Package 'DECIPHER'

April 5, 2014

Type Package

Title Database Enabled Code for Ideal Probe Hybridization Employing R

Version 1.8.0

Date 2013-09-18

Author Erik Wright

Maintainer Erik Wright <DECIPHER@cae.wisc.edu>

biocViews Clustering, Genetics, Sequencing, Infrastructure,DataImport, Visualization, Microarray, QualityControl

Description A toolset that assist in the design of hybridization probes.

Depends R (>= 2.13.0), Biostrings (>= 2.29), RSQLite (>= 0.9),IRanges, stats, XVector, parallel

Imports Biostrings, parallel, RSQLite, IRanges, stats, XVector

LinkingTo Biostrings, parallel, RSQLite, IRanges, stats, XVector

License GPL-3

LazyData yes

R topics documented:

`DECIPHER-package` *Database Enabled Code for Ideal Probe Hybridization Employing R*

Description

Database Enabled Code for Ideal Probe Hybridization Employing R (DECIPHER) is a software toolset that can be used for deciphering and managing DNA sequences efficiently using the R statistical programming language. The program is designed to be used with non-destructive workflows that guide the user through the process of importing, maintaining, analyzing, manipulating, and exporting a massive amount of DNA sequences. Some functionality of the program is provided online through web tools. DECIPHER is an ongoing project at the University of Wisconsin Madison and is freely available for download.

Details

Package:	DECIPHER
Type:	Package
Depends:	R (>= 2.13.0), Biostrings (>= 2.29), RSQLite (>= 0.9), IRanges, stats, XVector, parallel
Imports:	Biostrings, parallel, RSQLite, IRanges, stats, XVector
LinkingTo:	Biostrings, parallel, RSQLite, IRanges, stats, XVector
License:	GPL-3
LazyLoad:	yes

Index:

Author(s)

Erik Wright

Maintainer: Erik Wright <DECIPHER@cae.wisc.edu>

Add2DB *Add Data To A Database*

Description

Adds a `data.frame` to a database table by `row.names`.

Usage

```
Add2DB(myData,
       dbFile,
       tblName = "DNA",
       verbose = TRUE,
       ...)
```

Arguments

myData	Data frame containing information to be added to the dbFile.
dbFile	A SQLite connection object or a character string specifying the path to the database file.
tblName	Character string specifying the table in which to add the data.
verbose	Logical indicating whether to display each query as it is sent to the database.
...	Additional expressions to add as part of a where clause in the query. Further arguments provided in ... will be added to the query separated by " and " as part of the where clause.

Details

Data contained in myData will be added to the tblName by its respective row.names.

Value

Returns TRUE if the data was added successfully.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

See Also

Seqs2DB, SearchDB, BrowseDB

Examples

```
# Create a sequence database
gen <- system.file("extdata", "Bacteria_175seqs.gen", package="DECIPHER")
dbConn <- dbConnect(SQLite(), ":memory:")
Seqs2DB(gen, "GenBank", dbConn, "Bacteria")

# Identify the sequence lengths
l <- IdLengths(dbConn)

# Add lengths to the database
Add2DB(l, dbConn)

# View the added lengths
BrowseDB(dbConn)
dbDisconnect(dbConn)
```

AlignProfiles *Aligns Two Sets of Aligned Sequences*

Description

Aligns two sets of one or more aligned sequences by first generating representative profiles, then aligning the profiles with dynamic programming, and finally merging the two aligned sequence sets.

Usage

```
AlignProfiles(pattern,
              subject,
              p.weight = 1,
              s.weight = 1,
              perfectMatch = 2,
              misMatch = -3,
              gapOpening = -6,
              gapExtension = -4,
              terminalGap = -1,
              processors = NULL)
```

Arguments

pattern	A DNAStringSet object of aligned sequences to use as the pattern.
subject	A DNAStringSet object of aligned sequences to use as the subject.
p.weight	A numeric vector of weights for each sequence in the pattern to use in generating a profile, or a single number implying equal weights.
s.weight	A numeric vector of weights for each sequence in the subject to use in generating a profile, or a single number implying equal weights.
perfectMatch	Numeric giving the reward for aligning two matching nucleotides in the alignment.
misMatch	Numeric giving the cost for aligning two mismatched nucleotides in the alignment.
gapOpening	Numeric giving the cost for opening a gap in the alignment.
gapExtension	Numeric giving the cost for extending an open gap in the alignment.
terminalGap	Numeric giving the cost for allowing leading and trailing gaps in the alignment. Either two numbers, the first for leading gaps and the second for trailing gaps, or a single number for both.
processors	The number of processors to use, or NULL (the default) for all available processors.

Details

Profiles are aligned using dynamic programming, a variation of the Needleman-Wunsch algorithm.

Value

A DNAStringSet of aligned sequences.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

References

Needleman S., Wunsch, C. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology, **48(3)**, 443-453.

See Also

AlignSeqs

Examples

```
db <- system.file("extdata", "Bacteria_175seqs.sqlite", package="DECIPHER")
dna1 <- SearchDB(db, remove="common", limit=100) # the first 100 sequences
dna2 <- SearchDB(db, remove="common", limit="100,100") # the rest
alignedDNA <- AlignProfiles(dna1, dna2)
BrowseSequences(alignedDNA, highlight=1)
```

AlignSeqs *Aligns A Set of Unaligned Sequences*

Description

Performs profile-to-profile alignment of multiple unaligned sequences following a guide tree.

Usage

```
AlignSeqs(myDNAStringSet,
          perfectMatch = 2,
          misMatch = -3,
          gapOpening = -6,
          gapExtension = -4,
          terminalGap = -1,
          doNotAlign = 0.75,
          guideTree = NULL,
          orient = TRUE,
          processors = NULL,
          verbose = TRUE)
```

Arguments

myDNAStringSet	A DNAStringSet object of unaligned sequences to be aligned.
perfectMatch	Numeric giving the reward for aligning two matching nucleotides in the alignment.
misMatch	Numeric giving the cost for aligning two mismatched nucleotides in the alignment.
gapOpening	Numeric giving the cost for opening a gap in the alignment.
gapExtension	Numeric giving the cost for extending an open gap in the alignment.
terminalGap	Numeric giving the cost for allowing leading and trailing gaps in the alignment. Either two numbers, the first for leading gaps and the second for trailing gaps, or a single number for both.
doNotAlign	Numeric specifying the threshold above which to not attempt to align sequences. Roughly equivalent to the fraction dissimilarity between unalignable sequences. (See details section below.)
guideTree	Either NULL or a data.frame giving the ordered tree structure in which to align profiles. If NULL then a guide tree will be constructed. (See details section below.)
orient	Logical specifying whether some sequences may need to be reoriented (reverse complemented) before alignment. If TRUE, an attempt to determine orientation will be performed with sequences reoriented as necessary.
processors	The number of processors to use, or NULL (the default) for all available processors.
verbose	Logical indicating whether to display progress.

Details

The profile-to-profile method aligns a sequence set by merging profiles along a guide tree until all sequences are aligned. If guideTree=NULL, an initial UPGMA guide tree is constructed based on a distance matrix of shared k-mers. A second guide tree is built based on the initial alignment, and the alignment is refined using this tree. If a guideTree is provided then sequences are only aligned once. The guideTree should be provided in the output given by IdClusters with increasing levels of cutoff.

If doNotAlign is less than the height of the guideTree, then sequences more than doNotAlign distance apart are not aligned in order to save time. For example, setting doNotAlign to 0.75 (the default) would not align groups of sequences that are more than 75% dissimilar, whereas setting it to 1.0 would attempt to align all sequences. The doNotAlign argument can be used to discourage attempts at aligning completely unrelated DNA sequences, such as random DNA which would be 75% dissimilar on average. In such cases, groups of sequences are not aligned to each other, and instead have gaps added to make them the same length. In most reasonable alignment scenarios the sequences being aligned are generally not more than 50% dissimilar.

Value

A DNAStringSet of aligned sequences.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

See Also

AlignProfiles, IdClusters

Examples

```
db <- system.file("extdata", "Bacteria_175seqs.sqlite", package="DECIPHER")
dna <- SearchDB(db, limit=10, remove="all")
alignedDNA <- AlignSeqs(dna)
BrowseSequences(alignedDNA, highlight=1)
```

Array2Matrix *Creates a Matrix Representation of a Microarray*

Description

Converts the output of DesignArray into the sparse matrix format used by NNLS.

Usage

```
Array2Matrix(probes,
             verbose = TRUE)
```

Arguments

probes	A set of microarray probes in the format output by DesignArray.
verbose	Logical indicating whether to display progress.

Details

A microarray can be represented by a matrix of hybridization efficiencies, where the rows represent each of the probes and the columns represent each the possible templates. This matrix is sparse since microarray probes are designed to only target a small subset of the possible templates.

Value

A list specifying the hybridization efficiency of each probe to its potential templates.

i	Element's row index in the sparse matrix.
j	Element's column index in the sparse matrix.
x	Non-zero elements' values representing hybridization efficiencies.
dimnames	A list of two components: the names of each probe, and the names of each template.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

References

Coming Soon!

See Also

DesignArray, NNLS

Examples

```
fas <- system.file("extdata", "Bacteria_175seqs.fas", package="DECIPHER")
dna <- readDNAStringSet(fas)
names(dna) <- 1:length(dna)
probes <- DesignArray(dna)
A <- Array2Matrix(probes)
```

BrowseDB *View A Database Table In A Web Browser*

Description

Opens an html file in a web browser to show the contents of a table in a database.

Usage

```
BrowseDB(dbFile,
         htmlFile = paste(tempdir(), "/db.html", sep = ""),
         tblName = "DNA",
         identifier = "",
         limit = -1,
         orderBy = "row_names",
         maxChars = 50,
         ...)
```

Arguments

dbFile	A SQLite connection object or a character string specifying the path to the database file.
htmlFile	Character string giving the location where the html file should be written.
tblName	Character string specifying the table to view.
identifier	Optional character string used to narrow the search results to those matching a specific identifier. If "" then all identifiers are selected.
limit	Number of results to display. The default (-1) does not limit the number of results.

orderBy	Character string giving the column name for sorting the results. Defaults to the order of entries in the database. Optionally can be followed by " ASC" or " DESC" to specify ascending (the default) or descending order.
maxChars	Maximum number of characters to display in each column.
...	Additional expressions to add as part of a where clause in the query. Further arguments provided in ... will be added to the query separated by " and " as part of the where clause.

Value

Creates a table containing all the fields of the database table and opens it in the web browser for easy viewing.

Returns TRUE if the html file was written successfully.

Note

If viewing a table containing sequences, the sequences are purposefully not shown in the output.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

See Also

BrowseSequences

Examples

```
db <- system.file("extdata", "Bacteria_175seqs.sqlite", package="DECIPHER")
BrowseDB(db)
```

BrowseSequences *View Sequences In A Web Browser*

Description

Opens an html file in a web browser to show the sequences in a DNAStringSet.

Usage

```
BrowseSequences(myDNAStringSet,
                htmlFile = paste(tempdir(), "/dna.html", sep = ""),
                colorBases=TRUE,
                highlight=0,
                ...)
```

Arguments

myDNAStringSet	A DNAStringSet object of sequences.
htmlFile	Character string giving the location where the html file should be written.
colorBases	Logical specifying whether to color each type of base (A, C, G, and T) the same color.
highlight	Numeric specifying which sequence in the set to use for comparison or 0 to color every sequence (default).
...	Additional arguments to be passed directly to ConsensusSequence.

Details

Some web browsers cannot quickly display a large amount data, so it is recommended to use color = FALSE (the default) when viewing a large DNAStringSet.

Value

Creates an html file containing sequence data and opens it in a web browser for easy viewing. The viewer has the sequence name on the left, position legend on the top, number of characters on the right, and consensus sequence on the bottom.

Returns TRUE if the html file was written successfully.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

See Also

BrowseDB

Examples

```
db <- system.file("extdata", "Bacteria_175seqs.sqlite", package="DECIPHER")
dna <- SearchDB(db)
BrowseSequences(dna[1:5], colorBases=TRUE)
```

CalculateEfficiencyArray

Predicts the Hybridization Efficiency of Probe/Target Sequence Pairs

---

Description

Calculates the Gibb's free energy and hybridization efficiency of probe/target pairs at varying concentrations of the denaturant formamide.

Usage

```
CalculateEfficiencyArray(probe,
                         target,
                         FA = 0,
                         dGini = 1.96,
                         Po = 10^-2.0021,
                         m = 0.1731,
                         temp = 42,
                         deltaGrules = NULL)
```

Arguments

probe	A DNAStringSet object or character vector with pairwise-aligned probe sequences in 5' to 3' orientation.
target	A DNAStringSet object or character vector with pairwise-aligned target sequences in 5' to 3' orientation.
FA	A vector of one or more formamide concentrations (as percent v/v).
dGini	The initiation free energy. The default is 1.96 [kcal/mol].
Po	The effective probe concentration.
m	The m-value defining the linear relationship of denaturation in the presence of formamide.
temp	Equilibrium temperature in degrees Celsius.
deltaGrules	Free energy rules for all possible base pairings in quadruplets. If NULL, defaults to the parameters obtained using NimbleGen microarrays and a Linear Free Energy Model developed by Yilmaz *et al*.

Details

This function calculates the free energy and hybridization efficiency (HE) for a given formamide concentration ([FA]) using the linear free energy model given by:

$$HE = Po * exp[-(dG_0 + m * FA)/RT]/(1 + Po * exp[-(dG_0 + m * FA)/RT])$$

Probe and target input sequences must be entered in pairwise alignment, such as that given by pairwiseAlignment. Only "A", "C", "G", "T", and "-" characters are permitted in the probe sequence.

If deltaGrules is NULL then the rules defined in data(deltaGrules) are used.

Value

A matrix with the predicted Gibb's free energy (dG) and hybridization efficiency (HE) at each concentration of formamide ([FA]).

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

References

Yilmaz LS, Loy A, Wright ES, Wagner M, Noguera DR (2012) Modeling Formamide Denaturation of Probe-Target Hybrids for Improved Microarray Probe Design in Microbial Diagnostics. PLoS ONE 7(8): e43862. doi:10.1371/journal.pone.0043862.

See Also

deltaGrules

Examples

```
probes <- c("AAAAACGGGGAGCGGGGGGATACTG", "AAAAACTCAACCCGAGGAGCGGGGG")
targets <- c("CAACCCGGGGAGCGGGGGGATACTG", "TCGGGCTCAACCCGAGGAGCGGGGG")
result <- CalculateEfficiencyArray(probes, targets, FA=0:40)
dG0 <- result[, "dG_0"]
HE0 <- result[, "HybEff_0"]
plot(result[1, 1:40], xlab="[FA]", ylab="HE", main="Probe/Target # 1", type="l")
```

CalculateEfficiencyFISH

Predicts Thermodynamic Parameters of Probe/Target Sequence Pairs

---

Description

Calculates the Gibb's free energy, formamide melt point, and hybridization efficiency of probe/target (DNA/RNA) pairs.

Usage

```
CalculateEfficiencyFISH(probe,
                        target,
                        temp,
                        P,
                        ions,
                        FA,
                        batchSize = 1000)
```

Arguments

probe	A DNAStringSet object or character vector with unaligned probe sequences in 5' to 3' orientation.
target	A DNAStringSet object, RNAStringSet, or character vector with unaligned target or non-target sequences in 5' to 3' orientation. The DNA base Thymine will be treated the same as Uracil.
temp	Numeric specifying the hybridization temperature, typically 46 degrees Celsius.
P	Numeric giving the molar concentration of probes during hybridization.

ions	Numeric giving the molar sodium equivalent ionic concentration. Values may range between 0.01M and 1M. Note that salt correction is not available for thermodynamic rules of RNA/RNA interactions, which were determined at 1 molar concentration.
FA	Numeric concentration (as percent v/v) of the denaturant formamide in the hybridization buffer.
batchSize	Integer specifying the number of probes to simulate hybridization per batch. See the Description section below.

Details

Hybridization efficiency of pairwise probe/target (DNA/RNA) pairs is simulated *in silico*. Gibb's free energies are obtained from system calls to OligoArrayAux, which must be properly installed (see the Notes section below). Probe/target pairs are sent to OligoArrayAux in batches of `batchSize`, which prevents systems calls from being too many characters.

Value

A matrix of predicted hybridization efficiency (`HybEff`), formamide melt point (`FAm`), and free energy (`ddG1` and `dG1`) for each probe/target pair.

Note

The program OligoArrayAux (http://mfold.rna.albany.edu/?q=DINAMelt/OligoArrayAux) must be installed in a location accessible by the system. For example, the following code should print the installed OligoArrayAux version when executed from the R console:

```
system("hybrid-min -V")
```

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

References

Coming Soon!

See Also

DesignProbes, TileSeqs

Examples

```
probe <- c("GGGCTTTCACATCAGACTTAAGAAACC", "CCCCACGCTTTCGCGCC")
target <- reverseComplement(DNAStringSet(probe))
# not run (must have OligoArrayAux installed first):
#CalculateEfficiencyFISH(probe, target, temp=46, P=250e-9, ions=1, FA=35)
```

CalculateEfficiencyPCR

Predicts Amplification Efficiency of Primer Sequences

Description

Calculates the amplification efficiency of primers from their hybridization efficiency and elongation efficiency at the target site.

Usage

```
CalculateEfficiencyPCR(primer,
                       target,
                       temp,
                       P,
                       ions,
                       batchSize = 1000,
                       taqEfficiency = TRUE,
                       maxDistance = 0.4,
                       maxGaps = 2,
                       processors = NULL)
```

Arguments

primer	A DNAStringSet object or character vector with unaligned primer sequences in 5' to 3' orientation.
target	A DNAStringSet object or character vector with unaligned target or non-target sequences in 5' to 3' orientation.
temp	Numeric specifying the annealing temperature used in the PCR reaction.
P	Numeric giving the molar concentration of primers in the reaction.
ions	Numeric giving the molar sodium equivalent ionic concentration. Values may range between 0.01M and 1M.
batchSize	Integer specifying the number of primers to simulate hybridization per batch. See the Description section below.
taqEfficiency	Logical determining whether to make use of elongation efficiency and maxDistance to increase predictive accuracy for *Taq* DNA Polymerase amplifying primers with mismatches near the 3' terminus. Note that this should be set to FALSE if using a high-fidelity polymerase with 3' to 5' exonuclease activity.
maxDistance	Numeric specifying the maximal fraction of mismatched base pairings on a rolling basis beginning from the 3' end of the primer. Only used if taqEfficiency is TRUE.
maxGaps	Integer specifying the maximum number of insertions or deletions (indels) in the primer/target alignment. Only used if taqEfficiency is TRUE.
processors	The number of processors to use, or NULL (the default) for all available processors.

Details

Amplification of pairwise primer/target pairs is simulated *in silico*. A complex model of hybridization is employed that takes into account the side reactions resulting in probe-folding, target-folding, and primer-dimer formation. The resulting hybridization efficiency is multiplied by the elongation efficiency to predict the overall efficiency of amplification.

Free energy is obtained from system calls to OligoArrayAux, which must be properly installed (see the Notes section below). Primer/target pairs are sent to OligoArrayAux in batches of `batchSize`, which prevents systems calls from being too many characters.

Value

A vector of predicted efficiencies for amplifying each primer/target pair of sequences.

Note

The program OligoArrayAux (http://mfold.rna.albany.edu/?q=DINAMelt/OligoArrayAux) must be installed in a location accessible by the system. For example, the following code should print the installed OligoArrayAux version when executed from the R console:

```
system("hybrid-min -V")
```

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

References

ES Wright et al. (2013) "Exploiting Extension Bias in PCR to Improve Primer Specificity in Ensembles of Nearly Identical DNA Templates." Environmental Microbiology, doi:10.1111/1462-2920.12259.

See Also

DesignPrimers, TileSeqs

Examples

```
primers <- c("AAAAACGGGGAGCGGGGGG", "AAAAACTCAACCCGAGGAGCGCGT")
targets <- reverseComplement(DNAStringSet(primers))
# not run (must have OligoArrayAux installed first):
#CalculateEfficiencyPCR(primers, targets, temp=75, P=4e-7, ions=.225)
```

ConsensusSequence *Create A Consensus Sequence*

Description

Forms a consensus sequence representing a set of sequences.

Usage

```
ConsensusSequence(myDNAStringSet,
                  threshold = 0.05,
                  ambiguity = TRUE,
                  noConsensusChar = "N",
                  minInformation = 0.75,
                  ignoreNonBases = FALSE,
                  includeTerminalGaps = FALSE,
                  verbose = TRUE)
```

Arguments

myDNAStringSet	A DNAStringSet object of aligned sequences.
threshold	Maximum fraction of sequence information that may be lost in forming the consensus.
ambiguity	Logical specifying whether to consider ambiguity as split between their respective nucleotides. Degeneracy codes are specified in the IUPAC_CODE_MAP.
noConsensusChar	Single character from the DNA_ALPHABET giving the base to use when there is no consensus in a position.
minInformation	Minimum fraction of information required to form consensus in each position.
ignoreNonBases	Logical specifying whether to count gap ("-") or mask ("+") characters towards the consensus.
includeTerminalGaps	Logical specifying whether or not to include terminal gaps ("-" characters on each end of the sequence) into the formation of consensus.
verbose	Logical indicating whether to print the elapsed time upon completion.

Details

Two key parameters control the degree of consensus. The default threshold (0.05) indicates that at least 95% of sequence information will be represented by the consensus sequence. The default minInformation (0.75) specifies that at least 75% of sequences must contain the information in the consensus, otherwise the noConsensusChar is used.

If ambiguity = TRUE (the default) then degeneracy codes are split between their respective bases according to the IUPAC_CODE_MAP. For example, an "R" would count as half an "A" and half a "G". If ambiguity = FALSE then degeneracy codes are not considered in forming the consensus. If includeNonBases = TRUE (the default) then gap ("-") and mask ("+") characters are counted towards the consensus, otherwise they are omitted from development of the consensus.

Value

A DNAStringSet with a single consensus sequence.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

See Also

IdConsensus, Seqs2DB

Examples

```
dna <- DNAStringSet(c("ANGCT-","-ACCT-"))
ConsensusSequence(dna)
# returns "ANSCT-"
```

CreateChimeras *Creates Artificial Chimeras*

Description

Creates artificial random chimeras from a set of sequences.

Usage

```
CreateChimeras(myDNAStringSet,
               numChimeras = 10,
               numParts = 2,
               minLength = 80,
               maxLength = Inf,
               minChimericRegionLength = 30,
               randomLengths = TRUE,
               includeParents = TRUE,
               processors = NULL,
               verbose = TRUE)
```

Arguments

myDNAStringSet	A DNAStringSet object with aligned sequences.
numChimeras	Number of chimeras desired.
numParts	Number of chimeric parts from which to form a single chimeric sequence.
minLength	Minimum length of the complete chimeric sequence.
maxLength	Maximum length of the complete chimeric sequence.
minChimericRegionLength	Minimum length of the chimeric region of each sequence part.

randomLengths	Logical specifying whether to create random length chimeras in addition to random breakpoints.
includeParents	Whether to include the parents of each chimera in the output.
processors	The number of processors to use, or NULL (the default) for all available processors.
verbose	Logical indicating whether to display progress.

Details

Forms a set of random chimeras from the input set of (typically good quality) sequences. The chimeras are created by merging random sequences at random breakpoints. These chimeras can be used for testing the accuracy of the FindChimeras or other chimera finding functions.

Value

A DNAStringSet object containing chimeras. The names of the chimeras are specified as "parent #1 name [chimeric region] (distance from parent to chimera), ...".

If includeParents = TRUE then the parents of the the chimeras are included at the end of the result. The parents are made to be the same length as the chimera if randomLengths = TRUE. The names of the parents are specified as "parent #1 name [region] (distance to parent #2, ...)".

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

See Also

FindChimeras, Seqs2DB

Examples

```
db <- system.file("extdata", "Bacteria_175seqs.sqlite", package="DECIPHER")
dna <- SearchDB(db)
chims <- CreateChimeras(dna)
BrowseSequences(chims)
```

DB2FASTA *Export Database Sequences to FASTA File*

Description

Exports a database containing sequences to a FASTA formatted file of sequences.

Usage

```
DB2FASTA(file,
         dbFile,
         tblName = "DNA",
         identifier = "",
         limit = -1,
         replaceChar = NULL,
         orderBy = "row_names",
         append = FALSE,
         comments = FALSE,
         removeGaps = "none",
         chunkSize = 1e5,
         verbose = TRUE,
         ...)
```

Arguments

file	Character string giving the location where the FASTA file should be written.
dbFile	A SQLite connection object or a character string specifying the path to the database file.
tblName	Character string specifying the table in which to extract the data.
identifier	Optional character string used to narrow the search results to those matching a specific identifier. If "" then all identifiers are selected.
limit	Number of results to display. The default (-1) does not limit the number of results.
replaceChar	Optional character used to replace any sequence characters not present in the DNA_ALPHABET. If NULL (the default) then no replacement occurs and the sequences are exported identical to how they were upon import.
orderBy	Character string giving the column name for sorting the results. Defaults to the order of entries in the database. Optionally can be followed by " ASC" or " DESC" to specify ascending (the default) or descending order.
append	Logical indicating whether to append the results to the existing file.
comments	Logical specifying whether to add the value of any database fields into the FASTA record description separated by semicolons.
removeGaps	Determines how gaps are removed in the sequences. This should be (an unambiguous abbreviation of) one of "none", "all" or "common".
chunkSize	Number of lines of the file to write at a time.
verbose	Logical indicating whether to display status.
...	Additional expressions to add as part of a where clause in the query. Further arguments provided in ... will be added to the query separated by " and " as part of the where clause.

Value

Writes a FASTA formatted file containing the sequences in the database.

Returns TRUE if the file was written successfully.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

Examples

```
db <- system.file("extdata", "Bacteria_175seqs.sqlite", package="DECIPHER")
tf <- tempfile()
DB2FASTA(tf, db, limit=10)
file.show(tf)
unlink(tf)
```

---

deltaGrules *Free Energy of Hybridization of Probe/Target Quadruplets*

Description

The 8D array works with four adjacent base pairs of the probe and target sequence at a time. Each dimension has five elements defining the residue at that position ("A", "C", "G", "T", or "-"). The array contains the standard Gibb's free energy change of probe binding (dG, [kcal/mol]) for every quadruple base pairing.

Usage

```
data(deltaGrules)
```

Format

The format is: num [1:5, 1:5, 1:5, 1:5, 1:5, 1:5, 1:5, 1:5] -0.141 0 0 0 0 ... - attr(*, "dimnames")=List of 8 ..$: chr [1:5] "A" "C" "G" "T" ... ..$: chr [1:5] "A" "C" "G" "T" ... ..$: chr [1:5] "A" "C" "G" "T" ... ..$: chr [1:5] "A" "C" "G" "T" ... ..$: chr [1:5] "A" "C" "G" "T" ... ..$: chr [1:5] "A" "C" "G" "T" ... ..$: chr [1:5] "A" "C" "G" "T" ... ..$: chr [1:5] "A" "C" "G" "T" ...

Details

The first four dimensions correspond to the 4 probe positions from 5' to 3'. The fifth to eighth dimensions correspond to the 4 positions from 5' to 3' of the target sequence.

Source

Data obtained using NimbleGen microarrays and a Linear Free Energy Model developed by Yilmaz *et al*.

References

Yilmaz LS, Loy A, Wright ES, Wagner M, Noguera DR (2012) Modeling Formamide Denaturation of Probe-Target Hybrids for Improved Microarray Probe Design in Microbial Diagnostics. PLoS ONE 7(8): e43862. doi:10.1371/journal.pone.0043862.

Examples

```
data(deltaGrules)
# dG of probe = AGCT / target = A-CT pairing
deltaGrules["A", "G", "C", "T", "A", "-", "C", "T"]
```

DesignArray *Designs a set of DNA Microarray Probes for Detecting Sequences*

Description

Chooses the set of microarray probes maximizing sensitivity and specificity to each target consensus sequence.

Usage

```
DesignArray(myDNAStringSet,
            maxProbeLength = 24,
            minProbeLength = 20,
            maxPermutations = 4,
            numRecordedMismatches = 500,
            numProbes = 10,
            start = 1,
            end = NULL,
            maxOverlap = 5,
            hybridizationFormamide = 10,
            minMeltingFormamide = 15,
            maxMeltingFormamide = 20,
            minScore = -1e+12,
            processors = NULL,
            verbose = TRUE)
```

Arguments

myDNAStringSet	A DNAStringSet object of aligned consensus sequences.
maxProbeLength	The maximum length of probes, not including the poly-T spacer. Ideally less than 27 nucleotides.
minProbeLength	The minimum length of probes, not including the poly-T spacer. Ideally more than 18 nucleotides.
maxPermutations	The maximum number of probe permutations required to represent a target site. For example, if a target site has an 'N' then 4 probes are required because probes cannot be ambiguous. Typically fewer permutations are preferably because this requires less space on the microarray and simplifies interpretation of the results.
numRecordedMismatches	The maximum number of recorded potential cross-hybridizations for any target site.

`numProbes` The target number of probes on the microarray per input consensus sequence.

`start` Integer specifying the starting position in the alignment where potential forward primer target sites begin. Preferably a position that is included in most sequences in the alignment.

`end` Integer specifying the ending position in the alignment where potential reverse primer target sites end. Preferably a position that is included in most sequences in the alignment.

`maxOverlap` Maximum overlap in nucleotides between target sites on the sequence.

`hybridizationFormamide`
The formamide concentration (%, vol/vol) used in hybridization at 42 degrees Celsius. Note that this concentration is used to approximate hybridization efficiency of cross-amplifications.

`minMeltingFormamide`
The minimum melting point formamide concentration (%, vol/vol) of the designed probes. The melting point is defined as the concentration where half of the template is bound to probe.

`maxMeltingFormamide`
The maximum melting point formamide concentration (%, vol/vol) of the designed probes. Must be greater than the `minMeltingFormamide`.

`minScore` The minimum score of designed probes before exclusion. A greater `minScore` will accelerate the code because more target sites will be excluded from consideration. However, if the `minScore` is too high it will prevent any target sites from being recorded.

`processors` The number of processors to use, or `NULL` (the default) for all available processors.

`verbose` Logical indicating whether to display progress.

Details

The algorithm begins by determining the optimal length of probes required to meet the input constraints while maximizing sensitivity to the target consensus sequence at the specified hybridization formamide concentration. This set of potential target sites is then scored based on the possibility of cross-hybridizing to the other non-target sequences. The set of probes is returned with the minimum possibility of cross-hybridizing is chosen to represent each consensus sequence.

Value

A `data.frame` with the optimal set of probes matching the specified constraints. Each row lists the probe's target sequence (`name`), `start` position, `length` in nucleotides, start and end position in the sequence alignment, number of `permutations`, `score`, melt point in percent `formamide` at 42 degrees Celsius, hybridization efficiency (`hyb_eff`), target site, and probe(s). Probes are designed such that the stringency is determined by the equilibrium hybridization conditions and not subsequent washing steps.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

References

ES Wright et al. (2013) Identification of Bacterial and Archaeal Communities From Source to Tap. Water Research Foundation, Denver, CO.

See Also

Array2Matrix, NNLS

Examples

```
fas <- system.file("extdata", "Bacteria_175seqs.fas", package="DECIPHER")
dna <- readDNAStringSet(fas)
names(dna) <- 1:length(dna)
probes <- DesignArray(dna)
probes[1,]
```

DesignPrimers *Designs PCR Primers Targeting a Specific Group of Sequences*

Description

Assists in the design of primer sets targeting a specific group of sequences while minimizing the potential to cross-amplify other groups of sequences.

Usage

```
DesignPrimers(tiles,
              identifier = "",
              start = 1,
              end = NULL,
              minLength = 17,
              maxLength = 26,
              maxPermutations = 4,
              minCoverage = 0.9,
              minGroupCoverage = 0.2,
              annealingTemp = 64,
              P = 4e-07,
              monovalent = 0.07,
              divalent = 0.003,
              dNTPs = 8e-04,
              minEfficiency = 0.8,
              worstScore = -Inf,
              numPrimerSets = 0,
              minProductSize = 75,
              maxProductSize = 1200,
              maxSearchSize = 1500,
              batchSize = 1000,
```

```
                maxDistance = 0.4,
                primerDimer = 1e-07,
                ragged5Prime = TRUE,
                taqEfficiency = TRUE,
                induceMismatch = FALSE,
                processors = NULL,
                verbose = TRUE)
```

Arguments

tiles	A set of tiles representing each group of sequences, as in the format created by the function TileSeqs.
identifier	Optional character string used to narrow the search results to those matching a specific identifier. Determines the target group(s) for which primers will be designed. If "" then all identifiers are selected.
start	Integer specifying the starting position in the alignment where potential forward primer target sites begin. Preferably a position that is included in most sequences in the alignment.
end	Integer specifying the ending position in the alignment where potential reverse primer target sites end. Preferably a position that is included in most sequences in the alignment.
minLength	Integer providing the minimum length of primers to consider in the design.
maxLength	Integer providing the maximum length of primers to consider in the design, which must be less than or equal to the maxLength of tiles.
maxPermutations	Integer providing the maximum number of permutations considered as part of a forward or reverse primer set.
minCoverage	Numeric giving the minimum fraction of the target group's sequences that must be covered with the primer set.
minGroupCoverage	Numeric giving the minimum fraction of the target group that must have sequence information (not terminal gaps) in the region covered by the primer set.
annealingTemp	Numeric indicating the desired annealing temperature that will be used in the PCR experiment.
P	Numeric giving the molar concentration of primers in the reaction.
monovalent	The molar concentration of monovalent ([Na] and [K]) ions in solution that will be used to determine a sodium equivalent concentration.
divalent	The molar concentration of divalent ([Mg]) ions in solution that will be used to determine a sodium equivalent concentration.
dNTPs	Numeric giving the molar concentration of free nucleotides added to the solution that will be used to determine a sodium equivalent concentration.
minEfficiency	Numeric giving the minimum efficiency of hybridization desired for the primer set. Note that an efficiency of 99

worstScore	Numeric specifying the score cutoff to remove target sites from consideration. For example, a worstScore of -5 will remove all primer sets scoring below -5, although this may eventually result in no primer sets meeting the design criteria.
numPrimerSets	Integer giving the optimal number of primer sets (forward and reverse primer sets) to design. If set to zero then all possible forward and reverse primers are returned, but the primer sets minimizing potential cross-amplifications are not chosen.
minProductSize	Integer giving the minimum number of nucleotides desired in the PCR product.
maxProductSize	Integer giving the maximum number of nucleotides desired in the PCR product.
maxSearchSize	Integer giving the maximum number of nucleotides to search for false priming upstream and downstream of the expected binding site.
batchSize	Integer specifying the number of primers to simulate hybridization per batch that is passed to CalculateEfficiencyPCR.
maxDistance	Numeric specifying the maximal fraction of mismatched base pairings on a rolling basis beginning from the 3' end of the primer.
primerDimer	Numeric giving the maximum amplification efficiency of primer-dimer products.
ragged5Prime	Logical specifying whether the 5' end or 3' end of primer permutations targeting the same site should be varying lengths.
taqEfficiency	Logical determining whether to make use of elongation efficiency and maxDistance to increase predictive accuracy for *Taq* DNA Polymerase amplifying primers with mismatches near the 3' terminus. Note that this should be set to FALSE if using a high-fidelity polymerase with 3' to 5' exonuclease activity.
induceMismatch	Logical or integer specifying whether to induce a mismatch in the primer with the template DNA. If TRUE then a mismatch is induced at the 6th primer position. If an integer value is provided between 2 and 6 then a mismatch is induced in that primer position.
processors	The number of processors to use, or NULL (the default) for all available processors.
verbose	Logical indicating whether to display progress.

Details

Primers are designed for use with *Taq* DNA Polymerase to maximize sensitivity and specificity for the target group of sequences. The design makes use of *Taq*'s bias against certain 3' terminal mismatch types in order to increase specificity further than can be achieve with hybridization efficiency alone.

Primers are designed from a set of tiles to target each identifier while minimizing affinity for all other tiled groups. Arguments provide constraints that ensure the designed primer sets meet the specified criteria as well as being optimized for the particular experimental conditions. A search is conducted through all tiles in the same alignment position to estimate the chance of cross-amplification with a non-target group.

If numPrimers is greater than or equal to one then the set of forward and reverse primers that minimizes potential false positive overlap is returned. This will also initiate a thorough search

through all target sites upstream and downstream of the expected binding sites to ensure that the primers do not bind to nearby positions. Lowering the `maxSearchSize` will speed up the thorough search at the expense of potentially missing an unexpected target site. The number of possible primer sets assessed is increased with the size of `numPrimers`.

Value

A different `data.frame` will be returned depending on number of primer sets requested. If no primer sets are required then columns contain the forward and reverse primers for every possible position scored by their potential to amplify other identified groups. If one or more primer sets are requested then columns contain information for the optimal set of forward and reverse primers that could be used in combination to give the fewest potential cross-amplifications.

Note

The program OligoArrayAux (http://mfold.rna.albany.edu/?q=DINAMelt/OligoArrayAux) must be installed in a location accessible by the system. For example, the following code should print the installed OligoArrayAux version when executed from the R console:

```
system("hybrid-min -V")
```

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

References

ES Wright et al. (2013) "Exploiting Extension Bias in PCR to Improve Primer Specificity in Ensembles of Nearly Identical DNA Templates." Environmental Microbiology, doi:10.1111/1462-2920.12259.

See Also

CalculateEfficiencyPCR, TileSeqs

Examples

```
db <- system.file("extdata", "Bacteria_175seqs.sqlite", package="DECIPHER")
# not run (must have OligoArrayAux installed first):
#tiles <- TileSeqs(db, identifier=c("Acinetobacter","Pseudomonas"))
#primers <- DesignPrimers(tiles, identifier="Acinetobacter", start=280, end=420,
# minProductSize=50, numPrimerSets=1)
```

DesignProbes *Designs FISH Probes Targeting a Specific Group of Sequences*

Description

Assists in the design of single or dual probes targeting a specific group of sequences while minimizing the potential to cross-hybridize with other groups of sequences.

Usage

```
DesignProbes(tiles,
             identifier = "",
             start = 1,
             end = NULL,
             minLength = 17,
             maxLength = 26,
             maxPermutations = 4,
             minCoverage = 0.9,
             minGroupCoverage = 0.2,
             hybTemp = 46,
             P = 2.5e-07,
             Na = 1,
             FA = 35,
             minEfficiency = 0.5,
             worstScore = -Inf,
             numProbeSets = 0,
             batchSize = 1000,
             target = "SSU",
             verbose = TRUE)
```

Arguments

tiles	A set of tiles representing each group of sequences, as in the format created by the function TileSeqs.
identifier	Optional character string used to narrow the search results to those matching a specific identifier. Determines the target group(s) for which probes will be designed. If "" then all identifiers are selected.
start	Integer specifying the starting position in the alignment where potential target sites begin. Preferably a position that is included in most sequences in the alignment.
end	Integer specifying the ending position in the alignment where potential target sites end. Preferably a position that is included in most sequences in the alignment.
minLength	Integer providing the minimum length of probes to consider in the design.
maxLength	Integer providing the maximum length of probes to consider in the design, which must be less than or equal to the maxLength of tiles.

`maxPermutations`
: Integer providing the maximum number of probe permutations required to reach the desired coverage of a target site.

`minCoverage`
: Numeric giving the minimum fraction of the target group's sequences that must be covered by the designed probe(s).

`minGroupCoverage`
: Numeric giving the minimum fraction of the target group that must have sequence information (not terminal gaps) in the target site's region.

`hybTemp`
: Numeric specifying the hybridization temperature, typically `46` degrees Celsius.

`P`
: Numeric giving the molar concentration of probes during hybridization.

`Na`
: Numeric giving the molar sodium concentration in the hybridization buffer. Values may range between 0.01M and 1M. Note that salt correction from `1` molar is not available for the thermodynamic rules of RNA/RNA interactions.

`FA`
: Numeric concentration (as percent v/v) of the denaturant formamide in the hybridization buffer.

`minEfficiency`
: Numeric giving the minimum equilibrium hybridization efficiency desired for designed probe(s) at the defined experimental conditions.

`worstScore`
: Numeric specifying the score cutoff to remove target sites from consideration. For example, a `worstScore` of -5 will remove all probes scoring below -5, although this may eventually result in no probes meeting the design criteria.

`numProbeSets`
: Integer giving the optimal number of dual probe sets to design. If set to zero then all potential single probes are returned, and the probe sets minimizing potential false cross-hybridizations are not chosen.

`batchSize`
: Integer specifying the number of probes to simulate hybridization per batch that is passed to `CalculateEfficiencyFISH`.

`target`
: The target molecule used in the generation of tiles. Either `"SSU"` for the small-subunit rRNA, `"LSU"` for the large-subunit rRNA, or "Other". Used to determine the domain for `dG3` calculations, which is plus or minus 200 nucleotides of the target site if `"Other"`.

`verbose`
: Logical indicating whether to display progress.

Details

Probes are designed to maximize sensitivity and specificity to the target group(s) (`identifier`(s)). If `numProbeSets > 0` then that many pairs of probes with minimal cross-hybridization overlap are returned, enabling increased specificity with a dual-color approach.

Probes are designed from a set of tiles to target each `identifier` while minimizing affinity for all other tiled groups. Arguments provide constraints that ensure the designed probes meet the specified criteria as well as being optimized for the particular experimental conditions. A search is conducted through all tiles in the same alignment position to estimate the chance of cross-hybridization with a non-target group.

Two models are used in design, both of which were experimentally calibrated using denaturation profiles from 5 organisms belonging to all three domains of life. Probe lengths are chosen to meet the `minEfficiency` using a fast model of probe-target hybridization. Candidate probes are then

confirmed using a slower model that also takes into account probe-folding and target-folding. Finally, probes are scored for their inability to cross-hybridize with non-target groups by using the fast model and taking into account any mismatches.

Value

A different `data.frame` will be returned depending on number of primer sets requested. If no probe sets are required then columns contain the designed probes for every possible position scored by their potential to cross-hybridize with other identified groups. If one or more probe sets are requested then columns contain information for the optimal set of probes (probe one and probe two) that could be used in combination to give the fewest potential cross-hybridizations.

Note

The program OligoArrayAux (http://mfold.rna.albany.edu/?q=DINAMelt/OligoArrayAux) must be installed in a location accessible by the system. For example, the following code should print the installed OligoArrayAux version when executed from the R console:

```
system("hybrid-min -V")
```

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

References

Coming Soon!

See Also

CalculateEfficiencyFISH, TileSeqs

Examples

```
db <- system.file("extdata", "Bacteria_175seqs.sqlite", package="DECIPHER")
# not run (must have OligoArrayAux installed first):
#tiles <- TileSeqs(db, identifier=c("Acinetobacter","Pseudomonas"))
#probes <- DesignProbes(tiles, identifier="Acinetobacter", start=280, end=420)
```

DistanceMatrix *Calculate the Distance Between DNA Sequences*

Description

Calculates a distance matrix for a `DNAStringSet`. Each element of the distance matrix corresponds to the dissimilarity between two sequences in the `DNAStringSet`.

Usage

```
DistanceMatrix(myDNAStringSet,
               includeTerminalGaps = FALSE,
               penalizeGapLetterMatches = TRUE,
               penalizeGapGapMatches = FALSE,
               removeDuplicates = FALSE,
               correction = "none",
               processors = NULL,
               verbose = TRUE)
```

Arguments

myDNAStringSet	A DNAStringSet object of aligned sequences.
includeTerminalGaps	Logical specifying whether or not to include terminal gaps ("-" characters on each end of the sequence) into the calculation of distance.
penalizeGapLetterMatches	Logical specifying whether or not to consider gap-to-letter matches as mismatches.
penalizeGapGapMatches	Logical specifying whether or not to consider gap-to-gap matches as mismatches.
removeDuplicates	Logical specifying whether to remove any identical sequences from the input sequences before calculating distance. If FALSE (the default) then the distance matrix is calculated with the entire DNAStringSet provided as input.
correction	The substitution model used for distance correction. This should be (an unambiguous abbreviation of) one of "none" or "Jukes-Cantor".
processors	The number of processors to use, or NULL (the default) for all available processors.
verbose	Logical indicating whether to display progress.

Details

The uncorrected distance matrix represents the percent distance between each of the sequences in the DNAStringSet. Ambiguity can be represented using the characters of the IUPAC_CODE_MAP. For example, the distance between an 'N' and any other base is zero.

If includeTerminalGaps = FALSE then terminal gaps are not included in sequence length. This can be faster since only the positions common to each pair of sequences are compared. Similarly, if removeDuplicates = TRUE then the distance matrix will only represent unique sequences in the DNAStringSet. This is can be faster because less sequences need to be compared. For example, if only two sequences in the set are exact duplicates then one is removed and the distance is calculated on the remaining set. Note that the distance matrix can still contain values of 100% after removing duplicates because only exact duplicates are removed without taking into account ambiguous matches represented by the IUPAC_CODE_MAP or the treatment of gaps.

The elements of the distance matrix can be referenced by dimnames corresponding to the names of the DNAStringSet. Additionally, an attribute named "correction" specifying the method of correction used can be accessed using the function attr.

Value

A symmetric matrix where each element is the distance between the sequences referenced by the respective row and column. The `dimnames` of the matrix correspond to the `names` of the `DNAStringSet`. Sequences with no overlapping positions in the alignment are given a value of `NA`.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

See Also

IdClusters

Examples

```
# defaults compare intersection of internal ranges:
dna <- DNAStringSet(c("ANGCT-","-ACCT-"))
d <- DistanceMatrix(dna)
# d[1,2] is still 1 base in 4 = 0.25

# compare union of internal ranges:
dna <- DNAStringSet(c("ANGCT-","-ACCT-"))
d <- DistanceMatrix(dna, includeTerminalGaps=TRUE)
# d[1,2] is now 2 bases in 5 = 0.40

# compare the entire sequence ranges:
dna <- DNAStringSet(c("ANGCT-","-ACCT-"))
d <- DistanceMatrix(dna, includeTerminalGaps=TRUE,
                    penalizeGapGapMatches=TRUE)
# d[1,2] is now 3 bases in 6 = 0.50
```

FindChimeras *Find Chimeras In A Sequence Database*

Description

Finds chimeras present in a database of sequences. Makes use of a reference database of (presumed to be) good quality sequences.

Usage

```
FindChimeras(dbFile,
             tblName = "DNA",
             identifier = "",
             dbFileReference,
             batchSize = 100,
             minNumFragments = 20000,
```

```
                tb.width = 5,
                multiplier = 20,
                minLength = 70,
                minCoverage = 0.6,
                overlap = 100,
                minSuspectFragments = 6,
                showPercentCoverage = FALSE,
                add2tbl = FALSE,
                maxGroupSize = -1,
                minGroupSize = 100,
                verbose = TRUE)
```

Arguments

dbFile	A SQLite connection object or a character string specifying the path to the database file to be checked for chimeric sequences.
tblName	Character string specifying the table in which to check for chimeras.
identifier	Optional character string used to narrow the search results to those matching a specific identifier. If "" then all identifiers are selected.
dbFileReference	A SQLite connection object or a character string specifying the path to the reference database file of (presumed to be) good quality sequences. A 16S reference database is available from DECIPHER.cee.wisc.edu.
batchSize	Number sequences to tile with fragments at a time.
minNumFragments	Number of suspect fragments to accumulate before searching through other groups.
tb.width	A single integer [1..14] giving the number of nucleotides at the start of each fragment that are part of the trusted band.
multiplier	A single integer specifying the multiple of fragments found out-of-group greater than fragments found in-group in order to consider a sequence a chimera.
minLength	Minimum length of a chimeric region in order to be considered as a chimera.
minCoverage	Minimum fraction of coverage necessary in a chimeric region.
overlap	Number of nucleotides at the end of the sequence that the chimeric region must overlap in order to be considered a chimera.
minSuspectFragments	Minimum number of suspect fragments belonging to another group required to consider a sequence a chimera.
showPercentCoverage	Logical indicating whether to list the percent coverage of suspect fragments in each chimeric region in the output.
add2tbl	Logical or a character string specifying the table name in which to add the result.
maxGroupSize	Maximum number of sequences searched in a group. A value of less than 0 means the search is unlimited.

minGroupSize	The minimum number of sequences in a group to be considered as part of the search for chimeras. May need to be set to a small value for reference database with mostly small groups.
verbose	Logical indicating whether to display progress.

Details

The algorithm works by finding suspect fragments that are uncommon in the group where the sequence belongs, but very common in another group where the sequence does not belong. Each sequence in the dbFile is tiled into short sequence segments called fragments. If the fragments are infrequent in their respective group in the dbFileReference then they are considered suspect. If enough suspect fragments from a sequence meet the specified constraints then the sequence is flagged as a chimera.

The default parameters are optimized for full-length 16S sequences (> 1,000 nucleotides). Shorter 16S sequences require two parameters that are different than the defaults: minLength = 40, and minSuspectFragments = 2.

Groups are determined by the identifier present in each database. For this reason, the groups in the dbFile should exist in the groups of the dbFileReference. The reference database is assumed to contain many sequences of only good quality.

If a reference database is not present then it is feasible to create a reference database by using the input database as the reference database. Removing chimeras from the reference database and then iteratively repeating the process can result in a clean reference database.

For non-16S sequences it may be necessary to optimize the parameters for the particular sequences. The simplest way to perform an optimization is to experiment with different input parameters on artificial chimeras such as those created using CreateChimeras. Adjusting input parameters until the maximum number of artificial chimeras are identified is the easiest way to determine new defaults.

Value

A data.frame containing only the sequences that meet the specifications for being chimeric. The chimera column contains information on the chimeric region and to which group it belongs. The row.names of the data.frame correspond to those of the sequences in dbFile.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

References

ES Wright et al. (2012) "DECIPHER: A Search-Based Approach to Chimera Identification for 16S rRNA Sequences." Applied and Environmental Microbiology, doi:10.1128/AEM.06516-11.

See Also

CreateChimeras, Add2DB

Examples

```
db <- system.file("extdata", "Bacteria_175seqs.sqlite", package="DECIPHER")
# It is necessary to set dbFileReference to the file path of the
# 16S reference database available from DECIPHER.cee.wisc.edu
chimeras <- FindChimeras(db, dbFileReference=db)
```

`FormGroups` *Forms Groups By Rank*

Description

Agglomerates sequences into groups in a certain size range based on taxonomic rank.

Usage

```
FormGroups(dbFile,
           tblName = "DNA",
           goalSize = 1000,
           minGroupSize = 500,
           maxGroupSize = 10000,
           add2tbl = FALSE,
           verbose = TRUE)
```

Arguments

`dbFile`	A SQLite connection object or a character string specifying the path to the database file.
`tblName`	Character string specifying the table where the rank information is located.
`goalSize`	Number of sequences required in each group to stop adding more sequences.
`minGroupSize`	Minimum number of sequences in each group required to stop trying to recombine with a larger group.
`maxGroupSize`	Maximum number of sequences in each group allowed to continue agglomeration.
`add2tbl`	Logical or a character string specifying the table name in which to add the result.
`verbose`	Logical indicating whether to print database queries and other information.

Details

Form groups uses the rank field in the `dbFile` table to group sequences with similar taxonomic rank. Requires that rank information be present in the `tblName`, such as that created when importing sequences from a GenBank file.

Beginning with the least common ranks, the algorithm agglomerates groups with similar ranks until the `goalSize` is reached. If the group size is below `minGroupSize` then further agglomeration is attempted with a larger group. If additional agglomeration results in a group larger than `maxGroupSize` then the agglomeration is undone so that the group is smaller.

Value

Returns a data.frame of rank and id for each group. If add2tbl is not FALSE then the tblName is updated with the group as the identifier.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

See Also

IdentifyByRank

Examples

```
db <- system.file("extdata", "Bacteria_175seqs.sqlite", package="DECIPHER")
g <- FormGroups(db, goalSize=10, minGroupSize=5, maxGroupSize=20)
```

IdClusters *Cluster Sequences By Distance*

Description

Groups the sequences represented by a distance matrix into clusters of similarity.

Usage

```
IdClusters(myDistMatrix,
           method = "UPGMA",
           cutoff = -Inf,
           showPlot = FALSE,
           asDendrogram = FALSE,
           myDNAStringSet = NULL,
           model = MODELS,
           add2tbl = FALSE,
           dbFile = NULL,
           processors = NULL,
           verbose = TRUE)
```

Arguments

myDistMatrix	A symmetric N x N distance matrix with the values of dissimilarity between N sequences.
method	An agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "complete", "single", "UPGMA", "average", "NJ" or "ML". (See details section below.)
cutoff	A vector with the maximum edge length separating the sequences in the same cluster. If asDendrogram=TRUE then only one cutoff may be specified.

showPlot	Logical specifying whether or not to plot the resulting dendrogram.
asDendrogram	Logical. If TRUE the object returned is of class dendrogram.
myDNAStringSet	DNAStringSet used in the creation of myDistMatrix. Only necessary if method is "ML".
model	One or more of the available MODELS of DNA evolution. Only necessary if method is "ML".
add2tbl	Logical or a character string specifying the table name in which to add the result.
dbFile	A connection to a SQLite database or character string giving the path to the database file. Only necessary if add2tbl is not FALSE.
processors	The number of processors to use, or NULL (the default) for all available processors.
verbose	Logical indicating whether to display progress.

Details

Groups the input sequences into clusters using a set dissimilarities representing the distance between N sequences. Initially a phylogenetic tree is formed using the specified method. Then each leaf (sequence) of the tree is assigned to a cluster based on its edge lengths to the other sequences.

A number of different clustering methods are provided. The method (complete assigns clusters using complete-linkage so that sequences in the same cluster are no more than cutoff percent apart. The method single assigns clusters using single-linkage so that sequences in the same cluster are within cutoff of at least one other sequence in the same cluster. UPGMA or average (the default) assigns clusters using average-linkage which is a compromise between the sensitivity of complete-linkage clustering to outliers and the tendency of single-linkage clustering to connect distant relatives that do not appear to be closely related.

NJ uses the Neighbor-Joining method proposed by Saitou and Nei that does not assume lineages evolve at the same rate (the molecular clock hypothesis). The NJ method is typically the most phylogenetically accurate of the above distance-based methods. ML creates a neighbor-joining tree and then iteratively maximizes the likelihood of the tree given the aligned sequences (myDNAStringSet). This is accomplished through a combination of optimizing edge lengths with Brent's method and improving tree topology with nearest-neighbor interchanges (NNIs). When method="ML", one or more MODELS of DNA evolution must be specified. If multiple models are given their parameters are optimized and the best model is automatically chosen based on BIC using the sample size defined as the number of variable sites in the DNA sequence.

If a add2tbl=TRUE then the resulting data.frame is added/updated into column(s) of the default table "DNA" in dbFile. If add2tbl is a character string then the result is added to the specified table name in dbFile. The added/updated column names are printed if verbose=TRUE.

Value

If asDendrogram=FALSE (the default), returns a data.frame with a column for each cutoff specified. The row.names of the data.frame correspond to the dimnames of myDistMatrix. Each one of N sequences is assigned to one of M clusters. If asDendrogram=TRUE, returns an object of class dendrogram that can be used for further manipulation and plotting. Leaves of the dendrogram are randomly colored by cluster number.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

References

Felsenstein, J. (1981) Evolutionary trees from DNA sequences: a maximum likelihood approach. *Journal of Molecular Evolution*, **17(6)**, 368-376

Saitou, N. and Nei, M. (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, **4(4)**, 406-425.

See Also

DistanceMatrix, Add2DB, MODELS

Examples

```
# using the matrix from the original paper by Saitou and Nei
m <- matrix(0,8,8)
m[2:8,1] <- c(7, 8, 11, 13, 16, 13, 17)
m[3:8,2] <- c(5, 8, 10, 13, 10, 14)
m[4:8,3] <- c(5, 7, 10, 7, 11)
m[5:8,4] <- c(8, 11, 8, 12)
m[6:8,5] <- c(5, 6, 10)
m[7:8,6] <- c(9, 13)
m[8,7] <- c(8)

# returns an object of class "dendrogram"
myClusters <- IdClusters(m, cutoff=10, method="NJ", showPlot=TRUE, asDendrogram=TRUE)

# example of specifying a cutoff
# returns a data frame
IdClusters(m, cutoff=c(2,6,10,20))
```

IdConsensus *Create Consensus Sequences by Groups*

Description

Forms a consensus sequence representing the sequences in each group.

Usage

```
IdConsensus(dbFile,
            tblName = "DNA",
            identifier = "",
            colName = "cluster",
            add2tbl = FALSE,
            verbose = TRUE,
            ...)
```

Arguments

dbFile	A SQLite connection object or a character string specifying the path to the database file.
tblName	Character string specifying the table in which to form consensus.
identifier	Optional character string used to narrow the search results to those matching a specific identifier. If "" then all identifiers are selected.
colName	Column containing the group name of each sequence.
add2tbl	Logical or a character string specifying the table name in which to add the result.
verbose	Logical indicating whether to display progress.
...	Additional arguments to be passed directly to ConsensusSequence.

Details

Creates a consensus sequence for each of the distinct groups defined in colName. The resulting DNAStringSet contains as many consensus sequences as there are groups in colName. For example, it is possible to create a set of consensus sequences with one consensus sequence for each "id" or "cluster".

Value

A DNAStringSet object containing the consensus sequence for each group. The names of the DNAStringSet contain the number of sequences and name of each group.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

See Also

Seqs2DB

Examples

```
db <- system.file("extdata", "Bacteria_175seqs.sqlite", package="DECIPHER")
con <- IdConsensus(db, colName="id")
BrowseSequences(con, colorBases=TRUE)
```

IdentifyByRank *Identify By Taxonomic Rank*

Description

Identifies sequences by a specific level of their taxonomic rank.

Usage

```
IdentifyByRank(dbFile,
               tblName = "DNA",
               level = 3,
               add2tbl = FALSE,
               verbose = TRUE)
```

Arguments

dbFile	A SQLite connection object or a character string specifying the path to the database file.
tblName	Character string specifying the table where the rank information is located.
level	Level of the taxonomic rank. (See details section below.)
add2tbl	Logical or a character string specifying the table name in which to add the result.
verbose	Logical indicating whether to print database queries and other information.

Details

Simply identifies a sequence by a specific level of its taxonomic rank. Requires that `rank` information be present in the `tblName`, such as that created when importing sequences from a GenBank file.

The input parameter `level` should be a non-zero integer giving the "level" of the taxonomic rank to choose as the identifier. Negative `level`s are interpreted as that many levels from the final level.

If the specified level of rank does not exist then the closest rank is chosen. This makes it possible to determine the lowest level classification (e.g., genus) by specifying `level = Inf`.

Value

A `data.frame` with the `rank` and corresponding identifier as `"id"`.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

See Also

FormGroups

Examples

```
db <- system.file("extdata", "Bacteria_175seqs.sqlite", package="DECIPHER")
ids <- IdentifyByRank(db)
```

IdLengths *Determine the Number of Bases, Nonbases, and Width of Each Sequence*

Description

Counts the number of bases (A, C, G, T) and ambiguities/degeneracies in each sequence.

Usage

```
IdLengths(dbFile,
          tblName = "DNA",
          identifier = "",
          add2tbl = FALSE,
          verbose = TRUE)
```

Arguments

dbFile	A SQLite connection object or a character string specifying the path to the database file.
tblName	Character string specifying the table where the sequences are located.
identifier	Optional character string used to narrow the search results to those matching a specific identifier. If "" then all identifiers are selected.
add2tbl	Logical or a character string specifying the table name in which to add the result.
verbose	Logical indicating whether to display progress.

Value

A `data.frame` with the number of `bases`, `nonbases`, and `width` of each sequence. The `width` is defined as the sum of `bases` and `nonbases` in each sequence. The `row.names` of the `data.frame` correspond to the "row_names" in the `tblName` of the `dbFile`.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

See Also

Add2DB

Examples

```
db <- system.file("extdata", "Bacteria_175seqs.sqlite", package="DECIPHER")
l <- IdLengths(db)
```

MaskAlignment *Masks Highly Variable Regions of An Alignment*

Description

Automatically masks poorly aligned regions of an alignment based on sequence conservation and gap frequency.

Usage

```
MaskAlignment(myDNAStringSet,
              windowSize = 5,
              threshold = 1,
              maxFractionGaps = 0.2,
              showPlot = FALSE)
```

Arguments

myDNAStringSet	A DNAStringSet object of aligned sequences.
windowSize	Integer value specifying the size of the center-point moving average to use in determining variable regions.
threshold	Numeric giving the average entropy in bits from 0 to 2 below which a region is masked.
maxFractionGaps	Numeric specifying the maximum faction of gaps in an alignment column to be masked.
showPlot	Logical specifying whether or not to show a plot of the positions that were kept or masked.

Details

Poorly aligned regions of a multiple sequence alignment may lead to incorrect results in downstream analyses. One method to mitigate their effects is to mask columns of the alignment that may be poorly aligned, such as highly-variable regions or regions with many insertions and deletions (gaps).

Highly variable regions are detected by their signature of having low information content. A moving average of windowSize nucleotides to the left and right of the center-point is applied to smooth noise in the information content signal along the sequence. Regions dropping below threshold bits or more than maxFractionGaps are masked in the returned alignment.

Value

A DNAMultipleAlignment object with masked columns where the input criteria are met.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

See Also

AlignSeqs, IdClusters

Examples

```
fas <- system.file("extdata", "Streptomyces_ITS_aligned.fas", package="DECIPHER")
dna <- readDNAStringSet(fas)
masked_dna <- MaskAlignment(dna, showPlot=TRUE)

# display only unmasked nucleotides for use in downstream analyses
not_masked <- as(masked_dna, "DNAStringSet")
BrowseSequences(not_masked)

# display only masked nucleotides that are covered by the mask
masked <- masked_dna
colmask(masked, append="replace", invert=TRUE) <- colmask(masked)
masked <- as(masked, "DNAStringSet")
BrowseSequences(masked)

# display the complete DNA sequence set including the mask
masks <- lapply(width(colmask(masked_dna)), rep, x="+")
masks <- unlist(lapply(masks, paste, collapse=""))
masked_dna <- replaceAt(dna, at=IRanges(colmask(masked_dna)), value=masks)
BrowseSequences(masked_dna)
```

MODELS *Available Models of DNA Evolution*

Description

The MODELS character vector contains the models of DNA evolution that can be used by IdClusters.

Usage

```
MODELS
```

Details

Six models of DNA evolution are available, with or without the discrete Gamma rates distribution. These are described in order of increasing number of parameters as follows:

JC69 (Jukes and Cantor, 1969) The simplest substitution model that assumes equal base frequencies (1/4) and equal mutation rates.

K80 (Kimura, 1980) Assumes equal base frequencies, but distinguishes between the rate of transitions and transversions.

T92 (Tamura, 1992) In addition to distinguishing between transitions and transversions, a parameter is added to represent G+C content bias.

F81 (Felsenstein, 1981) Assumes equal mutation rates, but allows all bases to have different frequencies.

HKY85 (Hasegawa, Kishino and Yano, 1985) Distinguishes transitions from transversions and allows bases to have different frequencies.

TN93 (Tamura and Nei, 1993) Allows for unequal base frequencies and distinguishes between transversions and the two possible types of transitions (i.e., A <-> G & C <-> T).

+G (Yang, 1993) Specifying a model+G4 adds a single parameter to any of the above models to relax the assumption of equal rates among sites in the DNA sequence. The single parameter specifies the shape of the Gamma Distribution. The continuous distribution is represented with 2-10 discrete rates and their respective probabilities as determined by the Laguerre Quadrature method (Felsenstein, 2001). For example, specifying a model+G8 would represent the continuous Gamma Distribution with eight rates and their associated probabilities.

References

Felsenstein, J. (1981). Evolutionary trees from DNA sequences: a maximum likelihood approach. Journal of Molecular Evolution, **17(6)**, 368-376.

Felsenstein, J. (2001). Taking Variation of Evolutionary Rates Between Sites into Account in Inferring Phylogenies. Journal of molecular evolution, **53(4-5)**, 447-455.

Hasegawa, M., Kishino H., Yano T. (1985). Dating of human-ape splitting by a molecular clock of mitochondrial DNA. Journal of Molecular Evolution, **22(2)**, 160-174.

Jukes, T. and Cantor C. (1969). Evolution of Protein Molecules. New York: Academic Press. pp. 21-132.

Kimura, M. (1980). A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. Journal of Molecular Evolution, **16(2)**, 111-120.

Tamura, K. (1992). Estimation of the number of nucleotide substitutions when there are strong transition-transversion and G+C content biases. Molecular Biology and Evolution, **9(4)**, 678-687.

Tamura, K. and Nei M. (1993). Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. Molecular Biology and Evolution, **10(3)**, 512-526.

Yang, Z. (1993). Maximum-likelihood estimation of phylogeny from DNA sequences when substitution rates differ over sites. Molecular Biology and Evolution, **10(6)**, 1396-1401.

See Also

IdClusters

Examples

```
MODELS
```

NNLS *Sequential Coordinate-wise Algorithm for the Non-negative Least Squares Problem*

Description

Consider the linear system $\mathbf{A}x = b$ where $\mathbf{A} \in R^{\mathrm{m\ x\ n}}$, $x \in R^{\mathrm{n}}$, and $b \in R^{\mathrm{m}}$. The technique of least squares proposes to compute x so that the sum of squared residuals is minimized. NNLS solves the least squares problem $\min ||\mathbf{A}x = b||^2$ subject to the constraint $x \geq 0$. This implementation of the Sequential Coordinate-wise Algorithm uses a sparse input matrix **A**, which makes it efficient for large sparse problems.

Usage

```
NNLS(A,
     b,
     precision = sqrt(.Machine$double.eps),
     processors = NULL,
     verbose = TRUE)
```

Arguments

A	List representing the sparse matrix with integer components i and j, numeric component x. The fourth component, dimnames, is a list of two components that contains the names for every row (component 1) and column (component 2).
b	Numeric matrix for the set of observed values. (See details section below.)
precision	The desired accuracy.
processors	The number of processors to use, or NULL (the default) for all available processors.
verbose	Logical indicating whether to display progress.

Details

The input b can be either a matrix or a vector of numerics. If it is a matrix then it is assumed that each column contains a set of observations, and the output x will have the same number of columns. This allows multiple NNLS problems using the same **A** matrix to be solved simultaneously, and greatly accelerates computation relative to solving each sequentially.

Value

A list of two components:

x	The matrix of non-negative values that best explains the observed values given by b.
res	A matrix of residuals given by $\mathbf{A}x - b$.

References

Franc, V., et al. (2005). Sequential coordinate-wise algorithm for the non-negative least squares problem. Computer Analysis of Images and Patterns, 407-414.

See Also

Array2Matrix, DesignArray

Examples

```
# unconstrained least squares:
A <- matrix(c(1, -3, 2, -3, 10, -5, 2, -5, 6), ncol=3)
b <- matrix(c(27, -78, 64), ncol=1)
x <- solve(crossprod(A), crossprod(A, b))

# Non-negative least squares:
w <- which(A > 0, arr.ind=TRUE)
A <- list(i=w[,"row"], j=w[,"col"], x=A[w],
          dimnames=list(1:dim(A)[1], 1:dim(A)[2]))
x_nonneg <- NNLS(A, b)

# compare the unconstrained and constrained solutions:
cbind(x, x_nonneg$x)

# the input value "b" can also be a matrix:
b2 <- matrix(b, nrow=length(b), ncol=2) # repeat b in two columns
x_nonneg <- NNLS(A, b2) # solution is repeated in two output columns
```

SearchDB *Obtain Specific Sequences from A Database*

Description

Returns the set of sequences meeting the search criteria.

Usage

```
SearchDB(dbFile,
         tblName = "DNA",
         identifier = "",
         limit = -1,
         replaceChar = "-",
         nameBy="row_names",
         orderBy = "row_names",
         countOnly = FALSE,
         removeGaps = "none",
         verbose = TRUE,
         ...)
```

Arguments

dbFile	A SQLite connection object or a character string specifying the path to the database file.
tblName	Character string specifying the table where the sequences are located.
identifier	Optional character string used to narrow the search results to those matching a specific identifier. If "" then all identifiers are selected.
limit	Number of results to display. The default (-1) does not limit the number of results.
replaceChar	Optional character used to replace any characters of the sequence that are not present in the DNA_ALPHABET.
nameBy	Character string giving the column name for naming the DNAStringSet.
orderBy	Character string giving the column name for sorting the results. Defaults to the order of entries in the database. Optionally can be followed by " ASC" or " DESC" to specify ascending (the default) or descending order.
countOnly	Logical specifying whether to return only the number of sequences.
removeGaps	Determines how gaps are removed in the sequences. This should be (an unambiguous abbreviation of) one of "none", "all" or "common".
verbose	Logical indicating whether to display queries as they are sent to the database.
...	Additional expressions to add as part of a where clause in the query. Further arguments provided in ... will be added to the query separated by " and " as part of the where clause.

Details

If RNA is present in the database then all U's are converted to T's before creating the DNAStringSet.

Value

A DNAStringSet with the sequences that meet the specified criteria. The names of the DNAStringSet correspond to the value in the "row_names" column of the database.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

See Also

Seqs2DB, DB2FASTA

Examples

```
db <- system.file("extdata", "Bacteria_175seqs.sqlite", package="DECIPHER")
dna <- SearchDB(db)
```

Seqs2DB *Add Sequences from Text File to Database*

Description

Adds sequences to a database.

Usage

```
Seqs2DB(seqs,
        type,
        dbFile,
        identifier,
        tblName = "DNA",
        chunkSize = 1e5,
        replaceTbl = FALSE,
        verbose = TRUE)
```

Arguments

seqs	Either a character string specifying the file path to the file containing the sequences, or a DNAStringSet object.
type	The type of sequences being imported. This should be (an unambiguous abbreviation of) one of "FASTA", "GenBank", or "DNAStringSet".
dbFile	A SQLite connection object or a character string specifying the path to the database file. If the dbFile does not exist then a new database is created at this location.
identifier	Character string specifying the "id" to give the imported sequences in the database.
tblName	Character string specifying the table in which to add the sequences.
chunkSize	Number of lines of the seqs to read at a time.
replaceTbl	Logical. If FALSE (the default) then the sequences are appended to any already existing in the table. If TRUE then any sequences already in the table are overwritten.
verbose	Logical indicating whether to display each query as it is sent to the database.

Details

Sequences are imported into the database in chunks of lines specified by chunkSize. The sequences can then be identified by searching the database for the identifier provided. Sequences are added to the database verbatim, so that no sequence information is lost when the sequences are exported from the database.

Value

The total number of sequences in the database table is returned after import.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

See Also

SearchDB, DB2FASTA

Examples

```
gen <- system.file("extdata", "Bacteria_175seqs.gen", package="DECIPHER")
dbConn <- dbConnect(SQLite(), ":memory:")
Seqs2DB(gen, "GenBank", dbConn, "Bacteria")
BrowseDB(dbConn)
dbDisconnect(dbConn)
```

TerminalChar *Determine the Number of Terminal Characters*

Description

Counts the number of terminal characters for every sequence in a DNAStringSet. Terminal characters are defined as a specific character repeated at the beginning and end of a sequence.

Usage

```
TerminalChar(myDNAStringSet,
             char = "-")
```

Arguments

myDNAStringSet	A DNAStringSet object of sequences.
char	A single character giving the terminal character to count.

Value

A matrix containing the results for each sequence in its respective row. The first column contains the number of leading char, the second contains the number of trailing char, and the third contains the total number of characters in between.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

See Also

IdLengths

Examples

```
db <- system.file("extdata", "Bacteria_175seqs.sqlite", package="DECIPHER")
dna <- SearchDB(db)
t <- TerminalChar(dna)
```

TileSeqs *Form a Set of Tiles for Each Group of Sequences.*

Description

Creates a set of tiles that represent each group of sequences in the database for downstream applications.

Usage

```
TileSeqs(dbFile,
         tblName = "DNA",
         identifier = "",
         minLength = 26,
         maxLength = 27,
         maxTilePermutations = 10,
         minCoverage = 0.9,
         add2tbl = FALSE,
         verbose = TRUE,
         ...)
```

Arguments

dbFile	A SQLite connection object or a character string specifying the path to the database file.
tblName	Character string specifying the table of sequences to use for forming tiles.
identifier	Optional character string used to narrow the search results to those matching a specific identifier. If "" then all identifiers are selected.
minLength	Integer providing the minimum number of nucleotides in each tile. Typically the same or slightly less than maxLength.
maxLength	Integer providing the maximum number of nucleotides in each tile. Tiles are designed primarily for this length, which should ideally be slightly greater than the maximum length of oligos used in downstream functions.
maxTilePermutations	Integer specifying the maximum number of tiles in each target site.
minCoverage	Numeric providing the fraction of coverage that is desired for each target site in the group. For example, a minCoverage of 0.9 request that additional tiles are added until 90% of the group is represented by the tile permutations.
add2tbl	Logical or a character string specifying the table name in which to add the result.
verbose	Logical indicating whether to display progress.
...	Additional arguments to be passed directly to SearchDB.

Details

This function will create a set of overlapping tiles representing each target site in an alignment of sequences. The most common tile permutations are added until the minimum group coverage is obtained.

Target sites with one more more tiles not meeting a set of requirements are marked with `misprime` equals `TRUE`. Requirements are a minimum group coverage, minimum length, and a maximum length. Additionally, tiles are required not to contain more than four runs of a single base or four di-nucleotide repeats.

Value

A `data.frame` with a row for each tile, and multiple columns of information. The `row_names` column gives the row number. The `start`, `end`, `start_aligned`, and `end_aligned` columns provide positioning of the tile in a consensus sequence formed from the group. The column `misprime` is a logical specifying whether the tile meets the specified constraints. The columns `width` and `id` indicate the tiles length and group of origin, respectively.

The `coverage` field gives the fraction of sequences containing the tile in the group that encompass the tiles start and end positions in the alignment, whereas the `groupCoverage` contains the fraction of all sequences in the group containing a tile at their respective target site. For example, if 10

The final column, `target_site`, provides the sequence of the tile.

Note

If `add2tbl` is `TRUE` then the tiles will be added to the database table that currently contains the sequences used for tiling. The added tiles may cause interference when querying a table of sequences. Therefore, it is recommended to add the tiles to their own table, for example, by using `add2tbl="Tiles"`.

Author(s)

Erik Wright <DECIPHER@cae.wisc.edu>

See Also

DesignPrimers

Examples

```
db <- system.file("extdata", "Bacteria_175seqs.sqlite", package="DECIPHER")
tiles <- TileSeqs(db, identifier="Pseudomonas")
```

Index