Package ‘packFinder’

December 4, 2024

Type Package

Title de novo Annotation of Pack-TYPE Transposable Elements

Version 1.19.0

Description Algorithm and tools for in silico pack-TYPE transposon discovery. Filters a given genome for properties unique to DNA transposons and provides tools for the investigation of returned matches. Sequences are input in DNAString format, and ranges are returned as a dataframe (in the format returned by as.dataframe(GRanges)).

License GPL-2

Encoding UTF-8

Depends R (>= 4.1.0)

Imports Biostrings, GenomicRanges, kmer, ape, methods, IRanges, S4Vectors

Suggests biomartr, knitr, rmarkdown, testthat, dendextend, biocViews, BiocCheck, BiocStyle

RoxygenNote 7.2.0

VignetteBuilder knitr

URL https://github.com/jackgisby/packFinder

BugReports https://github.com/jackgisby/packFinder/issues

biocViews Genetics, SequenceMatching, Annotation

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

git_branch devel

git_last_commit 7ce4093

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2024-12-03

Author Jack Gisby [aut, cre] (ORCID: <https://orcid.org/0000-0003-0511-8123>), Marco Catoni [aut] (ORCID: <https://orcid.org/0000-0002-3258-2522>)

Maintainer Jack Gisby <jackgisby@gmail.com>

Contents

arabidopsisThalianaRefseq . . . 2
blastAnalysis . . . 3
blastAnnotate . . . 5
collapseSeqs . . . 6
filterWildcards . . . 7
getPackSeqs . . . 8
getPacksFromCsv . . . 10
getPacksFromFasta . . . 11
getPacksFromGRanges . . . 12
getTsds . . . 13
identifyPotentialPackElements . . . 14
identifyTirMatches . . . 15
makeBlastDb . . . 17
packAlign . . . 18
packBlast . . . 20
packClust . . . 22
packFinder . . . 24
packMatches . . . 25
packSearch . . . 26
packsToCsv . . . 28
packsToFasta . . . 29
packsToGRanges . . . 30
readBlast . . . 31
readUc . . . 33
tirClust . . . 35

Index **37**

`arabidopsisThalianaRefseq`
Arabidopsis thaliana Refseq Genome Chromosome 3 Subset

Description

The chromosome 3 reference sequence for Arabidopsis thaliana as a `DNAStringSet`. Can be used as a test data set, as in the associated introduction vignette. The DNA sequence between bases 10,500,000 and 14,300,000 was extracted for use in this dataset.

Usage

```
data(arabidopsisThalianaRefseq)
```

Format

A `DNAStringSet` object containing a `DNAString` for Arabidopsis thaliana's chromosome 3 sequence.

Author(s)

Jack Gisby

Source

The Arabidopsis thaliana genome was downloaded from the NCBI refseq database on 20/SEP/2019, using getGenome, and chromosome 3 was extracted. The genome may also be accessed from the NCBI ftp server: ftp://ftp.ncbi.nlm.nih.gov/genomes.

See Also

getGenome, DNAStringSet, DNAString, packSearch

Examples

```
data(arabidopsisThalianaRefseq)

packMatches <- packSearch(
    Biostrings::DNAString("CACTACAA"),
    arabidopsisThalianaRefseq,
    elementLength = c(300, 3500),
    tsdLength = 3
)
```

---

blastAnalysis *BLAST Analysis of PackTYPE Elements*

Description

Run BLAST against user-specified databases of non-transposon and transposon-related proteins. Can be used to classify transposons based on their internal sequences.

Usage

```
blastAnalysis(
  packMatches,
  Genome,
  blastPath,
  protDb = NULL,
  autoDb = NULL,
  minE = 0.001,
  blastTask = "blastn-short",
  maxHits = 100,
  threads = 1,
  saveFolder = NULL,
  tirCutoff = 0
)
```

Arguments

packMatches	A dataframe of potential Pack-TYPE transposable elements, in the format given by packSearch. This dataframe is in the format produced by coercing a link[GenomicRanges:GRanges-object to a dataframe: data.frame(GRanges). Will be saved as a FASTA file for VSEARCH.
Genome	A DNAStringSet object containing sequences referred to in packMatches (the object originally used to predict the transposons packSearch).
blastPath	Path to the BLAST+ executable, or name of the BLAST+ application for Linux/MacOS users.
protDb	For assigning Pack-TYPE elements. Path to the blast database containing nucleotide or protein sequences to be matched against internal transposon sequences. Can be generated using BLAST+, or with link{makeBlastDb}.
autoDb	For assigning autonomous elements. Path to the blast database containing nucleotide or protein sequences to be matched against internal transposon sequences. Can be generated using BLAST+, or with link{makeBlastDb}.
minE	Blast results with e values greater than the specified cutoff will be ignored.
blastTask	Type of BLAST+ task, defaults to "blastn-short".
maxHits	Maximum hits returned by BLAST+ per query.
threads	Allowable number of threads to be utilised by BLAST+.
saveFolder	Directory to save BLAST+ results in; defaults to the working directory.
tirCutoff	How many bases to ignore at the terminal ends of the transposons to prevent hits to TIR sequences.

Value

No return value; executes BLAST+ to generate hits which are stored in a .blast file in the chosen directory.

Author(s)

Jack Gisby

References

For further information, see the NCBI BLAST+ application documentation and help pages (https://www.ncbi.nlm.nih.gov/pub

See Also

blastAnnotate, readBlast, packBlast

Examples

```
## Not run:
packMatches <- data(packMatches)
Genome <- data(arabidopsisThalianaRefseq)
```

```
blastAnalysis(packMatches, Genome,
    protDb = "C:/data/TAIR10_CDS",
    autoDb = "C:/data/TAIR10_transposons",
    blastPath = "C:/blast/bin/blastn.exe")

## End(Not run)
```

`blastAnnotate` *Functional Annotation of PackTYPE Elements*

Description

Uses hits, previously generated using blast, to annotate transposon hits. Transposons with non-redundant transposase hits are classed as autonomous ("auto"), while others are classed as "other" or "pack" based on whether the element has non-redundant hits to other proteins.

Usage

```
blastAnnotate(protHits, autoHits, packMatches)
```

Arguments

`protHits`	BLAST results for non-transposon related genes or proteins (as a `data.frame`). Generated using `blastAnalysis`.
`autoHits`	BLAST results for transposon related genes or proteins (as a `data.frame`). Generated using `blastAnalysis`.
`packMatches`	A dataframe of potential Pack-TYPE transposable elements, in the format given by `packSearch`. This dataframe is in the format produced by coercing a `link[GenomicRanges:GRanges-` object to a dataframe: `data.frame(GRanges)`. Will be saved as a FASTA file for VSEARCH.

Value

Returns the original `packMatches` dataframe, with the addition of a "classification" column containing one of the following values:

- auto - elements that match known transposases or transposon-related proteins are classified as autonomous elements
- pack - elements that match other proteins or genic sequences may be classified as Pack-TYPE elements
- other - elements that generate no significant hits

Note

Requires that the query ids in the protein and autonomous hits match the row names in packMatches.

Author(s)

Jack Gisby

References

For further information, see the NCBI BLAST+ application documentation and help pages (https://www.ncbi.nlm.nih.gov/pub

See Also

blastAnalysis, readBlast, packBlast

Examples

```
data("packMatches")

# read in some protein hits
p <- data.frame(
    query_id = c(2, 3),
    subject_id = c("prot", "hyp")
)

# read in some autonomous hits
a <- data.frame(
    query_id = c(3, 4),
    subject_id = c("transposase", "mutator")
)

blastAnnotate(p, a, packMatches)
```

---

collapseSeqs *Collapse Overlapping Sequences*

Description

The sequences predicted by packSearch often overlap, which may be due to the presence of closely interspersed elements or false TIR identification. In such cases, these elements can be combined using link[GenomicRanges:GRanges-class]{GRanges} in order to collapse overlapping elements, preventing over-estimation of transposon numbers. Also removes duplicate elements that have been generated in the case of multiple searches.

Usage

```
collapseSeqs(packMatches, Genome)
```

Arguments

`packMatches` A dataframe containing genomic ranges and names referring to sequences to be extracted. This dataframe is in the format produced by coercing a `link[GenomicRanges:GRanges-class` object to a dataframe: `data.frame(GRanges)`.

Must contain the following features:

- start - the predicted element's start base sequence position.
- end - the predicted element's end base sequence position.
- seqnames - character string referring to the sequence name in `Genome` to which `start` and `end` refer to.

`Genome` A DNAStringSet object containing sequences referred to in `packMatches` (the object originally used to predict the transposons `packSearch`).

Value

A set of non-overlapping transposon sequences in the format of the input dataframe.

Author(s)

Jack Gisby

See Also

`packSearch`, `link[GenomicRanges:GRanges-class]{GRanges}`

Examples

```
data(packMatches)
data(arabidopsisThalianaRefseq)

packMatches$start <- 1
packMatches$end <- 10

collapseSeqs(packMatches, arabidopsisThalianaRefseq)
```

`filterWildcards` *Remove Low Quality Sequences*

Description

Takes transposable elements detected by `packSearch` and removes those with large numbers of wildcard ("N") bases. Used by `packClust` and `packAlign` to remove poor quality sequences that may interfere with the quality of sequence alignments.

Usage

```
filterWildcards(packMatches, Genome, maxWildcards = 0.05)
```

Arguments

packMatches	A dataframe containing genomic ranges and names referring to sequences to be extracted.
Genome	The original set of sequences used to generate the transposons detected by packSearch.
maxWildcards	The maximal allowable proportion of wildcards in the sequence of each match (defaults to 0.05).

Value

The original dataframe, packMatches, with sequences removed that are found to contain a proportion of wildcards ("N") greater than that specified in maxWildcards.

Author(s)

Jack Gisby

See Also

packClust, packAlign, packMatches, data(arabidopsisThalianaRefseq)

Examples

```
data(arabidopsisThalianaRefseq)
data(packMatches)

filteredMatches <- filterWildcards(
    packMatches,
    arabidopsisThalianaRefseq,
    maxWildcards = 0.05
)
```

---

getPackSeqs *Extract Sequences of Pack-TYPE Elements*

Description

Method to quickly extract the sequences of predicted Pack-TYPE elements (as created by packSearch).

Usage

```
getPackSeqs(packMatches, Genome, output = "DNAStringSet")
```

Arguments

`packMatches` A dataframe containing genomic ranges and names referring to sequences to be extracted. This dataframe is in the format produced by coercing a `link[GenomicRanges:GRanges-class` object to a dataframe: `data.frame(GRanges)`.

Must contain the following features:

- start - the predicted element's start base sequence position.
- end - the predicted element's end base sequence position.
- seqnames - character string referring to the sequence name in `Genome` to which `start` and end refer to.

`Genome` A DNAStringSet object containing sequences referred to in `packMatches` (the object originally used to predict the transposons `packSearch`).

`output` The type of object to be returned:

- output = "DNAStringSet", returns a `DNAStringSet` object (default).
- output = "character", returns a `character` vector.

Value

transposon sequences extracted from `packMatches`. At default returns the sequences as a `DNAStringSet` or, if `output` is set to "character", returns a character vector.

Author(s)

Jack Gisby

See Also

`DNAStringSet`, `packSearch`, `DNAString`

Examples

```
data(arabidopsisThalianaRefseq)

packMatches <- packSearch(
    Biostrings::DNAString("CACTACAA"),
    arabidopsisThalianaRefseq,
    elementLength = c(300, 3500),
    tsdLength = 3
)

packSeqs <- getPackSeqs(packMatches, arabidopsisThalianaRefseq)
```

getPacksFromCsv — *Retrieve Saved packFinder Results (.csv)*

Description

Retrieves a dataframe of potential Pack-TYPE elements, previously saved using packSearch followed by packsToCsv.

Usage

```
getPacksFromCsv(file)
```

Arguments

file	File path to predicted transposons in CSV format.

Value

Dataframe in the format used by packSearch.

Author(s)

Jack Gisby

See Also

packsToCsv, read.table, packSearch

Examples

```
data(packMatches)

packMatches <- getPacksFromCsv(
    system.file("extdata", "packMatches.csv", package = "packFinder")
)
```

getPacksFromFasta *Retrieve Saved packFinder Results (.fasta)*

Description

Retrieves a dataframe of potential Pack-TYPE elements, previously saved using packSearch followed by packsToFasta. Parses the .fasta file and title field containing:

- seqnames - name of origin sequence
- start - transposon base start position on origin sequence
- end - transposon base end position on origin sequence
- width - width of transposon
- strand - direction of transposon ("+", "-" or "*")
- TSD - terminal site duplication (TSD) sequence

Usage

```
getPacksFromFasta(file)
```

Arguments

file	Path to predicted transposons in FASTA format.

Value

Dataframe in the format used by packSearch.

Author(s)

Jack Gisby

See Also

packsToFasta, packSearch

Examples

```
data(arabidopsisThalianaRefseq)
data(packMatches)

packMatches <- getPacksFromFasta(
    system.file("extdata", "packMatches.fasta", package = "packFinder")
)
```

getPacksFromGRanges *Retrieve packFinder Results from GRanges Object*

Description

A link[GenomicRanges:GRanges-class]{GRanges} object, potentially generated using packSearch and packsToGRanges, can be converted to a dataframe. If a GRanges object is supplied without TSD information, this can be calculated and appended to the final dataframe.

Usage

```
getPacksFromGRanges(packGRanges, Genome = NULL, tsdLength = NULL)
```

Arguments

packGRanges	link[GenomicRanges:GRanges-class]{GRanges} object to be coerced.
Genome	(optional) Sequences referred to by packGRanges.
tsdLength	(optional) Length of TSD sequences.

Value

Dataframe in the format used by packSearch. If Genome and tsdLength are supplied, then TSD sequences are retrieved and returned as part of the dataframe.

Author(s)

Jack Gisby

See Also

packsToGRanges, link[GenomicRanges:GRanges-class]{GRanges}, packSearch

Examples

```
data(packMatches)

GRangesObject <- packsToGRanges(packMatches)
packMatches <- getPacksFromGRanges(GRangesObject)
```

getTsds *Get Flanking Terminal Site Duplication Sequences*

Description

Gets the flanking TSD sequences of TIRs or predicted Pack-TYPE transposable elements. A dataframe of these elements can be in `tirMatches`.

Usage

```
getTsds(tirMatches, Genome, tsdLength, strand = "+", output = "character")
```

Arguments

`tirMatches`	A dataframe containing genomic ranges and names referring to TIR sequences or predicted Pack-TYPE transposable elements. Should be in the format used by `packSearch`.
`Genome`	A `DNAStringSet` object containing sequences referred to in `tirMatches`.
`tsdLength`	The length of the TSD region to be retrieved (integer).
`strand`	The strand of the TIR; "+" for forward, "-" for reverse. If the TSD sequences of transposable elements are being predicted, then this parameter can be left as default ("+"); if the TSD sequences of TIRs are being found then the strand direction must be supplied.
`output`	The type of object to be returned: • output = "DNAStringSet", returns a `DNAStringSet` object. • output = "character", returns a `character` vector (default).

Details

Called by `packSearch`. It is recommended to use the general pipeline function `packSearch` for identification of potential pack elements, which returns TSD sequences as a feature of results, however each stage may be called individually.

Value

Flanking TSD sequences as a vector of characters, or if output is specified as "DNAStringSet", TSD sequences will be returned as a `DNAStringSet` object.

Author(s)

Jack Gisby

See Also

`DNAStringSet`, `packSearch`, `tirMatches`

Examples

```
data(arabidopsisThalianaRefseq)
data(packMatches)

tsdSeqs <- getTsds(packMatches, arabidopsisThalianaRefseq, 3)
```

---

identifyPotentialPackElements

Pack Element Filtering

Description

Primary filtering stage for the `packSearch` algorithm. Identifies potential Pack-TYPE transposable elements based on proximity of matching inverted repeats and equality of TSD sequences.

Usage

```
identifyPotentialPackElements(
  forwardMatches,
  reverseMatches,
  Genome,
  elementLength,
  tsdMismatch = 0
)
```

Arguments

`forwardMatches`	A dataframe containing genomic ranges and names referring to forwards-facing TIR sequences and their respective TSD sequences.
`reverseMatches`	A dataframe containing genomic ranges and names referring to reverse-facing TIR sequences and their respective TSD sequences.
`Genome`	A DNAStringSet object containing the matches referred to in `forwardMatches` and `reverseMatches`
`elementLength`	A vector of two integers containing the minimum and maximum transposable element length.
`tsdMismatch`	An integer referring to the allowable mismatch (substitutions or indels) between a transposon's TSD sequences. `matchPattern` from Biostrings is used for pattern matching.

Details

Used by `packSearch` as a primariy filtering stage. Identifies matches likely to be transposons based on their TIR region, from `identifyTirMatches`, and their TSD region, from `getTsds`. It is recommended to use the general pipeline function `packSearch` for identification of potential pack elements, however each stage may be called individually. Note that only exact TSD matches are considered, so supplying long sequences for TSD elements may lead to false-negative results.

Value

A dataframe, packMatches, containing the locations of potential Pack-TYPE transposable elements in Genome.

Author(s)

Jack Gisby

See Also

packSearch

Examples

```
data(arabidopsisThalianaRefseq)

forwardMatches <- identifyTirMatches(
    Biostrings::DNAString("CACTACAA"),
    arabidopsisThalianaRefseq,
    tsdLength = 3,
    strand = "+"
)

reverseMatches <- identifyTirMatches(
    Biostrings::reverseComplement(Biostrings::DNAString("CACTACAA")),
    arabidopsisThalianaRefseq,
    tsdLength = 3,
    strand = "-"
)

packMatches <- identifyPotentialPackElements(
    forwardMatches,
    reverseMatches,
    arabidopsisThalianaRefseq,
    c(300, 3500)
)
```

---

identifyTirMatches *Identify Terminal Inverted Repeat Matches*

Description

Searches a DNAStringSet for potential TIRs based on sequence similarity.

Usage

```
identifyTirMatches(
  tirSeq,
  Genome,
  mismatch = 0,
  strand = "*",
  tsdLength,
  fixed = TRUE
)
```

Arguments

tirSeq	A DNAString object to be searched for.
Genome	A DNAStringSet object containing the DNAString objects to be searched.
mismatch	The allowable mismatch between tirSeq and a given slice of Genome. Includes indels.
strand	The directionality of the search string ("+" or "-"). Note that this does affect the search for tirSeqs, if you wish to search the reverse strand you should use the reverse complement of your sequence.
tsdLength	Integer referring to the length of the flanking TSD region.
fixed	Logical that will be passed to the 'fixed' argument of matchPattern. Determines the behaviour of IUPAC ambiguity codes when searching for TIR sequences.

Details

Called by packSearch. Used by packSearch as an initial filtering stage. matchPattern from Biostrings is used for pattern matching. It is recommended to use the general pipeline function packSearch for identification of potential pack elements, however each stage may be called individually.

Value

A dataframe, tirMatches, containing identified matches. The dataframe is in the format generated by packSearch.

Author(s)

Jack Gisby

See Also

DNAStringSet, packSearch, matchPattern, DNAString

Examples

```
data(arabidopsisThalianaRefseq)

forwardMatches <- identifyTirMatches(
    Biostrings::DNAString("CACTACAA"),
    arabidopsisThalianaRefseq,
    tsdLength = 3,
    strand = "+"
)
```

---

makeBlastDb *Make Blast Database*

Description

Generates a BLAST database to be queried. Required for identifying sequences using the BLAST+ software.

Usage

```
makeBlastDb(fastaFile, dbPath, blastPath, dbType = "nucl")
```

Arguments

fastaFile	FASTA file containing sequences to generate a BLAST database from.
dbPath	Path to save the BLAST database to.
blastPath	Path/name of BLAST program to use. Name of the application for Linux/MacOS, absolute path for the executable for windows users.
dbType	Type of BLAST database to create, e.g. "nucl" for a nucleotide database.

Value

No return value; generates a blast database in the chosen directory.

Author(s)

Jack Gisby

References

For further information, see the NCBI BLAST+ application documentation and help pages (https://www.ncbi.nlm.nih.gov/pub

See Also

packSearch

Examples

```
## Not run:
makeBlastDb("genes.fasta", "blastdb.db", "C:/blast.exe")

## End(Not run)
```

---

packAlign *Global Alignment with VSEARCH*

Description

A global pairwise alignment of pack-TYPE elements by sequence similarity. mIt may be useful to run packClust to identify groups of similar transposable elements, before generating alignments of each group.

Usage

```
packAlign(
  packMatches,
  Genome,
  identity = 0,
  threads = 1,
  identityDefinition = 2,
  maxWildcards = 0.05,
  saveFolder,
  vSearchPath = "vsearch"
)
```

Arguments

packMatches	A dataframe of potential Pack-TYPE transposable elements, in the format given by packSearch. This dataframe is in the format produced by coercing a link[GenomicRanges:GRanges-object to a dataframe: data.frame(GRanges). Will be saved as a FASTA file for VSEARCH.
Genome	A DNAStringSet object containing sequences referred to in packMatches (the object originally used to predict the transposons packSearch).
identity	The sequence identity of two transposable elements in packMatches required to be grouped into a cluster.
threads	The number of threads to be used by VSEARCH.
identityDefinition	The pairwise identity definition used by VSEARCH. Defaults to 2, the standard VSEARCH definition.
maxWildcards	The maximal allowable proportion of wildcards in the sequence of each match (defaults to 0.05).

saveFolder	The folder to save saveFolder files (uc, blast6out, FASTA)
vSearchPath	When the package is run on windows systems, the location of the VSEARCH executable file must be given; this should be left as default on Linux/MacOS systems.

Value

Saves alignment information, including a uc, blast6out and a pairwise alignment fasta file, to the specified location. Returns the uc summary file generated by the alignment.

Note

In order to align sequences using VSEARCH, the executable file must first be installed.

Author(s)

Jack Gisby

References

VSEARCH may be downloaded from https://github.com/torognes/vsearch, along with a manual documenting the program's parameters. See https://www.ncbi.nlm.nih.gov/pubmed/27781170 for further information.

See Also

tirClust, packClust, readBlast, readUc, filterWildcards, packSearch

Examples

```
data(arabidopsisThalianaRefseq)
data(packMatches)

# packAlign run on a Linux/MacOS system
## Not run:
    packAlign(packMatches, Genome)

## End(Not run)

# packAlign run on a Windows system
## Not run:
    packAlign(packMatches, Genome,
            vSearchPath = "path/to/vsearch/vsearch.exe")

## End(Not run)
```

packBlast *Pipeline for BLAST/Classification of PackTYPE Elements*

Description

Run BLAST against user-specified databases of non-transposon and transposon-relates proteins. Can be used to classify transposons based on their internal sequences.

Usage

```
packBlast(
  packMatches,
  Genome,
  blastPath,
  protDb,
  autoDb,
  minE = 0.001,
  blastTask = "blastn-short",
  maxHits = 100,
  threads = 1,
  saveFolder = NULL,
  tirCutoff = 100,
  autoCutoff = 1e-05,
  autoLength = 150,
  autoIdentity = 70,
  autoScope = NULL,
  protCutoff = 1e-05,
  protLength = 250,
  protIdentity = 70,
  protScope = 0.3
)
```

Arguments

packMatches	A dataframe of potential Pack-TYPE transposable elements, in the format given by packSearch. This dataframe is in the format produced by coercing a link[GenomicRanges:GRanges-object to a dataframe: data.frame(GRanges). Will be saved as a FASTA file for VSEARCH.
Genome	A DNAStringSet object containing sequences referred to in packMatches (the object originally used to predict the transposons packSearch).
blastPath	Path to the BLAST+ executable, or name of the BLAST+ application for Linux/MacOS users.
protDb	For assigning Pack-TYPE elements. Path to the blast database containing nucleotide or protein sequences to be matched against internal transposon sequences. Can be generated using BLAST+, or with link{makeBlastDb}.

autoDb	For assigning autonomous elements. Path to the blast database containing nucleotide or protein sequences to be matched against internal transposon sequences. Can be generated using BLAST+, or with link{makeBlastDb}.
minE	Blast results with e values greater than the specified cutoff will be ignored. This will be passed to BLASTN and applied to both transposon and non-transposon matches.
blastTask	Type of BLAST+ task, defaults to "blastn-short".
maxHits	Maximum hits returned by BLAST+ per query.
threads	Allowable number of threads to be utilised by BLAST+.
saveFolder	Directory to save BLAST+ results in; defaults to the working directory.
tirCutoff	How many bases to ignore at the terminal ends of the transposons to prevent hits to TIR sequences.
autoCutoff	Blast results for transposon-related elements will be filtered to ignore those with e values above the specified cutoff.
autoLength	Blast results for transposon-related elements containing hits with alignment lengths lower than this value will be ignored
autoIdentity	Blast results for transposon-related elements containing hits with sequence identities lower than this value will be ignored
autoScope	If specified, transposon-related blast results below the specified value will be ignored. Note that the dataframe of transposon matches must also be supplied to calculate scope. Scope is the proportion of the transposon's internal sequence occupied by the BLAST hit.
protCutoff	Blast results for genic/other matches will be filtered to ignore those with e values above the specified cutoff.
protLength	Blast results for genic/other matches containing hits with alignment lengths lower than this value will be ignored
protIdentity	Blast results for genic/other matches containing hits with sequence identities lower than this value will be ignored
protScope	If specified, genic/other blast matches below the specified value will be ignored. Note that the dataframe of transposon matches must also be supplied to calculate scope. Scope is the proportion of the transposon's internal sequence occupied by the BLAST hit.

Value

Returns the original packMatches dataframe, with the addition of a "classification" column containing one of the following values:

- auto - elements that match known transposases or transposon-related proteins are classified as autonomous elements
- pack - elements that match other proteins or genic sequences may be classified as Pack-TYPE elements
- other - elements that generate no significant hits

Author(s)

Jack Gisby

References

For further information, see the NCBI BLAST+ application documentation and help pages (https://www.ncbi.nlm.nih.gov/pub

See Also

blastAnalysis, packSearch, readBlast, blastAnnotate

Examples

```
## Not run:
packMatches <- data(packMatches)
Genome <- data(arabidopsisThalianaRefseq)

packBlast(packMatches, Genome,
    protDb = "C:/data/TAIR10_CDS",
    autoDb = "C:/data/TAIR10_transposons",
    blastPath = "C:/blast/bin/blastn.exe")

## End(Not run)
```

---

packClust — *Cluster Transposons with VSEARCH*

Description

Cluster potential pack-TYPE elements by sequence similarity. Resulting groups may be aligned with packAlign, or the clusters may be analysed with tirClust

Usage

```
packClust(
  packMatches,
  Genome,
  identity = 0.6,
  threads = 1,
  identityDefinition = 2,
  maxWildcards = 0.05,
  strand = "both",
  saveFolder = NULL,
  vSearchPath = "vsearch"
)
```

Arguments

- `packMatches` — A dataframe of potential Pack-TYPE transposable elements, in the format given by `packSearch`. This dataframe is in the format produced by coercing a `link[GenomicRanges:GRanges-` object to a dataframe: `data.frame(GRanges)`. Will be saved as a FASTA file for VSEARCH.
- `Genome` — A DNAStringSet object containing sequences referred to in `packMatches` (the object originally used to predict the transposons `packSearch`).
- `identity` — The sequence identity of two transposable elements in `packMatches` required to be grouped into a cluster.
- `threads` — The number of threads to be used by VSEARCH.
- `identityDefinition` — The pairwise identity definition used by VSEARCH. Defaults to 2, the standard VSEARCH definition.
- `maxWildcards` — The maximal allowable proportion of wildcards in the sequence of each match (defaults to `0.05`).
- `strand` — The strand direction (+, - or *) to be clustered.
- `saveFolder` — The folder to save output files (uc, blast6out, FASTA)
- `vSearchPath` — When the package is run on windows systems, the location of the VSEARCH executable file must be given; this should be left as default on Linux/MacOS systems.

Value

Saves cluster information, including a `uc` and `blast6out` file, to the specified location. Returns the given `packMatches` dataframe with an additional column, `cluster`, containing cluster IDs.

Note

In order to cluster sequences using VSEARCH, the executable file must first be installed.

Author(s)

Jack Gisby

References

VSEARCH may be downloaded from https://github.com/torognes/vsearch. See https://www.ncbi.nlm.nih.gov/pubmed/27781170 for further information.

See Also

`tirClust`, `packAlign`, `readBlast`, `readUc`, `filterWildcards`, `packSearch`

Examples

```
data(arabidopsisThalianaRefseq)
data(packMatches)

# packClust run on a Linux/MacOS system
## Not run:
    packClust(packMatches, Genome)

## End(Not run)

# packClust run on a Windows system
## Not run:
    packClust(packMatches, Genome,
            vSearchPath = "path/to/vsearch/vsearch.exe")

## End(Not run)
```

---

packFinder *packFinder: a package for the de novo Annotation of Pack-TYPE Transposable Elements*

Description

Algorithm and tools for in silico pack-TYPE transposon discovery. Filters a given genome for properties unique to DNA transposons and provides tools for the investigation of returned matches.

Main Algorithm

The goal of packFinder was to implement a simple tool for the prediction of potential Pack-TYPE elements. packFinder uses the following prior knowledge, provided by the user, to detect transposons:

- Terminal Inverted Repeat (TIR) Base Sequence
- Length of Terminal Site Duplication (TSD)
- Length of the Transposon

These features provide enough information to detect autonomous and pack-TYPE elements. For a transposon to be predicted by packFinder its TSD sequences must be identical to each other, its forward TIR sequence must match the base sequence provided and its reverse TIR sequence must match its reverse complement.

Transposons are therefore predicted by searching a given genome for these characteristics, and further analysis steps can reveal the nature of these elements - while the packFinder tool is sensitive for the detection of transposons, it does not discriminate between autonomous and Pack-TYPE elements. Autonomous elements will contain a transposase gene within the terminal inverted repeats and tend to be larger than their Pack-TYPE counterparts; pack-TYPE elements instead capture sections of host genomes. Following cluster analysis, BLAST can be used to discern which predicted elements are autonomous (transposase-containing) and with are true Pack-TYPE elements.

Workflow

An example of a standard workflow can be found using `browseVignettes(package = "packFinder")`. The primary functions include:

- packSearch - the packSearch algorithm uses simple pattern matching to detect DNA transposons.
- packClust - VSEARCH is used for clustering elements based on sequence similarity.

Having obtained the sequences of transposable elements in a given genome, it is recommende to carry out a BLAST search for each transposon cluster. This can identify which elements are likely autonomous, and which may be Pack-TYPE.

The `packFinder` functions report the position of elements in a given genome using a dataframe in the format of `packMatches`. This dataframe is in the format produced by coercing a `link[GenomicRanges:GRanges-class]` object to a dataframe: `data.frame(GRanges)`.

Author(s)

Jack Gisby

See Also

`packSearch`

---

packMatches *Sample packFinder Output*

Description

A sample output from `packSearch` with cluster information. This dataframe is in the format produced by coercing a `link[GenomicRanges:GRanges-class]{GRanges}` object to a dataframe: `data.frame(GRanges)`.

Usage

```
data(packMatches)
```

Format

A dataframe of 9 obs. and 7 variables.

Details

Was obtained from running packSearch on the Arabidopsis thaliana chromosome 3 reference sequence, followed by clustering using packClust. Contains the following features:

- start - the predicted element's start base sequence position.
- end - the predicted element's end base sequence position.
- seqnames - character string referring to the sequence name in Genome to which start and end refer to.

The dataset was generated as in the example below.

See Also

packSearch, data.frame, arabidopsisThalianaRefseq

Examples

```
data(arabidopsisThalianaRefseq)

packMatches <- packSearch(
    Biostrings::DNAString("CACTACAA"),
    arabidopsisThalianaRefseq,
    elementLength = c(300, 3500),
    tsdLength = 3
)
```

---

packSearch *packFinder Algorithm Pipeline*

Description

General use pipeline function for the Pack-TYPE transposon finding algorithm.

Usage

```
packSearch(
  tirSeq,
  Genome,
  mismatch = 0,
  elementLength,
  tsdLength,
  tsdMismatch = 0,
  fixed = TRUE
)
```

Arguments

`tirSeq`	A `DNAString` object containing the TIR sequence to be searched for.
`Genome`	A `DNAStringSet` object to be searched.
`mismatch`	The maximum edit distance to be considered for TIR matches (indels + substitutions). See `matchPattern` for details.
`elementLength`	The maximum element length to be considered, as a vector of two integers. E.g. `c(300, 3500)`
`tsdLength`	Integer referring to the length of the flanking TSD region.
`tsdMismatch`	An integer referring to the allowable mismatch (substitutions or indels) between a transposon's TSD sequences. `matchPattern` from Biostrings is used for pattern matching.
`fixed`	Logical that will be passed to the 'fixed' argument of `matchPattern`. Determines the behaviour of IUPAC ambiguity codes when searching for TIR sequences.

Details

Finds potential pack-TYPE elements based on:

- Similarity of TIR sequence to `tirSeq`
- Proximity of potential TIR sequences
- Directionality of TIR sequences
- Similarity of TSD sequences

The algorithm finds potential forward and reverse TIR sequences using `identifyTirMatches` and their associated TSD sequence via `getTsds`. The main filtering stage, `identifyPotentialPackElements`, filters matches to obtain a dataframe of potential PACK elements. Note that this pipeline does not consider the possibility of discovered elements being autonomous elements, so it is recommended to cluster and/or BLAST elements for further analysis. Furthermore, only exact TSD matches are considered, so supplying long sequences for TSD elements may lead to false-negative results.

Value

A dataframe, containing elements identified by thealgorithm. These may be autonomous or pack-TYPE elements. Will contain the following features:

- start - the predicted element's start base sequence position.
- end - the predicted element's end base sequence position.
- seqnames - character string referring to the sequence name in `Genome` to which `start` and `end` refer to.
- width - the width of the predicted element.
- strand - the strand direction of the transposable element. This will be set to "*" as the `packSearch` function does not consider transposons to have a direction - only TIR sequences. Passing the `packMatches` dataframe to `packClust` will assign a direction to each predicted Pack-TYPE element.

This dataframe is in the format produced by coercing a `link[GenomicRanges:GRanges-class]{GRanges}` object to a dataframe: `data.frame(GRanges)`. Downstream functions, such as `packClust`, use this dataframe to manipulate predicted transposable elements.

Note

This algorithm does not consider:

- Autonomous elements - autonomous elements will be predicted by this algorithm as there is no BLAST step. It is recommended that, after clustering elements using `packClust`, the user analyses each group to determine which predicted elements are autonomous and which are likely Pack-TYPE elements. Alternatively, databases such as Repbase (https://www.girinst.org/repbase/) supply annotations for autonomous transposable elements that can be used to filter autonomous matches.
- TSD Mismatches - if two TIRs do not have exact matches for their terminal site duplications they will be ignored. Supplying longer TSD sequences will likely lead to a lower false-positive rate, however may also cause a greater rate of false-negative results.

Pattern matching is done via `matchPattern`.

Author(s)

Jack Gisby

See Also

`identifyTirMatches`, `getTsds`, `identifyPotentialPackElements`, `packClust`, `packMatches`, `DNAStringSet`, `DNAString`, `matchPattern`

Examples

```
data(arabidopsisThalianaRefseq)

packMatches <- packSearch(
    Biostrings::DNAString("CACTACAA"),
    arabidopsisThalianaRefseq,
    elementLength = c(300, 3500),
    tsdLength = 3
)
```

---

packsToCsv *Save packFinder Results in CSV Format (.csv)*

Description

Saves a dataframe of potential Pack-TYPE elements, usually generated via `packSearch`. May be retrieved using `getPacksFromCsv`.

Usage

```
packsToCsv(packMatches, file)
```

Arguments

`packMatches` A dataframe containing genomic ranges and names referring to sequences to be extracted. Can be obtained from `packSearch` or generated from a `GRanges` object, after conversion to a dataframe. Must contain the following features:

- start - the predicted element's start base sequence position.
- end - the predicted element's end base sequence position.
- seqnames - character string referring to the sequence name in `Genome` to which `start` and `end` refer to.

`file` CSV file save path.

Value

Save location of csv file.

Author(s)

Jack Gisby

See Also

`getPacksFromCsv`, `write.table`, `packSearch`

Examples

```
data(packMatches)

packsToCsv(
    packMatches,
    system.file("extdata", "packMatches.csv", package = "packFinder")
)
```

---

packsToFasta *Save packFinder Results in FASTA Format (.fasta)*

Description

Saves a dataframe of potential Pack-TYPE elements, usually generated via `packSearch`. May be retrieved using `getPacksFromFasta`.

Usage

```
packsToFasta(packMatches, file, Genome)
```

Arguments

`packMatches` taframe containing genomic ranges and names referring to sequences to be extracted. Can be obtained from `packSearch` or generated from a `GRanges` object, after conversion to a dataframe. Must contain the following features:

- start - the predicted element's start base sequence position.
- end - the predicted element's end base sequence position.
- seqnames - character string referring to the sequence name in `Genome` to which `start` and `end` refer to.

`file` FASTA file save path.

`Genome` A DNAStringSet object containing sequences referred to in `packMatches` (the object originally used to predict the transposons `packSearch`).

Value

Save location of Fasta file.

Author(s)

Jack Gisby

See Also

`getPacksFromFasta`, `packSearch`

Examples

```
data(arabidopsisThalianaRefseq)
data(packMatches)

packsToFasta(
    packMatches,
    system.file("extdata", "packMatches.fasta", package = "packFinder"),
    arabidopsisThalianaRefseq
)
```

---

packsToGRanges *Export packFinder Results to a GRanges Object*

Description

A dataframe containing genomic ranges and names referring to sequences to be extracted, likely obtained from `packSearch`, can be converted to a GRanges object. Can be converted back to a dataframe using `getPacksFromGRanges`. Additional features, such as clusters and TSD sequences, will be included in the object as metadata columns.

Usage

```
packsToGRanges(packMatches)
```

Arguments

`packMatches` A dataframe containing genomic ranges and names referring to sequences to be extracted. Can be obtained from `packSearch` or generated from a `GRanges` object, after conversion to a dataframe. Must contain the following features:

- start - the predicted element's start base sequence position.
- end - the predicted element's end base sequence position.
- seqnames - character string referring to the sequence name in `Genome` to which `start` and end refer to.

Value

A GRanges object containing the ranges contained in `packMatches` and additional metadata columns. May be easily converted between dataframe and GRanges format for use in the `packFinder` package and `link[GenomicRanges:GRanges-class]{GRanges}` package. Note that most functions in the `packFinder` package require sequence ranges to be provided in dataframe format.

Author(s)

Jack Gisby

See Also

`getPacksFromGRanges`, `link[GenomicRanges:GRanges-class]{GRanges}`

Examples

```
data(packMatches)
packGRanges <- packsToGRanges(packMatches)
```

---

readBlast *Convert NCBI BLAST+ Files to Dataframe*

Description

Reads .blast6out files (NCBI Blast Format) generated by the VSEARCH clustering and alignment algorithms.

Usage

```
readBlast(
  file,
  minE = 1,
  length = 0,
  identity = 0,
  removeExactMatches = FALSE,
  scope = NULL,
  packMatches = NULL
)
```

Arguments

`file` The file path of the blast file.

`minE` Blast results with e values greater than the specified cutoff will be ignored.

`length` Blast results alignment lengths lower below this value will be ignored

`identity` Blast results with target sequence identities below this value will be ignored.

`removeExactMatches` If true, matches with 100 be ignored to prevent self-hits.

`scope` If specified, blast results below the specified value will be ignored. Note that the dataframe of transposon matches must also be supplied to calculate scope. Scope is the proportion of the transposon's internal sequence occupied by the BLAST hit.

`packMatches` taframe containing genomic ranges and names referring to sequences to be extracted. Can be obtained from `packSearch` or generated from a `GRanges` object, after conversion to a dataframe. Must contain the following features:

- start - the predicted element's start base sequence position.
- end - the predicted element's end base sequence position.
- seqnames - character string referring to the sequence name in `Genome` to which `start` and `end` refer to.

Details

blast6out file is tab-separated text file compatible with NCBI BLAST m8 and NCBI BLAST+ outfmt 6 formats. One cluster/alignment can be found for each line.

Value

A dataframe containing the converted .blast6out file. The file contains the following features:

- Query sequence ID
- Target sequence ID
- Percenty sequence identity
- Alignment length
- Number of mismatches

- Number of gaps
- Base position of alignment start in query sequence
- Base position of alignment end in query sequence
- Base position of alignment start in target sequence
- Base position of alignment end in target sequence
- E-value
- Bit score

Author(s)

Jack Gisby

References

For further information, see the NCBI BLAST+ application documentation and help pages (https://www.ncbi.nlm.nih.gov/pub
VSEARCH may be downloaded from https://github.com/torognes/vsearch; see https://www.ncbi.nlm.nih.gov/pubmed/27781170 for further information.

See Also

codeblastAnalysis, codeblastAnnotate, codepackAlign, coderead Uc, codepackClust

Examples

```
readBlast(system.file(
    "extdata",
    "packMatches.blast6out",
    package = "packFinder"
))
```

---

readUc *Convert .uc Files to Dataframe*

Description

Reads .uc files (USEARCH Cluster Format) generated by the VSEARCH clustering and alignment algorithms.

Usage

```
readUc(file, output = "cluster")
```

Arguments

`file` The file path of the .uc file.

`output` The type of analysis that was carried out to produce the .uc file.

- If output is specified as "cluster", VSEARCH clustering was carried out.
- If output is specified as "alignment", VSEARCH pairwise global alignment was carried out.

Note that clustering produces one "H" record for each sequence, and one "C" record for each cluster, while an alignment produces an "H" record for each alignment (see details).

Details

USEARCH cluster format is a tab separated text file that contains clustering and/or alignment information for a set of sequences. For each sequence a record type, "H, C or N", is provided providing information about the type of "hit" in the dataframe. These refer to:

- H - Hit - for alignments, indicates an identified alignment of two supplied sequences. For clustering, indicates the cluster assignment for a query.
- C - Cluster record - a record for each cluster generated.
- N - No hit - indicates that no cluster was assigned or no alignment was found with a target sequence. For clustering, a query with no hits becomes the centroid of a new cluster.

Additionally, for each record a "compressed alignment" is generated. This is the alignment represented in a compact format including the letters "M", "D", and "I". Before each letter, the number of consecutive columns of the given letter type is also given. The letter types are as follows:

- "M" - Match - Identical bases between the query and target sequence
- "D" - Deletion - A gap in the target sequence
- "I" - Insertion - A gap in the query sequence

An example of this would be "13M", referring to 13 consecutive matches between the query and target sequence.

Value

A dataframe containing the converted .uc file. The fields contained within are as follows:

- Record type - "H, C or N", see details for further information.
- Cluster designation (`output = "cluster"` only)
- Sequence length, or cluster size
- Percent identity to target
- The nucleotide strand (`output = "cluster"` only)
- A compressed alignment - see details for further information.
- ID of query sequence
- ID of target sequence ("H" records only)

Author(s)

Jack Gisby

References

VSEARCH may be downloaded from https://github.com/torognes/vsearch. See https://www.ncbi.nlm.nih.gov/pubmed/27781170 for further information.

See Also

codetirClust, codepackAlign, codereadBlast, codepackClust

Examples

```
readUc(system.file(
    "extdata",
    "packMatches.uc",
    package = "packFinder"
))
```

---

tirClust — *Analyse TIR Sequences of Pre-clustered Transposable Elements*

Description

Takes transposable elements clustered by VSEARCH, packClust, and produces consensus sequences for the terminal inverted repeats of each. Allows for the visualisation of TIR similarities between clusters for both forward and reverse strands.

Usage

```
tirClust(
  packMatches,
  Genome,
  tirLength = 25,
  plot = TRUE,
  plotSavePath = NULL,
  k = 5,
  output = "consensus"
)
```

Arguments

packMatches	A dataframe containing genomic ranges and names referring to sequences to be extracted. This dataframe is in the format produced by coercing a link[GenomicRanges:GRanges-class object to a dataframe: data.frame(GRanges). Must contain the following features:

- start - the predicted element's start base sequence position.
- end - the predicted element's end base sequence position.
- seqnames - character string referring to the sequence name in `Genome` to which `start` and `end` refer to.

`Genome` A DNAStringSet object containing sequences referred to in `packMatches` (the object originally used to predict the transposons `packSearch`).

`tirLength` The TIR size to be considered. Consensus sequences will be generated based on the first and last `tirLength` bases of a transposon.

`plot` Argument specifying whether the TIR consensus sequences should be plottted as a dendrogram.

`plotSavePath` File path for the dendrogram plot. If unspecified, the dendrogram plot is not saved.

`k` The k-mer size to be used for calculating a distance matrix between TIR consensus sequences. See `kdistance`. Larger word sizes will not be suitable for longer TIR sequences, due to processing time required. Additionally, k must be greater than the TIR sequence length.

`output` Controls the output of `tirClust`. If output is specified as "consensus", the consensus sequences of each TIR cluster will be returned; else, if output is specified as "dendrogram", a dendrogram object will be returned for creation of customisable plots.

Value

If `output` is specified as "consensus" (default), returns a list of consensus sequences for each cluster specified in `packMatches` as a `DNAStringSet`. Else if `output` is specified as "dendrogram", returns a dendrogram object used to create hierarchical clustering diagrams.

Author(s)

Jack Gisby

See Also

codepackClust, codepackAlign, `kdistance`, `DNAStringSet`, `as.alignment`, `packSearch`

Examples

```
data(arabidopsisThalianaRefseq)
data(packMatches)

tirClust(packMatches, arabidopsisThalianaRefseq)
```

Index

∗ **datasets**
arabidopsisThalianaRefseq, 2
packMatches, 25

arabidopsisThalianaRefseq, 2, *26*
as.alignment, *36*

blastAnalysis, 3, *5, 6*, *22*, *33*
blastAnnotate, *4*, 5, *22*, *33*

collapseSeqs, 6

data.frame, *26*
DNAString, *2, 3*, *9*, *16*, *27, 28*
DNAStringSet, *2, 3*, *9*, *13*, *15, 16*, *27, 28*, *36*

filterWildcards, 7, *19*, *23*

getGenome, *3*
getPackSeqs, 8
getPacksFromCsv, 10, *28, 29*
getPacksFromFasta, 11, *29, 30*
getPacksFromGRanges, 12, *30, 31*
getTsds, 13, *14*, *27, 28*
GRanges, *29–32*

identifyPotentialPackElements, 14, *27, 28*
identifyTirMatches, *14*, 15, *27, 28*

kdistance, *36*

makeBlastDb, 17
matchPattern, *14*, *16*, *27, 28*

packAlign, *7, 8*, 18, *22, 23*, *33*, *35, 36*
packBlast, *4*, *6*, 20
packClust, *7, 8*, *18, 19*, 22, *26–28*, *33*, *35, 36*
packFinder, 24
packMatches, *25*, 25, *28*
packSearch, *3–14*, *16–20*, *22, 23*, *25, 26*, 26, *28–32*, *36*
packsToCsv, *10*, 28
packsToFasta, *11*, 29
packsToGRanges, *12*, 30

read.table, *10*
readBlast, *4*, *6*, *19*, *22, 23*, 31, *35*
readUc, *19*, *23*, *33*, 33

tirClust, *19*, *22, 23*, *35*, 35

write.table, *29*