPATH <- system.file("extdata",
                      "VignetteSeqs.sqlite",
                      package = "SynExtend")
Syn <- FindSynteny(dbFile = DBPATH)
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)
GeneCalls[[2L]] <- gffToDataFrame(GFF = system.file("extdata",
                                            "GCA_000956175.1_ASM95617v1_genomic.gff.gz",
                                                    package = "SynExtend"),
                                  Verbose = TRUE)
GeneCalls[[3L]] <- gffToDataFrame(GFF = system.file("extdata",
                                            "GCA_000875775.1_ASM87577v1_genomic.gff.gz",
                                                    package = "SynExtend"),
                                  Verbose = TRUE)
names(GeneCalls) <- seq(length(GeneCalls))
Links <- NucleotideOverlap(SyntenyObject = Syn,
                           GeneCalls = GeneCalls,
                           LimitIndex = FALSE,
                           Verbose = TRUE)
PredictedPairs <- PairSummaries(SyntenyLinks = Links,
                                DBPATH = DBPATH,
                                PIDs = FALSE,
                                AcceptContigNames = TRUE,
                                Verbose = TRUE)
PresentSeqs <- ExtractBy(x = PredictedPairs,
                         Method = "all",
                         DBPATH = DBPATH,
                         Verbose = TRUE)
```

```
Clusters <- DisjointSet(Pairs = PredictedPairs,
                        Verbose = TRUE)
SeqsByClusters <- ExtractBy(x = PredictedPairs,
                            y = Clusters,
                            Method = "clusters",
                            DBPATH = DBPATH,
                            Verbose = TRUE)

# Alternatively the same seqs can be accessed from the NCBI FTP site
# And gene calls can be accessed with the rtracklayer
## Not run:
DBPATH <- tempfile()
FNAs <- c("ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/006/740/685/GCA_006740685.1_ASM674068v1/GCA_006740685.1_A
       "ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/956/175/GCA_000956175.1_ASM95617v1/GCA_000956175.1_ASM95
       "ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/875/775/GCA_000875775.1_ASM87577v1/GCA_000875775.1_ASM87
for (m1 in seq_along(FNAs)) {
 X <- readDNAStringSet(filepath = FNAs[m1])
 X <- X[order(width(X),
              decreasing = TRUE)]

 Seqs2DB(seqs = X,
         type = "XStringSet",
         dbFile = DBPATH,
         identifier = as.character(m1),
         verbose = TRUE)
  }

GeneCalls <- vector(mode = "list",
                    length = ncol(Syn))
GeneCalls[[1L]] <- rtracklayer::import(system.file("extdata",
                                              "GCA_006740685.1_ASM674068v1_genomic.gff.gz",
                                                      package = "SynExtend"))
GeneCalls[[2L]] <- rtracklayer::import(system.file("extdata",
                                               "GCA_000956175.1_ASM95617v1_genomic.gff.gz",
                                                      package = "SynExtend"))
GeneCalls[[3L]] <- rtracklayer::import(system.file("extdata",
                                               "GCA_000875775.1_ASM87577v1_genomic.gff.gz",
                                                      package = "SynExtend"))
names(GeneCalls) <- seq(length(GeneCalls))
Links <- NucleotideOverlap(SyntenyObject = Syn,
                           GeneCalls = GeneCalls,
                           LimitIndex = FALSE,
                           Verbose = TRUE)
PredictedPairs <- PairSummaries(SyntenyLinks = Links,
                                DBPATH = DBPATH,
                                PIDs = FALSE,
                                AcceptContigNames = TRUE,
                                Verbose = TRUE)
PresentSeqs <- ExtractBy(x = PredictedPairs,
                         Method = "all",
                         DBPATH = DBPATH,
                         Verbose = TRUE)
Clusters <- DisjointSet(Pairs = PredictedPairs,
                        Verbose = TRUE)
SeqsByClusters <- ExtractBy(x = PredictedPairs,
                            y = Clusters,
                            Method = "clusters",
                            DBPATH = DBPATH,
                            Verbose = TRUE)

## End(Not run)
```

FindSets — *Find all single linkage clusters in an undirected pairs list.*

Description

Take in a pair of vectors representing the columns of an undirected pairs list and return the single linkage clusters.

Usage

```
FindSets(p1,
         p2,
         Verbose = FALSE)
```

Arguments

p1	Column 1 of a pairs matrix or list.
p2	Column 2 of a pairs matrix or list.
Verbose	Logical indicating whether or not to display a progress bar and print the time difference upon completion.

Details

`FindSets` uses a version of the union-find algorithm to collect single linkage clusters from a pairs list. Currently meant to be used inside a wrapper function, but left exposed for user convenience.

Value

A two column matrix with the first column being input nodes, and the second the node representing a single linkage cluster.

Author(s)

Nicholas Cooley <npc19@pitt.edu>

See Also

PairSummaries

Examples

```
set.seed(1986)
m <- cbind(as.integer(sample(30, size = 25,
                             replace = TRUE)),
           as.integer(sample(35, size = 25,
                             replace = TRUE)))

Levs <- unique(c(m[, 1],
                 m[, 2]))
m <- cbind("1" = as.integer(factor(x = m[, 1L],
                                   levels = Levs)),
           "2" = as.integer(factor(x = m[, 2L],
                                   levels = Levs)))
z <- FindSets(p1 = m[, 1],
              p2 = m[, 2])
```

`Generic` *Model for predicting PID based on k-mer statistics*

Description

Though the function PairSummaries provides an argument allowing users to ask for alignments, given the time consuming nature of that process on large data, models are provided for predicting PIDs of pairs based on k-mer statistics without performing alignments.

Usage

```
data("Generic")
```

Format

The format is an object of class “glm”.

Details

A model for predicting the PID of a pair of sequences based on the k-mers that were used to link the pair.

Examples

```
data(Generic)
```

`gffToDataFrame` *Generate a DataFrame of gene calls from a gff3 file*

Description

Generate a DataFrame of gene calls from a gff3 file

Usage

```
gffToDataFrame(GFF,
               AdditionalAttrs = NULL,
               AdditionalTypes = NULL,
               RawTableOnly = FALSE,
               Verbose = FALSE)
```

Arguments

`GFF`	A url or filepath specifying a gff3 file to import
`AdditionalAttrs`	A vector of character strings to designate the attributes to pull. Default Attributes include: “ID”, “Parent”, “Name”, “gbkey”, “gene”, “product”, “protein_id”, “gene_biotype”, “transl_table”, and “Note”.
`AdditionalTypes`	A vector of character strings to query from the the “Types” column. Default types are limited to “Gene” and “Pseudogene”, but any possible entry for “Type” in a gff3 format can be added, such as “rRNA”, or “CRISPR_REPEAT”.
`RawTableOnly`	Logical specifying whether to return the raw imported GFF without complex parsing. Remains as a holdover from function construction and debugging. For simple gff3 import see `rtracklayer::import`.
`Verbose`	Logical specifying whether to print a progress bar and time difference.

Details

Import a gff file into a rectangular parsable object.

Value

A DataFrame with relevant information extracted from a GFF.

Author(s)

Nicholas Cooley <npc19@pitt.edu>

Examples

```
ImportedGFF <- gffToDataFrame(GFF = system.file("extdata",
                                                "GCA_006740685.1_ASM674068v1_genomic.gff.gz",
                                                      package = "SynExtend"),
                                    Verbose = TRUE)
```

LinkedPairs   *Tables of where syntenic hits link pairs of genes*

Description

Syntenic blocks describe where order is shared between two sequences. These blocks are made up of exact match hits. These hits can be overlayed on the locations of sequence features to clearly illustrate where exact sequence similarity is shared between pairs of sequence features.

Usage

```
## S3 method for class 'LinkedPairs'
print(x,
      quote = FALSE,
      right = TRUE,
      ...)
```

Arguments

x	An object of class LinkedPairs.
quote	Logical indicating whether to print the output surrounded by quotes.
right	Logical specifying whether to right align strings.
...	Other arguments for print.

Details

Objects of class LinkedPairs are stored as square matrices of list elements with dimnames derived from the dimnames of the object of class "Synteny" from which it was created. The diagonal of the matrix is only filled if OutputFormat "Comprehensive" is selected in NucleotideOverlap, in which case it will be filled with the gene locations supplied to GeneCalls. The upper triangle is always filled, and contains location information in nucleotide space for all syntenic hits that link features between sequences in the form of an integer matrix with named columns. "QueryGene" and "SubjectGene" correspond to the integer rownames of the supplied gene calls. "QueryIndex" and "SubjectIndex" correspond to "Index1" and "Index2" columns of the source synteny object position. Remaining columns describe the exact positioning and size of extracted hits. The lower triangle is not filled if OutputFormat "Sparse" is selected and contains relative displacement positions for the 'left-most' and 'right-most' hit involved in linking the particular features indicated in the related line up the corresponding position in the upper triangle.

The object serves only as a simple package for input data to the PairSummaries function, and as such may not be entirely user friendly. However it has been left exposed to the user should they find this data interesting.

Value

An object of class ”LinkedPairs”.

Author(s)

Nicholas Cooley `<npc19@pitt.edu>`

`NucleotideOverlap` *Tabulating Pairs of Genomic Sequences*

Description

A function for concisely tabulating where genomic features are connected by syntenic hits.

Usage

```
NucleotideOverlap(SyntenyObject,
                  GeneCalls,
                  LimitIndex = FALSE,
                  AcceptContigNames = TRUE,
                  Verbose = FALSE)
```

Arguments

`SyntenyObject`	An object of class “Synteny” built from the `FindSynteny` in the package `DECIPHER`.
`GeneCalls`	A named list of objects of class “DFrame” built from `gffToDataFrame`, objects of class “GRanges” imported from `rtracklayer::import`, or objects of class “Genes” created from the `DECIPHER` function `FindGenes`. “DFrame”s built by “gffToDataFrame” can be used directly, while “GRanges” objects may also be used with limited functionality. Using a “GRanges” object will force all alignments to nucleotide alignments. Objects of class “Genes” generated by `FindGenes` function equivalently to those produced by `gffToDataFrame`. Using a “GRanges” object will force `LimitIndex` to `TRUE`.
`LimitIndex`	Logical indicating whether to limit which indices in a synteny object to query. `FALSE` by default, when `TRUE` only the first sequence in all selected identifiers will be used. `LimitIndex` can be used to skip analysis of plasmids, or solely query a single chromosome.
`AcceptContigNames`	Match names of contigs between gene calls object and synteny object. Where relevant, the first white space and everything following are removed from contig names. If “TRUE”, NucleotideOverlap assumes that the contigs at each position in the synteny object and “GeneCalls” object are in the same order. Is automatically set to TRUE when “GeneCalls” are of class “GRanges”.
`Verbose`	Logical indicating whether or not to display a progress bar and print the time difference upon completion.

Details

Builds a matrix of lists that contain information about linked pairs of genomic features.

Value

An object of class “LinkedPairs”. “LinkedPairs” is fundamentally just a list in the form of a matrix. The lower triangle of the matrix is populated with matrices that contain all kmer hits from the “Synteny” object that link features from the “GeneCalls” object. The upper triangle is populated by matrices of the summaries of those hits by feature. The diagonal is populated by named vectors of the lengths of the contigs, much like in the “Synteny” object. The “LinkedPairs” object also contains a “GeneCalls” attribute that contains the user supplied features in a slightly more trimmed down form. This allows users to only need to supply gene calls once and not again in the “PairSummaries” function.

Author(s)

Nicholas Cooley <npc19@pitt.edu>

See Also

FindSynteny, Synteny-class

Examples

```
DBPATH <- system.file("extdata",
                      "VignetteSeqs.sqlite",
                      package = "SynExtend")
Syn <- FindSynteny(dbFile = DBPATH)
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)
GeneCalls[[2L]] <- gffToDataFrame(GFF = system.file("extdata",
                                            "GCA_000956175.1_ASM95617v1_genomic.gff.gz",
                                                    package = "SynExtend"),
                                  Verbose = TRUE)
GeneCalls[[3L]] <- gffToDataFrame(GFF = system.file("extdata",
                                            "GCA_000875775.1_ASM87577v1_genomic.gff.gz",
                                                    package = "SynExtend"),
                                  Verbose = TRUE)
names(GeneCalls) <- seq(length(GeneCalls))
Links <- NucleotideOverlap(SyntenyObject = Syn,
                           GeneCalls = GeneCalls,
                           LimitIndex = FALSE,
                           Verbose = TRUE)

# Alternatively the same seqs can be accessed from the NCBI FTP site
## Not run:
```

```
DBPATH <- tempfile()
FNAs <- c("ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/006/740/685/GCA_006740685.1_ASM674068v1/GCA_006740685.1_A
       "ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/956/175/GCA_000956175.1_ASM95617v1/GCA_000956175.1_ASM95
       "ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/875/775/GCA_000875775.1_ASM87577v1/GCA_000875775.1_ASM87
for (m1 in seq_along(FNAs)) {
 X <- readDNAStringSet(filepath = FNAs[m1])
 X <- X[order(width(X),
              decreasing = TRUE)]

 Seqs2DB(seqs = X,
         type = "XStringSet",
         dbFile = DBPATH,
         identifier = as.character(m1),
         verbose = TRUE)
  }

GeneCalls <- vector(mode = "list",
                    length = ncol(Syn))
GeneCalls[[1L]] <- rtracklayer::import(system.file("extdata",
                                           "GCA_006740685.1_ASM674068v1_genomic.gff.gz",
                                                   package = "SynExtend"))
GeneCalls[[2L]] <- rtracklayer::import(system.file("extdata",
                                            "GCA_000956175.1_ASM95617v1_genomic.gff.gz",
                                                   package = "SynExtend"))
GeneCalls[[3L]] <- rtracklayer::import(system.file("extdata",
                                            "GCA_000875775.1_ASM87577v1_genomic.gff.gz",
                                                   package = "SynExtend"))
names(GeneCalls) <- seq(length(GeneCalls))
Links <- NucleotideOverlap(SyntenyObject = Syn,
                           GeneCalls = GeneCalls,
                           LimitIndex = FALSE,
                           Verbose = TRUE)

## End(Not run)
```

PairSummaries *Summarize connected pairs in a LinkedPairs object*

Description

Takes in a “LinkedPairs” object and gene calls, and returns a data.frame of paired features.

Usage

```
PairSummaries(SyntenyLinks,
              DBPATH,
              PIDs = FALSE,
              IgnoreDefaultStringSet = FALSE,
              Verbose = FALSE,
              Model = "Generic",
              DefaultTranslationTable = "11",
              AcceptContigNames = TRUE,
              OffSetsAllowed = 2L,
              Storage = 1,
              ...)
```

Arguments

`SyntenyLinks`	A `LinkedPairs` object. In previous versions of this function, a `GeneCalls` object was also required, but this object is now carried forward from `NucleotideOverlap` inside the `LinkedPairs` object.
`DBPATH`	A SQLite connection object or a character string specifying the path to the database file. Constructed from DECIPHER's `Seqs2DB` function. This path is always required as "PairsSummaries" computes the tetramer distance between paired sequences.
`PIDs`	Logical indicating whether to perform pairwise alignments. If `TRUE` all pairs will be aligned using DECIPHER's `AlignProfiles`. This step can be time consuming, especially for large numbers of pairs. Default is `FALSE`.
`IgnoreDefaultStringSet`	Logical indicating alignment type preferences. If `FALSE` (the default) pairs that can be aligned in amino acid space will be aligned as an `AAStringSet`. If `TRUE` all pairs will be aligned in nucleotide space. For `PairSummaries` to align the translation of a pair of sequences, both sequences must be tagged as coding in the "GeneCalls" object, and be the correct width for translation.
`Verbose`	Logical indicating whether or not to display a progress bar and print the time difference upon completion.
`...`	Arguments to be passed to `AlignProfiles`, and `DistanceMatrix`.
`Model`	A character string specifying a model to use to predict PIDs without performing an alignment. By default this argument is "Generic" specifying a generic PID prediction model based on PIDs computed from a randomly selected set of genomes. Currently no other models are included. Users may also supply their own model of type "glm" if they so desire in the form of an RData file. This model will need to take in some, or of the columns of statistics per pair that PairSummaries supplies.
`DefaultTranslationTable`	A character used to set the default translation table for `translate`. Is passed to `getGeneticCode`. Used when no translation table is specified in the "GeneCalls" object.
`AcceptContigNames`	Match names of contigs between gene calls object and synteny object. Where relevant, the first white space and everything following are removed from contig names. If `TRUE`, PairSummaries assumes that the contigs at each position in the synteny object and "GeneCalls" object are in the same order. Is automatically set to `TRUE` when "GeneCalls" are of class "GRanges". Is currently `TRUE` by default.

`OffSetsAllowed`	Integer vector defaulting to “2L” that sets the gap size that is allowed to be filled, if gaps are queried. The default value queries gaps of size 1. If set to “NULL” no gaps are queried. Setting to “c(2L, 3L)” would query all gaps of size 1 and 2.
`Storage`	Numeric indicating the approximate size a user wishes to allow for holding `StringSets` in memory to extract gene sequences, in “Gigabytes”. The lower `Storage` is set, the more likely that `PairSummaries` will need to reaccess `StringSets` when extracting gene sequences. The higher `Storage` is set, the more sequences `PairSummaries` will attempt to hold in memory, avoiding the need to re-access the source database many times. Set to 1 by default, indicating that `PairSummaries` can store a “Gigabyte” of sequences in memory at a time.

Details

The `LinkedPairs` object generated by `NucleotideOverlap` is a container for raw data that describes possible orthologous relationships, however ultimate assignment of orthology is up to user discretion. `PairSummaries` generates a clear table with relevant statistics for a user to work with as they choose. The option to align all pairs, though onerous can allow users to apply a hard threshold to predictions by PID, while built in models can allow more expedient thresholding from predicted PIDs.

Value

A data.frame of class “data.frame” and “PairSummaries” of paired genes that are connected by syntenic hits. Contains columns describing the k-mers that link the pair. Columns “p1” and “p2” give the location ids of the the genes in the pair in the form “DatabaseIdentifier_ContigIdentifier_GeneIdentifier”. “ExactMatch” provides an integer representing the exact number of nucleotides contained in the linking k-mers. “TotalKmers” provides an integer describing the number of distinct k-mers linking the pair. “MaxKmer” provides an integer describing the largest k-mer that links the pair. A column titled “Consensus” provides a value between zero and 1 indicating whether the kmers that link a pair of features are in the same position in each feature, with 1 indicating they are in exactly the same position and 0 indicating they are in as different a position as is possible. The “Adjacent” column provides an integer value ranging between 0 and 2 denoting whether a feature pair’s direct neighbors are also paired. Gap filled pairs neither have neighbors, or are included as neighbors. The “TetDist” column provides the euclidean distance between oligonucleotide - of size 4 - frequences between predicted pairs. “PIDType” provides a character vector with values of “NT” where either of the pair indicates it is not a translatable sequence or “AA” where both sequences are translatable. If users choose to perform pairwise alignments there will be a “PID” column providing a numeric describing the percent identity between the two sequences. If users choose to predict PIDs using their own, or a provided model, a “PredictedPID” column will be provided.

Author(s)

Nicholas Cooley <`npc19@pitt.edu`>

See Also

`FindSynteny`, `Synteny-class`

Examples

```
DBPATH <- system.file("extdata",
                      "VignetteSeqs.sqlite",
                      package = "SynExtend")
Syn <- FindSynteny(dbFile = DBPATH)
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)
GeneCalls[[2L]] <- gffToDataFrame(GFF = system.file("extdata",
                                          "GCA_000956175.1_ASM95617v1_genomic.gff.gz",
                                                   package = "SynExtend"),
                                  Verbose = TRUE)
GeneCalls[[3L]] <- gffToDataFrame(GFF = system.file("extdata",
                                          "GCA_000875775.1_ASM87577v1_genomic.gff.gz",
                                                   package = "SynExtend"),
                                  Verbose = TRUE)
names(GeneCalls) <- seq(length(GeneCalls))
Links <- NucleotideOverlap(SyntenyObject = Syn,
                           GeneCalls = GeneCalls,
                           LimitIndex = FALSE,
                           Verbose = TRUE)
PredictedPairs <- PairSummaries(SyntenyLinks = Links,
                                DBPATH = DBPATH,
                                PIDs = FALSE,
                                AcceptContigNames = TRUE,
                                Verbose = TRUE)

# Alternatively the same seqs can be accessed from the NCBI FTP site
# And gene calls can be accessed with the rtracklayer
## Not run:
DBPATH <- tempfile()
FNAs <- c("ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/006/740/685/GCA_006740685.1_ASM674068v1/GCA_006740685.1_A
       "ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/956/175/GCA_000956175.1_ASM95617v1/GCA_000956175.1_ASM95
       "ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/875/775/GCA_000875775.1_ASM87577v1/GCA_000875775.1_ASM87
for (m1 in seq_along(FNAs)) {
 X <- readDNAStringSet(filepath = FNAs[m1])
 X <- X[order(width(X),
              decreasing = TRUE)]

 Seqs2DB(seqs = X,
         type = "XStringSet",
         dbFile = DBPATH,
         identifier = as.character(m1),
         verbose = TRUE)
  }

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

```
GeneCalls[[1L]] <- rtracklayer::import(system.file("extdata",
                                           "GCA_006740685.1_ASM674068v1_genomic.gff.gz",
                                                   package = "SynExtend"))
GeneCalls[[2L]] <- rtracklayer::import(system.file("extdata",
                                            "GCA_000956175.1_ASM95617v1_genomic.gff.gz",
                                                   package = "SynExtend"))
GeneCalls[[3L]] <- rtracklayer::import(system.file("extdata",
                                            "GCA_000875775.1_ASM87577v1_genomic.gff.gz",
                                                   package = "SynExtend"))
names(GeneCalls) <- seq(length(GeneCalls))
Links <- NucleotideOverlap(SyntenyObject = Syn,
                           GeneCalls = GeneCalls,
                           LimitIndex = FALSE,
                           Verbose = TRUE)
PredictedPairs <- PairSummaries(SyntenyLinks = Links,
                                DBPATH = DBPATH,
                                PIDs = FALSE,
                                AcceptContigNames = TRUE,
                                Verbose = TRUE)

## End(Not run)
```

SequenceSimilarity *Return a numeric value that represents the similarity between two aligned sequences as determined by a provided subsitution matrix.*

Description

Takes in a DNAStringSet or AAStringSet representing a pairwise alignment and a subsitution matrix such as those present in PFASUM, and return a numeric value representing sequence similarity as defined by the substitution matrix.

Usage

```
SequenceSimilarity(Seqs,
                     SubMat,
                     penalizeGapLetter = TRUE,
                     includeTerminalGaps = TRUE,
                     allowNegative = TRUE)
```

Arguments

Seqs	A DNAStringSet or AAStringSet of length 2.
SubMat	A named matrix representing a substitution matrix. If left "NULL" and "Seqs" is a AAStringSet, the 40th "PFASUM" matrix is used. If left "NULL" and "Seqs" is a DNAStringSet, a matrix with only the diagonal filled with "1"'s is used.
penalizeGapLetter	A logical indicating whether or not to penalize Gap-Letter matches. Defaults to "TRUE".

`includeTerminalGaps`
: A logical indicating whether or not to penalize terminal matches. Defaults to "TRUE".

`allowNegative`
: A logical indicating whether or not allow negative scores. Defaults to "TRUE". If "FALSE" scores that are returned as less than zero are converted to zero.

Details

Takes in a `DNAStringSet` or `AAStringSet` representing a pairwise alignment and a subsitution matrix such as those present in `PFASUM`, and return a numeric value representing sequence similarity as defined by the substitution matrix.

Value

Returns a single numeric.

Author(s)

Erik Wright <`ESWRIGHT@pitt.edu`> Nicholas Cooley <`npc19@pitt.edu`>

See Also

`AlignSeqs`, `AlignProfiles`, `AlignTranslation`, `DistanceMatrix`

Examples

```
db <- system.file("extdata", "Bacteria_175seqs.sqlite", package="DECIPHER")
dna <- SearchDB(db, remove="all")
alignedDNA <- AlignSeqs(dna[1:2])

DNAPlaceholder <- diag(15)
dimnames(DNAPlaceholder) <- list(DNA_ALPHABET[1:15],
                                 DNA_ALPHABET[1:15])

SequenceSimilarity(Seqs = alignedDNA,
                   SubMat = DNAPlaceholder,
                   includeTerminalGaps = TRUE,
                   penalizeGapLetter = TRUE,
                   allowNegative = TRUE)
```

`SubSetPairs` *Subset a "PairSummaries" object.*

Description

For a given object of class "PairSummaries", pairs based on either competing predictions, user thresholds on prediction statistics, or both.

Usage

```
SubSetPairs(CurrentPairs,
            UserThresholds,
            RejectCompetitors = TRUE,
            RejectionCriteria = "PID",
            WinnersOnly = TRUE,
            Verbose = FALSE)
```

Arguments

`CurrentPairs`	An object of class “PairSummaries”. Can also take in a generic “data.frame”, as long as the feature naming scheme is the same as that followed by all `SynExtend` functions.
`UserThresholds`	A named vector where values indicate a threshold for statistics to be above, and names designate which statistic to threshold on.
`RejectCompetitors`	A logical that defaults to “TRUE”. Allowing users to choose to remove competing predictions. When set to “FALSE”, no competitor rejection is performed. When “TRUE” all competing pairs with the exception of the best pair as determined by “RejectionCriteria” are rejected. Can additionally be set to a numeric or integer, in which case only competing predictions below that value are dropped.
`RejectionCriteria`	A character indicating which column value competitor rejection should reference. Defaults to “PID”.
`WinnersOnly`	A logical indicating whether or not to return just the pairs that are selected. Defaults to “TRUE” to return a subset object of class “PairSummaries”. When “FALSE”, function returns a list of two “PairSummaries” objects, one of the selected pairs, and the second of the rejected pairs.
`Verbose`	Logical indicating whether or not to display a progress bar and print the time difference upon completion.

Details

`SubSetPairs` uses a naive competitor rejection algorithm to remove predicted pairs when nodes are predicted to be paired to multiple nodes within the same index.

Value

An object of class “PairSummaries”, or a list of two “PairSummaries” objects.

Author(s)

Nicholas Cooley <`npc19@pitt.edu`>

See Also

`PairSummaries` `NucleotideOverlap`

Examples

```
## Not run:
DBPATH <- system.file("extdata",
                      "VignetteSeqs.sqlite",
                      package = "SynExtend")
Syn <- FindSynteny(dbFile = DBPATH)
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)
GeneCalls[[2L]] <- gffToDataFrame(GFF = system.file("extdata",
                                            "GCA_000956175.1_ASM95617v1_genomic.gff.gz",
                                                    package = "SynExtend"),
                                  Verbose = TRUE)
GeneCalls[[3L]] <- gffToDataFrame(GFF = system.file("extdata",
                                            "GCA_000875775.1_ASM87577v1_genomic.gff.gz",
                                                    package = "SynExtend"),
                                  Verbose = TRUE)
names(GeneCalls) <- seq(length(GeneCalls))
Links <- NucleotideOverlap(SyntenyObject = Syn,
                           GeneCalls = GeneCalls,
                           LimitIndex = FALSE,
                           Verbose = TRUE)
PredictedPairs <- PairSummaries(SyntenyLinks = Links,
                                DBPATH = DBPATH,
                                PIDs = FALSE,
                                AcceptContigNames = TRUE,
                                Verbose = TRUE)
# remove competitors under default conditions
Pairs2 <- SubSetPairs(CurrentPairs = PredictedPairs,
                      Verbose = TRUE)
THRESH <- c(0.5, 21)
names(THRESH) <- c("Consensus", "ExactMatch")
# remove pairs only based on user defined thresholds
Pairs3 <- SubSetPairs(CurrentPairs = PredictedPairs,
                      UserThresholds = THRESH,
                      RejectCompetitors = FALSE,
                      Verbose = TRUE)

## End(Not run)
```

Index