Package 'alakazam'

August 29, 2016

Type Package

Version 0.2.5

Date 2016-08-05

Title Immunoglobulin Clonal Lineage and Diversity Analysis

Description Provides immunoglobulin (Ig) sequence lineage reconstruction, diversity profiling, and amino acid property analysis. Also provides core functionality for other tools in the Change-O suite.

License CC BY-NC-SA 4.0

URL http://alakazam.readthedocs.org

BugReports https://bitbucket.org/kleinstein/alakazam/issues

LazyData true

BuildVignettes true

VignetteBuilder knitr

SystemRequirements C++11

Depends R (>= 3.1.2), ggplot2 (>= 2.0.0)

Imports data.table (>= 1.9.4), dplyr, graphics, grid, igraph (>= 1.0.0), lazyeval, methods, Rcpp (>= 0.12.3), scales, seqinr, stats, stringi, utils

LinkingTo Rcpp

Suggests knitr, rmarkdown, testthat

RoxygenNote 5.0.1

Collate 'Alakazam.R' 'AminoAcids.R' 'Classes.R' 'Core.R' 'Data.R' 'Diversity.R' 'Gene.R' 'Lineage.R' 'RcppExports.R' 'Sequence.R' 'Topology.R'

NeedsCompilation yes

Author Jason Vander Heiden [aut, cre],
Namita Gupta [aut],
Susanna Marquez [ctb],
Daniel Gadala-Maria [ctb],
Roy Jiang [ctb],
Steven Kleinstein [aut, cph]

Maintainer Jason Vander Heiden <jason.vanderheiden@yale.edu>

Repository CRAN

Date/Publication 2016-08-06 00:19:53

R topics documented:

ABBREV_AA *Amino acid abbreviation translations*

Description

Mappings of amino acid abbreviations.

Usage

```
ABBREV_AA
```

Format

Named character vector defining single-letter character codes to three-letter abbreviation mappings.

Examples

```
aa <- c("Ala", "Ile", "Trp")
translateStrings(aa, ABBREV_AA)
```

alakazam *The alakazam package*

Description

alakazam in a member of the Change-O suite of tools and serves five main purposes:

- Providing core functionality for other R packages in the Change-O suite. This includes common tasks such as file I/O, basic DNA sequence manipulation, and interacting with V(D)J segment and gene annotations.
- Providing an R interface for interacting with the output of the pRESTO tool suite.
- Performing lineage reconstruction on clonal populations of immunoglobulin (Ig) sequences.
- Performing clonal abundance and diversity analysis on lymphocyte repertoires.
- Performing physicochemical property analyses of lymphocyte receptor sequences.

For additional details regarding the use of the alakazam package see the vignettes:
`browseVignettes("alakazam")`

File I/O

- readChangeoDb: Input Change-O style files.
- writeChangeoDb: Output Change-O style files.

Sequence cleaning

- maskSeqEnds: Mask ragged ends.
- maskSeqGaps: Mask gap characters.
- collapseDuplicates: Remove duplicate sequences.

Lineage reconstruction

- makeChangeoClone: Clean sequences for lineage reconstruction.
- buildPhylipLineage: Perform lineage reconstruction of Ig sequences.

Lineage topology analysis

- tableEdges: Tabulate annotation relationships over edges.
- testEdges: Significance testing of annotation edges.
- testMRCA: Significance testing of MRCA annotations.
- summarizeSubtrees: Various summary statistics for subtrees.
- plotSubtrees: Plot distributions of summary statistics for a population of trees.

Diversity analysis

- countClones: Calculate clonal abundance.
- estimateAbundance: Infer complete clonal abundance distribution with confidence intervals.
- rarefyDiversity: Generate clonal diversity curves.
- testDiversity: Test significance of clonal diversity scores.
- plotAbundance: Plot clone size distribution as a rank-abundance curve.
- plotDiversityCurve: Plot clonal diversity curves.

Ig and TCR sequence annotation

- countGenes: Calculate Ig and TCR allele, gene and family usage.
- extractVRegion: Extract CDRs and FWRs sub-sequences.
- getAllele: Get V(D)J allele names.
- getGene: Get V(D)J gene names.
- getFamily: Get V(D)J family names.

Sequence distance calculation

- seqDist: Calculate Hamming distance between two sequences.
- seqEqual: Test two sequences for equivalence.
- pairwiseDist: Calculate a matrix of pairwise Hamming distances for a set of sequences.
- pairwiseEqual: Calculate a logical matrix of pairwise equivalence for a set of sequences.

Amino acid propertes

- translateDNA: Translate DNA sequences to amino acid sequences.
- aminoAcidProperties: Calculate various physicochemical properties of amino acid sequences.
- countPatterns: Count patterns in sequences.

General data manipulation

- translateStrings: Perform multiple string replacement operations.

References

1. Vander Heiden JA, Yaari G, et al. pRESTO: a toolkit for processing high-throughput sequencing raw reads of lymphocyte receptor repertoires. Bioinformatics. 2014 30(13):1930-2.
2. Stern JNH, Yaari G, Vander Heiden JA, et al. B cells populating the multiple sclerosis brain mature in the draining cervical lymph nodes. Sci Transl Med. 2014 6(248):248ra107.
3. Wu Y-CB, et al. Influence of seasonal exposure to grass pollen on local and peripheral blood IgE repertoires in patients with allergic rhinitis. J Allergy Clin Immunol. 2014 134(3):604-12.
4. Gupta NT, Vander Heiden JA, et al. Change-O: a toolkit for analyzing large-scale B cell immunoglobulin repertoire sequencing data. Under review.

aliphatic *Calculates the aliphatic index of amino acid sequences*

Description

aliphatic calculates the aliphatic index of amino acid sequences using the method of Ikai. Non-informative positions are excluded, where non-informative is defined as any character in c("X", "-", ".", "*").

Usage

```
aliphatic(seq, normalize = TRUE)
```

Arguments

seq	vector of strings containing amino acid sequences.
normalize	if TRUE then divide the aliphatic index of each amino acid sequence by the number of informative positions. Non-informative position are defined by the presence any character in c("X", "-", ".", "*"). If FALSE then return the raw aliphatic index.

Value

A vector of the aliphatic indices for the sequence(s).

References

1. Ikai AJ. Thermostability and aliphatic index of globular proteins. J Biochem. 88, 1895-1898 (1980).

Examples

```
seq <- c("CARDRSTPWRRGIASTTVRTSW", NA, "XXTQMYVRT")
aliphatic(seq)
```

aminoAcidProperties *Calculates amino acid chemical properties for sequence data*

Description

aminoAcidProperties calculates amino acid sequence physicochemical properties, including length, hydrophobicity, bulkiness, polarity, aliphatic index, net charge, acidic residue content, basic residue content, and aromatic residue content.

Usage

```
aminoAcidProperties(data, property = c("length", "gravy", "bulk", "aliphatic",
  "polarity", "charge", "basic", "acidic", "aromatic"), seq = "JUNCTION",
  nt = FALSE, trim = FALSE, label = NULL, ...)
```

Arguments

`data`	`data.frame` containing sequence data.
`property`	vector strings specifying the properties to be calculated. Defaults to calculating all defined properties.
`seq`	`character` name of the column containing input sequences.
`nt`	boolean, TRUE if the sequences (or sequence) are DNA and will be translated.
`trim`	if `TRUE` remove the first and last codon/amino acids from each sequence before calculating properties. If `FALSE` do not modify input sequences.
`label`	name of sequence region to add as prefix to output column names.
`...`	additional named arguments to pass to the functions gravy, bulk, aliphatic, polar or charge.

Details

For all properties except for length, non-informative positions are excluded, where non-informative is defined as any character in `c("X", "-", ".", "*")`.

The scores for GRAVY, bulkiness and polarity are calculated as simple averages of the scores for each informative positions. The basic, acid and aromatic indices are calculated as the fraction of informative positions falling into the given category.

The aliphatic index is calculated using the Ikai, 1980 method.

The net charge is calculated using the method of Moore, 1985, excluding the N-terminus and C-terminus charges, and normalizing by the number of informative positions. The default pH for the calculation is 7.4.

The following data sources were used for the default property scores:

- hydropathy: Kyte & Doolittle, 1982.
- bulkiness: Zimmerman et al, 1968.
- polarity: Grantham, 1974.
- pK: EMBOSS.

Value

A modified `data` data.frame with the following columns:

- `*_AA_LENGTH`: number of amino acids.
- `*_AA_GRAVY`: grand average of hydrophobicity (GRAVY) index.
- `*_AA_BULK`: average bulkiness of amino acids.
- `*_AA_ALIPHATIC`: aliphatic index.

- *_AA_POLARITY: average polarity of amino acids.
- *_AA_CHARGE: net charge.
- *_AA_BASIC: fraction of informative positions that are Arg, His or Lys.
- *_AA_ACIDIC: fraction of informative positions that are Asp or Glu.
- *_AA_AROMATIC: fraction of informative positions that are His, Phe, Trp or Tyr.

Where * is the value from label or the name specified for seq if label=NULL.

References

1. Zimmerman JM, Eliezer N, Simha R. The characterization of amino acid sequences in proteins by statistical methods. J Theor Biol 21, 170-201 (1968).
2. Grantham R. Amino acid difference formula to help explain protein evolution. Science 185, 862-864 (1974).
3. Ikai AJ. Thermostability and aliphatic index of globular proteins. J Biochem 88, 1895-1898 (1980).
4. Kyte J, Doolittle RF. A simple method for displaying the hydropathic character of a protein. J Mol Biol 157, 105-32 (1982).
5. Moore DS. Amino acid and peptide net charges: A simple calculational procedure. Biochem Educ 13, 10-11 (1985).
6. Wu YC, et al. High-throughput immunoglobulin repertoire analysis distinguishes between human IgM memory and switched memory B-cell populations. Blood 116, 1070-8 (2010).
7. Wu YC, et al. The relationship between CD27 negative and positive B cell populations in human peripheral blood. Front Immunol 2, 1-12 (2011).
8. http://emboss.sourceforge.net/apps/cvs/emboss/apps/iep.html

See Also

See countPatterns for counting the occurance of specific amino acid subsequences. See gravy, bulk, aliphatic, polar and charge for functions that calculate the included properties individually.

Examples

```
# Subset example data
db <- ExampleDb[c(1,10,100), c("SEQUENCE_ID", "JUNCTION")]

# Calculate default amino acid properties from amino acid sequences
# Use a custom output column prefix
db$JUNCTION_TRANS <- translateDNA(db$JUNCTION)
aminoAcidProperties(db, seq="JUNCTION_TRANS", label="JUNCTION")
# Calculate default amino acid properties from DNA sequences
aminoAcidProperties(db, seq="JUNCTION", nt=TRUE)

# Use the Grantham, 1974 side chain volume scores from the seqinr package
# Set pH=7.0 for the charge calculation
# Calculate only average volume and charge
# Remove the head and tail amino acids from the junction, thus making it the CDR3
```

```
library(seqinr)
data(aaindex)
x <- aaindex[["GRAR740103"]]$I
# Rename the score vector to use single-letter codes
names(x) <- translateStrings(names(x), ABBREV_AA)
# Calculate properties
aminoAcidProperties(db, property=c("bulk", "charge"), seq="JUNCTION", nt=TRUE,
                    trim=TRUE, label="CDR3", bulkiness=x, pH=7.0)
```

`buildPhylipLineage` *Infer an Ig lineage using PHYLIP*

Description

`buildPhylipLineage` reconstructs an Ig lineage via maximum parsimony using the dnapars application of the PHYLIP package.

Usage

```
buildPhylipLineage(clone, dnapars_exec, rm_temp = FALSE, verbose = FALSE)
```

Arguments

`clone`	ChangeoClone object containing clone data.
`dnapars_exec`	path to the PHYLIP dnapars executable.
`rm_temp`	if `TRUE` delete the temporary directory after running dnapars; if `FALSE` keep the temporary directory.
`verbose`	if `FALSE` suppress the output of dnapars; if `TRUE` STDOUT and STDERR of dnapars will be passed to the console.

Details

`buildPhylipLineage` builds the lineage tree of a set of unique Ig sequences via maximum parsimony through an external call to the dnapars application of the PHYLIP package. dnapars is called with default algorithm options, except for the search option, which is set to "Rearrange on one best tree". The germline sequence of the clone is used for the outgroup.

Following tree construction using dnapars, the dnapars output is modified to allow input sequences to appear as internal nodes of the tree. Intermediate sequences inferred by dnapars are replaced by children within the tree having a Hamming distance of zero from their parent node. The distance calculation allows IUPAC ambiguous character matches, where an ambiguous character has distance zero to any character in the set of characters it represents. Distance calculation and movement of child nodes up the tree is repeated until all parent-child pairs have a distance greater than zero between them. The germline sequence (outgroup) is moved to the root of the tree and excluded from the node replacement processes, which permits the trunk of the tree to be the only edge with a distance of zero. Edge weights of the resultant tree are assigned as the distance between each sequence.

Value

An igraph graph object defining the Ig lineage tree. Each unique input sequence in clone is a vertex of the tree, with additional vertices being either the germline (root) sequences or inferred intermediates. The graph object has the following attributes.

Vertex attributes:

- name: value in the SEQUENCE_ID column of the data slot of the input clone for observed sequences. The germline (root) vertex is assigned the name "Germline" and inferred intermediates are assigned names with the format "Inferred1", "Inferred2", ...
- sequence: value in the SEQUENCE column of the data slot of the input clone for observed sequences. The germline (root) vertex is assigned the sequence in the germline slot of the input clone. The sequence of inferred intermediates are extracted from the dnapars output.
- label: same as the name attribute.

Additionally, each other column in the data slot of the input clone is added as a vertex attribute with the attribute name set to the source column name. For the germline and inferred intermediate vertices, these additional vertex attributes are all assigned a value of NA.

Edge attributes:

- weight: Hamming distance between the sequence attributes of the two vertices.
- label: same as the weight attribute.

Graph attributes:

- clone: clone identifier from the clone slot of the input ChangeoClone.
- v_gene: V-segment gene call from the v_gene slot of the input ChangeoClone.
- j_gene: J-segment gene call from the j_gene slot of the input ChangeoClone.
- junc_len: junction length (nucleotide count) from the junc_len slot of the input ChangeoClone.

References

1. Felsenstein J. PHYLIP - Phylogeny Inference Package (Version 3.2). Cladistics. 1989 5:164-166.
2. Stern JNH, Yaari G, Vander Heiden JA, et al. B cells populating the multiple sclerosis brain mature in the draining cervical lymph nodes. Sci Transl Med. 2014 6(248):248ra107.

See Also

Takes as input a ChangeoClone. Temporary directories are created with makeTempDir. Distance is calculated using seqDist. See igraph and igraph.plotting for working with igraph graph objects.

Examples

```
## Not run:
# Preprocess clone
clone <- subset(ExampleDb, CLONE == 3138)
clone <- makeChangeoClone(clone, text_fields=c("SAMPLE", "ISOTYPE"), num_fields="DUPCOUNT")
```

```
# Run PHYLIP and process output
dnapars_exec <- "~/apps/phylip-3.69/dnapars"
graph <- buildPhylipLineage(clone, dnapars_exec, rm_temp=TRUE)

# Plot graph with a tree layout
library(igraph)
plot(graph, layout=layout_as_tree, vertex.label=V(graph)$ISOTYPE,
     vertex.size=50, edge.arrow.mode=0, vertex.color="grey80")

## End(Not run)
```

---

bulk *Calculates the average bulkiness of amino acid sequences*

Description

bulk calculates the average bulkiness score of amino acid sequences. Non-informative positions are excluded, where non-informative is defined as any character in c("X", "-", ".", "*").

Usage

```
bulk(seq, bulkiness = NULL)
```

Arguments

seq	vector of strings containing amino acid sequences.
bulkiness	named numerical vector defining bulkiness scores for each amino acid, where names are single-letter amino acid character codes. If NULL, then the Zimmerman et al, 1968 scale is used.

Value

A vector of bulkiness scores for the sequence(s).

References

1. Zimmerman JM, Eliezer N, Simha R. The characterization of amino acid sequences in proteins by statistical methods. J Theor Biol 21, 170-201 (1968).

See Also

For additional size related indices see aaindex.

Examples

```
# Default bulkiness scale
seq <- c("CARDRSTPWRRGIASTTVRTSW", "XXTQMYVRT")
bulk(seq)

# Use the Grantham, 1974 side chain volumn scores from the seqinr package
library(seqinr)
data(aaindex)
x <- aaindex[["GRAR740103"]]$I
# Rename the score vector to use single-letter codes
names(x) <- translateStrings(names(x), ABBREV_AA)
# Calculate average volume
bulk(seq, bulkiness=x)
```

---

calcCoverage *Calculate sample coverage*

Description

calcCoverage calculates the sample coverage estimate, a measure of sample completeness, for varying orders using the method of Chao et al, 2015, falling back to the Chao1 method in the first order case.

Usage

```
calcCoverage(x, r = 1)
```

Arguments

x	numeric vector of abundance counts.
r	coverage order to calculate.

Value

The sample coverage of the given order r.

References

1. Chao A. Nonparametric Estimation of the Number of Classes in a Population. Scand J Stat. 1984 11, 265270.
2. Chao A, et al. Unveiling the species-rank abundance distribution by generalizing the Good-Turing sample coverage theory. Ecology. 2015 96, 11891201.

See Also

Used by rarefyDiversity.

Examples

```
# Calculate clone sizes
clones <- countClones(ExampleDb, groups="SAMPLE")

# Calculate 1first order coverage for a single sample
calcCoverage(clones$SEQ_COUNT[clones$SAMPLE == "+7d"])
```

`calcDiversity` *Calculate the diversity index*

Description

`calcDiversity` calculates the clonal diversity index for a vector of diversity orders.

Usage

```
calcDiversity(p, q)
```

Arguments

`p`	numeric vector of clone (species) counts or proportions.
`q`	numeric vector of diversity orders.

Details

This method, proposed by Hill (Hill, 1973), quantifies diversity as a smooth function (D) of a single parameter q. Special cases of the generalized diversity index correspond to the most popular diversity measures in ecology: species richness ($q = 0$), the exponential of the Shannon-Weiner index (q approaches 1), the inverse of the Simpson index ($q = 2$), and the reciprocal abundance of the largest clone (q approaches $+\infty$). At $q = 0$ different clones weight equally, regardless of their size. As the parameter q increase from 0 to $+\infty$ the diversity index (D) depends less on rare clones and more on common (abundant) ones, thus encompassing a range of definitions that can be visualized as a single curve.

Values of $q < 0$ are valid, but are generally not meaningful. The value of D at $q = 1$ is estimated by D at $q = 0.9999$.

Value

A vector of diversity scores D for each q.

References

1. Hill M. Diversity and evenness: a unifying notation and its consequences. Ecology. 1973 54(2):427-32.

See Also

Used by rarefyDiversity and testDiversity.

Examples

```
# May define p as clonal member counts
p <- c(1, 1, 3, 10)
q <- c(0, 1, 2)
calcDiversity(p, q)

# Or proportional abundance
p <- c(1/15, 1/15, 1/5, 2/3)
calcDiversity(p, q)
```

ChangeoClone-class *S4 class defining a clone*

Description

ChangeoClone defines a common data structure for perform lineage recontruction from Change-O data.

Slots

`data` data.frame containing sequences and annotations. Contains the columns `SEQUENCE_ID` and `SEQUENCE`, as well as any additional sequence-specific annotation columns.

`clone` string defining the clone identifier.

`germline` string containing the germline sequence for the clone.

`v_gene` string defining the V segment gene call.

`j_gene` string defining the J segment gene call.

`junc_len` numeric junction length (nucleotide count).

See Also

See makeChangeoClone and buildPhylipLineage for use.

charge *Calculates the net charge of amino acid sequences.*

Description

charge calculates the net charge of amino acid sequences using the method of Moore, 1985, with exclusion of the C-terminus and N-terminus charges.

Usage

```
charge(seq, pH = 7.4, pK = NULL, normalize = FALSE)
```

Arguments

seq	vector strings defining of amino acid sequences.
pH	environmental pH.
pK	named vector defining pK values for each charged amino acid, where names are the single-letter amino acid character codes c("R", "H", "K", "D", "E", "C", "Y")). If NULL, then the EMBOSS scale is used.
normalize	if TRUE then divide the net charge of each amino acid sequence by the number of informative positions. Non-informative position are defined by the presence any character in c("X", "-", ".", "*"). If FALSE then return the raw net charge.

Value

A vector of net charges for the sequence(s).

References

1. Moore DS. Amino acid and peptide net charges: A simple calculational procedure. Biochem Educ. 13, 10-11 (1985).
2. http://emboss.sourceforge.net/apps/cvs/emboss/apps/iep.html

See Also

For additional pK scales see pK.

Examples

```
seq <- c("CARDRSTPWRRGIASTTVRTSW", "XXTQMYVRT")
# Unnormalized charge
charge(seq)
# Normalized charge
charge(seq, normalize=TRUE)

# Use the Murray et al, 2006 scores from the seqinr package
```

```
library(seqinr)
data(pK)
x <- setNames(pK[["Murray"]], rownames(pK))
# Calculate charge
charge(seq, pK=x)
```

---

collapseDuplicates *Remove duplicate DNA sequences and combine annotations*

Description

collapseDuplicates identifies duplicate DNA sequences, allowing for ambiguous characters, removes the duplicate entries, and combines any associated annotations.

Usage

```
collapseDuplicates(data, id = "SEQUENCE_ID", seq = "SEQUENCE_IMGT",
  text_fields = NULL, num_fields = NULL, seq_fields = NULL,
  add_count = FALSE, ignore = c("N", "-", ".", "?"), sep = ",",
  verbose = FALSE)
```

Arguments

data	data.frame containing Change-O columns. The data.frame must contain, at a minimum, a unique identifier column and a column containg a character vector of DNA sequences.
id	name of the column containing sequence identifiers.
seq	name of the column containing DNA sequences.
text_fields	character vector of textual columns to collapse. The textual annotations of duplicate sequences will be merged into a single string with each unique value alphabetized and delimited by sep.
num_fields	vector of numeric columns to collapse. The numeric annotations of duplicate sequences will be summed.
seq_fields	vector of nucletoide sequence columns to collapse. The sequence with the fewest numer of non-informative characters will be retained. Where a non-informative character is one of c("N", "-", ".", "?"). Note, this is distinct from the seq parameter which is used to determine duplicates.
add_count	if TRUE add the column COLLAPSE_COUNT that indicates the number of sequences that were collapsed to build each unique entry.
ignore	vector of characters to ignore when testing for equality.
sep	character to use for delimiting collapsed annotations in the text_fields columns. Defines both the input and output delimiter.
verbose	if TRUE report the number input, discarded and output sequences; if FALSE process sequences silently.

Details

collapseDuplicates identifies duplicate sequences in the seq column by testing for character identity, with consideration of IUPAC ambiguous nucleotide codes. A cluster of sequences are considered duplicates if they are all equivalent, and no member of the cluster is equivalent to a sequence in a different cluster.

Textual annotations, specified by text_fields, are collapsed by taking the unique set of values within in each duplicate cluster and delimiting those values by sep. Numeric annotations, specified by num_fields, are collapsed by summing all values in the duplicate cluster. Sequence annotations, specified by seq_fields, are collapsed by retaining the first sequence with the fewest number of N characters.

Columns that are not specified in either text_fields, num_fields, or seq_fields will be retained, but the value will be chosen from a random entry amongst all sequences in a cluster of duplicates.

An ambiguous sequence is one that can be assigned to two different clusters, wherein the ambiguous sequence is equivalent to two sequences which are themselves non-equivalent. Ambiguous sequences arise due to ambiguous characters at positions that vary across sequences, and are discarded along with their annotations. Thus, ambiguous sequences are removed as duplicates of some sequence, but do not create a potential false-positive annotation merger. Ambiguous sequences are not included in the COLLAPSE_COUNT annotation that is added when add_count=TRUE.

Value

A modified data data.frame with duplicate sequences removed and annotation fields collapsed.

See Also

Equality is tested with seqEqual and pairwiseEqual. For IUPAC ambiguous character codes see IUPAC_DNA.

Examples

```
# Example Change-O data.frame
db <- data.frame(SEQUENCE_ID=LETTERS[1:4],
                 SEQUENCE_IMGT=c("CCCCTGGG", "CCCCTGGN", "NAACTGGN", "NNNCTGNN"),
                 TYPE=c("IgM", "IgG", "IgG", "IgA"),
                 SAMPLE=c("S1", "S1", "S2", "S2"),
                 COUNT=1:4,
                 stringsAsFactors=FALSE)

# Annotations are not parsed if neither text_fields nor num_fields is specified
# The retained sequence annotations will be random
collapseDuplicates(db, verbose=TRUE)

# Unique text_fields annotations are combined into a single string with ","
# num_fields annotations are summed
# Ambiguous duplicates are discarded
collapseDuplicates(db, text_fields=c("TYPE", "SAMPLE"), num_fields="COUNT",
                   verbose=TRUE)

# Use alternate delimiter for collapsing textual annotations
```

```
collapseDuplicates(db, text_fields=c("TYPE", "SAMPLE"), num_fields="COUNT",
                   sep="/", verbose=TRUE)

# Add count of duplicates
collapseDuplicates(db, text_fields=c("TYPE", "SAMPLE"), num_fields="COUNT",
                   add_count=TRUE, verbose=TRUE)

# Masking ragged ends may impact duplicate removal
db$SEQUENCE_IMGT <- maskSeqEnds(db$SEQUENCE_IMGT)
collapseDuplicates(db, text_fields=c("TYPE", "SAMPLE"), num_fields="COUNT",
                   add_count=TRUE, verbose=TRUE)
```

`countClones` *Tabulates clones sizes*

Description

`countClones` determines the number of sequences and total copy number of clonal groups.

Usage

```
countClones(data, groups = NULL, copy = NULL, clone = "CLONE")
```

Arguments

`data`	data.frame with Change-O style columns containing clonal assignments.
`groups`	character vector defining `data` columns containing grouping variables. If `group=NULL`, then do not group data.
`copy`	name of the `data` column containing copy numbers for each sequence. If this value is specified, then total copy abundance is determined by the sum of copy numbers within each clonal group.
`clone`	name of the `data` column containing clone identifiers.

Value

A data.frame summarizing clone counts and frequencies with columns:

- `CLONE`: clone identifier.
- `SEQ_COUNT`: total number of sequences for the clone.
- `SEQ_FREQ`: frequency of the clone as a fraction of the total number of sequences within each group.
- `COPY_COUNT`: sum of the copy counts in the `copy` column. Only present if the `copy` argument is specified.
- `COPY_FREQ`: frequency of the clone as a fraction of the total copy number within each group. Only present if the `copy` argument is specified.

Also includes additional columns specified in the `groups` argument.

Examples

```
# Without copy numbers
clones <- countClones(ExampleDb, groups="SAMPLE")

# With copy numbers and multiple groups
clones <- countClones(ExampleDb, groups=c("SAMPLE", "ISOTYPE"), copy="DUPCOUNT")
```

---

countGenes *Tabulates V(D)J allele, gene or family usage.*

Description

Determines the count and relative abundance of V(D)J alleles, genes or families within groups.

Usage

```
countGenes(data, gene, groups = NULL, copy = NULL, clone = NULL,
  mode = c("gene", "allele", "family"))
```

Arguments

data	data.frame with Change-O style columns.
gene	column containing allele assignments. Only the first allele in the column will be considered.
groups	columns containing grouping variables. If NULL do not group.
copy	name of the data column containing copy numbers for each sequence. If this value is specified, then total copy abundance is determined by the sum of copy numbers within each gene. This argument is ignored if clone is specified.
clone	name of the data column containing clone identifiers for each sequence. If this value is specified, then genes will be counted only once for each clone. Note, this is accomplished by using the most common gene within each clone identifier. As such, ambiguous alleles within a clone will not be accurately represented.
mode	one of c("gene", "family", "allele") defining the degree of specificity regarding allele calls. Determines whether to return counts for genes, families or alleles.

Value

A data.frame summarizing family, gene or allele counts and frequencies with columns:

- GENE: name of the family, gene or allele
- SEQ_COUNT: total number of sequences, or clones, for the gene.
- SEQ_FREQ: frequency of the gene as a fraction of the total number of sequences, or clones, within each grouping.

- COPY_COUNT: sum of the copy counts in the copy column. for each gene. Only present if the copy argument is specified.
- COPY_FREQ: frequency of the gene as a fraction of the total copy number within each group. Only present if the copy argument is specified.

Additional columns defined by the groups argument will also be present.

Examples

```
# Without copy numbers
genes <- countGenes(ExampleDb, gene="V_CALL", groups="SAMPLE", mode="family")
genes <- countGenes(ExampleDb, gene="V_CALL", groups="SAMPLE", mode="gene")
genes <- countGenes(ExampleDb, gene="V_CALL", groups="SAMPLE", mode="allele")

# With copy numbers and multiple groups
genes <- countGenes(ExampleDb, gene="V_CALL", groups=c("SAMPLE", "ISOTYPE"),
                    copy="DUPCOUNT", mode="family")

# Count by clone
genes <- countGenes(ExampleDb, gene="V_CALL", groups=c("SAMPLE", "ISOTYPE"),
                    clone="CLONE", mode="family")
```

---

countPatterns *Count sequence patterns*

Description

countPatterns counts the fraction of times a set of character patterns occur in a set of sequences.

Usage

```
countPatterns(seq, patterns, nt = FALSE, trim = FALSE, label = "REGION")
```

Arguments

seq	character vector of either DNA or amino acid sequences.
patterns	list of sequence patterns to count in each sequence. If the list is named, then names will be assigned as the column names of output data.frame.
nt	if TRUE then seq are DNA sequences and and will be translated before performing the pattern search.
trim	if TRUE remove the first and last codon or amino acid from each sequence before the pattern search. If FALSE do not modify the input sequences.
label	string defining a label to add as a prefix to the output column names.

Value

A data.frame containing the fraction of times each sequence pattern was found.

Examples

```
seq <- c("TGTCAACAGGCTAACAGTTTCCGGACGTTC",
         "TGTCAGCAATATTATATTGCTCCCTTCACTTTC",
         "TGTCAAAAGTATAACAGTGCCCCCTGGACGTTC")
patterns <- c("A", "V", "[LI]")
names(patterns) <- c("ARG", "VAL", "ISO_LEU")
countPatterns(seq, patterns, nt=TRUE, trim=TRUE, label="CDR3")
```

DEFAULT_COLORS *Default colors*

Description

Default color palettes for DNA characters, Ig isotypes, and TCR chains.

Usage

```
DNA_COLORS

IG_COLORS

TR_COLORS
```

Format

Named character vectors with hexcode colors as values.

- `DNA_COLORS`: DNA character colors `c("A", "C", "G", "T")`.
- `IG_COLORS`: Ig isotype colors `c("IgA", "IgD", "IgE", "IgG", "IgM", "IgK", "IgL")`.
- `TR_COLORS`: TCR chain colors `c("TRA", "TRB", "TRD", "TRG")`.

Examples

```
# IG_COLORS as an isotype color set for ggplot
isotype <- c("IgG", "IgM", "IgM", "IgA")
db <- data.frame(x=1:4, y=1:4, iso=isotype)
g1 <- ggplot(db, aes(x=x, y=y, color=iso)) +
    scale_color_manual(name="Isotype", values=IG_COLORS) +
    geom_point(size=10)
plot(g1)

# DNA_COLORS to translate nucleotide values to a vector of colors
# for use in base graphics plots
seq <- c("A", "T", "T", "C")
colors <- translateStrings(seq, setNames(names(DNA_COLORS), DNA_COLORS))
plot(1:4, 1:4, col=colors, pch=16, cex=6)
```

`DiversityCurve-class` *S4 class defining diversity curve*

Description

`DiversityCurve` defines diversity (D) scores over multiple diversity orders (Q).

Usage

```
## S4 method for signature 'DiversityCurve'
print(x)

## S4 method for signature 'DiversityCurve,missing'
plot(x, y, ...)
```

Arguments

`x`	DiversityCurve object
`y`	ignored.
`...`	arguments to pass to plotDiversityCurve.

Slots

`data` data.frame defining the diversity curve with the following columns:

- `GROUP`: group label.
- `Q`: diversity order.
- `D`: mean diversity index over all bootstrap realizations.
- `D_SD`: standard deviation of the diversity index over all bootstrap realizations.
- `D_LOWER`: diversity lower confidence inverval bound.
- `D_UPPER`: diversity upper confidence interval bound.
- `E`: evenness index calculated as `D` divided by `D` at `Q=0`.
- `E_LOWER`: evenness lower confidence inverval bound.
- `E_UPPER`: eveness upper confidence interval bound.

`groups` character vector of groups retained in the diversity calculation.

`n` numeric vector indication the number of sequences sampled from each group.

`nboot` number of bootstrap realizations performed.

`ci` confidence interval defining the upper and lower bounds (a value between 0 and 1).

`DiversityTest-class` *S4 class defining diversity significance*

Description

`DiversityTest` defines the signifance of diversity (D) differences at a fixed diversity order (q).

Usage

```
## S4 method for signature 'DiversityTest'
print(x)
```

Arguments

`x`	DiversityTest object.

Slots

`tests` data.frame describing the significance test results with columns:

- `TEST`: string listing the two groups tested.
- `DELTA_MEAN`: mean of the D bootstrap delta distribution for the test.
- `DELTA_SD`: standard deviation of the D bootstrap delta distribution for the test.
- `PVALUE`: p-value for the test.

`summary` data.frame containing summary statistics for the diversity index bootstrap distributions, at the given value of q, with columns:

- `GROUP`: the name of the group.
- `MEAN`: mean of the D bootstrap distribution.
- `SD`: standard deviation of the D bootstrap distribution.

`groups` character vector of groups retained in diversity calculation.

`q` diversity order tested (q).

`n` numeric vector indication the number of sequences sampled from each group.

`nboot` number of bootstrap realizations.

EdgeTest-class *S4 class defining edge significance*

Description

EdgeTest defines the significance of parent-child annotation enrichment.

Usage

```
## S4 method for signature 'EdgeTest'
print(x)

## S4 method for signature 'EdgeTest,missing'
plot(x, y, ...)
```

Arguments

x	EdgeTest object.
y	ignored.
...	arguments to pass to plotEdgeTest.

Slots

tests data.frame describing the significance test results with columns:

- PARENT: parent node annotation.
- CHILD: child node annotation
- COUNT: count of observed edges with the given parent-child annotation set.
- EXPECTED: mean count of expected edges for the given parent-child relationship.
- PVALUE: one-sided p-value for the hypothesis that the observed edge abundance is greater than expected.

permutations data.frame containing the raw permutation test data with columns:

- PARENT: parent node annotation.
- CHILD: child node annotation
- COUNT: count of edges with the given parent-child annotation set.
- ITER: numerical index define which permutation realization each observation corresponds to.

nperm number of permutation realizations.

`estimateAbundance` *Estimates the complete clonal relative abundance distribution*

Description

`estimateAbundance` estimates the complete clonal relative abundance distribution and confidence intervals on clone sizes using bootstrapping.

Usage

```
estimateAbundance(data, group, clone = "CLONE", copy = NULL, ci = 0.95,
  nboot = 2000, progress = FALSE)
```

Arguments

`data`	data.frame with Change-O style columns containing clonal assignments.
`group`	name of the `data` column containing group identifiers.
`clone`	name of the `data` column containing clone identifiers.
`copy`	name of the `data` column containing copy numbers for each sequence. If `copy=NULL` (the default), then clone abundance is determined by the number of sequences. If a `copy` column is specified, then clone abundances is determined by the sum of copy numbers within each clonal group.
`ci`	confidence interval to calculate; the value must be between 0 and 1.
`nboot`	number of bootstrap realizations to generate.
`progress`	if `TRUE` show a progress bar.

Details

The complete clonal abundance distribution determined inferred by using the Chao1 estimator to estimate the number of seen clones, and then applying the relative abundance correction and unseen clone frequencies described in Chao et al, 2015.

Confidence intervals are derived using the standard deviation of the resampling realizations, as described in Chao et al, 2015.

Value

A data.frame with relative clonal abundance data and confidence intervals, containing the following columns:

- `GROUP`: group identifier.
- `CLONE`: clone identifier.
- `P`: relative abundance of the clone.
- `LOWER`: lower confidence inverval bound.
- `UPPER`: upper confidence interval bound.
- `RANK`: the rank of the clone abundance.

References

1. Chao A. Nonparametric Estimation of the Number of Classes in a Population. Scand J Stat. 1984 11, 265270.
2. Chao A, et al. Rarefaction and extrapolation with Hill numbers: A framework for sampling and estimation in species diversity studies. Ecol Monogr. 2014 84:45-67.
3. Chao A, et al. Unveiling the species-rank abundance distribution by generalizing the Good-Turing sample coverage theory. Ecology. 2015 96, 11891201.

See Also

See plotAbundance for plotting of the abundance distribution. See rarefyDiversity for a similar application to clonal diversity.

Examples

```
abund <- estimateAbundance(ExampleDb, "SAMPLE", nboot=100)
```

ExampleDb *Example Change-O database*

Description

A small example database subset from Laserson and Vigneault et al, 2014.

Usage

```
ExampleDb
```

Format

A data.frame with the following Change-O style columns:

- SEQUENCE_ID: Sequence identifier
- SEQUENCE_IMGT: IMGT-gapped observed sequence.
- GERMLINE_IMGT_D_MASK: IMGT-gapped germline sequence with N, P and D regions masked.
- V_CALL: V region allele assignments.
- V_CALL_GENOTYPED: TIgGER corrected V region allele assignment.
- D_CALL: D region allele assignments.
- J_CALL: J region allele assignments.
- JUNCTION: Junction region sequence.
- JUNCTION_LENGTH: Length of the junction region in nucleotides.
- NP1_LENGTH: Combined length of the N and P regions proximal to the V region.
- NP2_LENGTH: Combined length of the N and P regions proximal to the J region.

- SAMPLE: Sample identifier. Time in relation to vaccination.
- ISOTYPE: Isotype assignment.
- DUPCOUNT: Copy count (number of duplicates) of the sequence.
- CLONE: Change-O assignment clonal group identifier.

References

1. Laserson U and Vigneault F, et al. High-resolution antibody dynamics of vaccine-induced immune responses. Proc Natl Acad Sci USA. 2014 111:4928-33.

See Also

ExampleTrees

ExampleTrees *Example Ig lineage trees*

Description

A set of Ig lineage trees generated from the ExampleDb file, subset to only those trees with at least four nodes.

Usage

```
ExampleTrees
```

Format

A list of igraph objects output by buildPhylipLineage. Each node of each tree has the following annotations (vertex attributes):

- SAMPLE: Sample identifier(s). Time in relation to vaccination.
- ISOTYPE: Isotype assignment(s).
- DUPCOUNT: Copy count (number of duplicates) of the sequence.

See Also

ExampleTrees

`extractVRegion` *Extracts FWRs and CDRs from IMGT-gapped sequences*

Description

`extractVRegion` extracts the framework and complementarity determining regions of the V-segment for IMGT-gapped immunoglobulin (Ig) nucleotide sequences according to the IMGT numbering scheme.

Usage

```
extractVRegion(sequences, region = c("FWR1", "CDR1", "FWR2", "CDR2", "FWR3"))
```

Arguments

`sequences`	character vector of IMGT-gapped nucleotide sequences.
`region`	string defining the region(s) of the V-segment to extract. May be a single region or multiple regions (as a vector) from `c("FWR1", "CDR1", "FWR2", "CDR2" ,"FWR3")`. By default, all regions will be returned.

Value

If only one region is specified in the `region` argument, a character vector of the extracted subsequences will be returned. If multiple regions are specified, then a character matrix will be returned with columns corresponding to the specified regions and a row for each entry in `sequences`.

References

1. Lefranc M-P, et al. IMGT unique numbering for immunoglobulin and T cell receptor variable domains and Ig superfamily V-like domains. Dev Comp Immunol. 2003 27(1):55-77.

See Also

IMGT-gapped region boundaries are defined in IMGT_REGIONS.

Examples

```
# Assign example clone
clone <- subset(ExampleDb, CLONE == 3138)

# Get all regions
extractVRegion(clone$SEQUENCE_IMGT)

# Get single region
extractVRegion(clone$SEQUENCE_IMGT, "FWR1")

# Get all CDRs
extractVRegion(clone$SEQUENCE_IMGT, c("CDR1", "CDR2"))
```

```
# Get all FWRs
extractVRegion(clone$SEQUENCE_IMGT, c("FWR1", "FWR2", "FWR3"))
```

`getAAMatrix` *Build an AA distance matrix*

Description

`getAAMatrix` returns a Hamming distance matrix for IUPAC ambiguous amino acid characters.

Usage

```
getAAMatrix(gap = 0)
```

Arguments

`gap`	value to assign to characters in the set `c("-", ".")`.

Value

A `matrix` of amino acid character distances with row and column names indicating the character pair.

See Also

Creates an amino acid distance matrix for seqDist. See getDNAMatrix for nucleotide distances.

Examples

```
getAAMatrix()
```

`getDNAMatrix` *Build a DNA distance matrix*

Description

`getDNAMatrix` returns a Hamming distance matrix for IUPAC ambiguous DNA characters with modifications for gap, `c("-", ".")`, and missing, `c("?")`, character values.

Usage

```
getDNAMatrix(gap = -1)
```

Arguments

gap	value to assign to characters in the set c("-", ".").

Value

A matrix of DNA character distances with row and column names indicating the character pair. By default, distances will be either 0 (equivalent), 1 (non-equivalent or missing), or -1 (gap).

See Also

Creates DNA distance matrix for seqDist. See getAAMatrix for amino acid distances.

Examples

```
# Set gap characters to Inf distance
# Distinguishes gaps from Ns
getDNAMatrix()

# Set gap characters to 0 distance
# Makes gap characters equivalent to Ns
getDNAMatrix(gap=0)
```

---

getMRCA *Retrieve the first non-root node of a lineage tree*

Description

getMRCA returns the set of lineage tree nodes with the minimum weighted or unweighted path length from the root (germline) of the lineage tree, allowing for exclusion of specific groups of nodes.

Usage

```
getMRCA(graph, path = c("distance", "steps"), root = "Germline",
  field = NULL, exclude = NULL)
```

Arguments

graph	igraph object containing an annotated lineage tree.
path	string defining whether to use unweighted (steps) or weighted (distance) measures for determining the founder node set..
root	name of the root (germline) node.
field	annotation field to use for both unweighted path length exclusion and consideration as an MRCA node. If NULL do not exclude any nodes.
exclude	vector of annotation values in field to exclude from the potential MRCA set. If NULL do not exclude any nodes. Has no effect if field=NULL.

Value

A data.frame of the MRCA node(s) containing the columns:

- NAME: node name
- STEPS: path length as the number of nodes traversed
- DISTANCE: path length as the sum of edge weights

Along with additional columns corresponding to the annotations of the input graph.

See Also

Path lengths are determined with getPathLengths.

Examples

```
# Define example graph
graph <- ExampleTrees[[23]]

# Use unweighted path length and do not exclude any nodes
getMRCA(graph, path="steps", root="Germline")

# Exclude nodes without an isotype annotation and use weighted path length
getMRCA(graph, path="distance", root="Germline", field="ISOTYPE", exclude=NA)
```

---

getPathLengths *Calculate path lengths from the tree root*

Description

getPathLengths calculates the unweighted (number of steps) and weighted (distance) path lengths from the root of a lineage tree.

Usage

```
getPathLengths(graph, root = "Germline", field = NULL, exclude = NULL)
```

Arguments

graph	igraph object containing an annotated lineage tree.
root	name of the root (germline) node.
field	annotation field to use for exclusion of nodes from step count.
exclude	annotation values specifying which nodes to exclude from step count. If NULL consider all nodes. This does not affect the weighted (distance) path length calculation.

Value

A data.frame with columns:

- NAME: node name
- STEPS: path length as the number of nodes traversed
- DISTANCE: path length as the sum of edge weights

See Also

See buildPhylipLineage for generating input trees.

Examples

```
# Define example graph
graph <- ExampleTrees[[24]]

# Consider all nodes
getPathLengths(graph, root="Germline")

# Exclude nodes without an isotype annotation from step count
getPathLengths(graph, root="Germline", field="ISOTYPE", exclude=NA)
```

---

getSegment *Get Ig segment allele, gene and family names*

Description

getSegment performs generic matching of delimited segment calls with a custom regular expression. getAllele, getGene and getFamily extract the allele, gene and family names, respectively, from a character vector of immunoglobulin (Ig) or TCR segment allele calls in IMGT format.

Usage

```
getSegment(segment_call, segment_regex, first = TRUE, collapse = TRUE,
  strip_d = TRUE, sep = ",")

getAllele(segment_call, first = TRUE, collapse = TRUE, strip_d = TRUE,
  sep = ",")

getGene(segment_call, first = TRUE, collapse = TRUE, strip_d = TRUE,
  sep = ",")

getFamily(segment_call, first = TRUE, collapse = TRUE, strip_d = TRUE,
  sep = ",")
```

Arguments

segment_call	character vector containing segment calls delimited by commas.
segment_regex	string defining the segment match regular expression.
first	if TRUE return only the first call in segment_call; if FALSE return all calls delimited by commas.
collapse	if TRUE check for duplicates and return only unique segment assignments; if FALSE return all assignments (faster). Has no effect if first=TRUE.
strip_d	if TRUE remove the "D" from the end of gene annotations (denoting a duplicate gene in the locus); if FALSE do not alter gene names.
sep	character defining both the input and output segment call delimiter.

Value

A character vector containing allele, gene or family names.

References

http://imgt.org

See Also

countGenes

Examples

```
kappa_call <- c("Homsap IGKV1D-39*01 F,Homsap IGKV1-39*02 F,Homsap IGKV1-39*01",
                "Homsap IGKJ5*01 F")

getAllele(kappa_call)
getAllele(kappa_call, first=FALSE)
getAllele(kappa_call, first=FALSE, strip_d=FALSE)

getGene(kappa_call)
getGene(kappa_call, first=FALSE)
getGene(kappa_call, first=FALSE, strip_d=FALSE)

getFamily(kappa_call)
getFamily(kappa_call, first=FALSE)
getFamily(kappa_call, first=FALSE, collapse=FALSE)
getFamily(kappa_call, first=FALSE, strip_d=FALSE)

heavy_call <- c("Homsap IGHV1-69*01 F,Homsap IGHV1-69D*01 F",
                "Homsap IGHD1-1*01 F",
                "Homsap IGHJ1*01 F")

getAllele(heavy_call, first=FALSE)
getAllele(heavy_call, first=FALSE, strip_d=FALSE)

getGene(heavy_call, first=FALSE)
```

```
getGene(heavy_call, first=FALSE, strip_d=FALSE)
```

---

gravy *Calculates the hydrophobicity of amino acid sequences*

Description

gravy calculates the Grand Average of Hydrophobicity (GRAVY) index of amino acid sequences using the method of Kyte & Doolittle. Non-informative positions are excluded, where non-informative is defined as any character in `c("X", "-", ".", "*")`.

Usage

```
gravy(seq, hydropathy = NULL)
```

Arguments

seq	vector of strings containing amino acid sequences.
hydropathy	named numerical vector defining hydropathy index values for each amino acid, where names are single-letter amino acid character codes. If NULL, then the Kyte & Doolittle scale is used.

Value

A vector of GRAVY scores for the sequence(s).

References

1. Kyte J, Doolittle RF. A simple method for displaying the hydropathic character of a protein. J Mol Biol. 157, 105-32 (1982).

See Also

For additional hydrophobicity indices see aaindex.

Examples

```
# Default scale
seq <- c("CARDRSTPWRRGIASTTVRTSW", "XXTQMYVRT")
gravy(seq)

# Use the Kidera et al, 1985 scores from the seqinr package
library(seqinr)
data(aaindex)
x <- aaindex[["KIDA850101"]]$I
# Rename the score vector to use single-letter codes
names(x) <- translateStrings(names(x), ABBREV_AA)
# Calculate hydrophobicity
```

```
gravy(seq, hydropathy=x)
```

`gridPlot` *Plot multiple ggplot objects*

Description

Plots multiple ggplot objects in an equally sized grid.

Usage

```
gridPlot(..., ncol = 1)
```

Arguments

`...`	ggplot objects to plot.
`ncol`	number of columns in the plot.

References

Modified from: http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)

See Also

ggplot.

`IMGT_REGIONS` *IMGT V-segment regions*

Description

A list defining the boundaries of V-segment framework regions (FWRs) and complementarity determining regions (CDRs) for IMGT-gapped immunoglobulin (Ig) nucleotide sequences according to the IMGT numbering scheme.

Usage

```
IMGT_REGIONS
```

Format

A list with regions named one of `c("FWR1", "CDR1", "FWR2", "CDR2", "FWR3")` with values containing a numeric vector of length two defining the `c(start, end)` positions of the named region.

References

http://imgt.org

isValidAASeq *Validate amino acid sequences*

Description

isValidAASeq checks that a set of sequences are valid non-ambiguous amino acid sequences. A sequence is considered valid if it contains only characters in the the non-ambiguous IUPAC character set or any characters in c("X", ".", "-", "*").

Usage

```
isValidAASeq(seq)
```

Arguments

seq	character vector of sequences to check.

Value

A logical vector with TRUE for each valid amino acid sequences and FALSE for each invalid sequence.

See Also

See ABBREV_AA for the set of non-ambiguous amino acid characters. See IUPAC_AA for the full set of ambiguous amino acid characters.

Examples

```
seq <- c("CARDRSTPWRRGIASTTVRTSW", "XXTQMYVR--XX", "CARJ", "10")
isValidAASeq(seq)
```

IUPAC_CODES *IUPAC ambiguous characters*

Description

A translation list mapping IUPAC ambiguous characters code to corresponding nucleotide amino acid characters.

Usage

```
IUPAC_DNA

IUPAC_AA
```

Format

A list with single character codes as names and values containing character vectors that define the set of standard characters that match to each each ambiguous character.

- IUPAC_DNA: DNA ambiguous character translations.
- IUPAC_AA: Amino acid ambiguous character translations.

---

makeChangeoClone *Generate a ChangeoClone object for lineage construction*

Description

makeChangeoClone takes a data.frame with Change-O style columns as input and masks gap positions, masks ragged ends, removes duplicates sequences, and merges annotations associated with duplicate sequences. It returns a ChangeoClone object which serves as input for lineage reconstruction.

Usage

```
makeChangeoClone(data, id = "SEQUENCE_ID", seq = "SEQUENCE_IMGT",
  germ = "GERMLINE_IMGT_D_MASK", vcall = "V_CALL", jcall = "J_CALL",
  junc_len = "JUNCTION_LENGTH", clone = "CLONE", max_mask = 0,
  text_fields = NULL, num_fields = NULL, seq_fields = NULL,
  add_count = TRUE)
```

Arguments

data	data.frame containing the Change-O data for a clone. See Details for the list of required columns and their default values.
id	name of the column containing sequence identifiers.
seq	name of the column containing observed DNA sequences. All sequences in this column must be multiple aligned.
germ	name of the column containing germline DNA sequences. All entries in this column should be identical for any given clone, and they must be multiple aligned with the data in the seq column.
vcall	name of the column containing V-segment allele assignments. All entries in this column should be identical to the gene level.
jcall	name of the column containing J-segment allele assignments. All entries in this column should be identical to the gene level.
junc_len	name of the column containing the length of the junction as a numeric value. All entries in this column should be identical for any given clone.
clone	name of the column containing the identifier for the clone. All entries in this column should be identical.

max_mask	maximum number of characters to mask at the leading and trailing sequence ends. If NULL then the upper masking bound will be automatically determined from the maximum number of observed leading or trailing Ns amongst all sequences. If set to 0 (default) then masking will not be performed.
text_fields	text annotation columns to retain and merge during duplicate removal.
num_fields	numeric annotation columns to retain and sum during duplicate removal.
seq_fields	sequence annotation columns to retain and collapse during duplicate removal. Note, this is distinct from the seq and germ arguments, which contain the primary sequence data for the clone and should not be repeated in this argument.
add_count	if TRUE add an additional annotation column called COLLAPSE_COUNT during duplicate removal that indicates the number of sequences that were collapsed.

Details

The input data.frame (data) must columns for each of the required column name arguments: id, seq, germ, vcall, jcall, junc_len, and clone. The default values are as follows:

- id = "SEQUENCE_ID": unique sequence identifier.
- seq = "SEQUENCE_IMGT": IMGT-gapped sample sequence.
- germ = "GERMLINE_IMGT_D_MASK": IMGT-gapped germline sequence.
- vcall = "V_CALL": V-segment allele call.
- jcall = "J_CALL": J-segment allele call.
- junc_len = "JUNCTION_LENGTH": junction sequence length.
- clone = "CLONE": clone identifier.

Additional annotation columns specified in the text_fields, num_fields or seq_fields arguments will be retained in the data slot of the return object, but are not required. If the input data.frame data already contains a column named SEQUENCE, which is not used as the seq argument, then that column will not be retained.

The default columns are IMGT-gapped sequence columns, but this is not a requirement. However, all sequences (both observed and germline) must be multiple aligned using some scheme for both proper duplicate removal and lineage reconstruction.

The value for the germline sequence, V-segment gene call, J-segment gene call, junction length, and clone identifier are determined from the first entry in the germ, vcall, jcall, junc_len and clone columns, respectively. For any given clone, each value in these columns should be identical.

Value

A ChangeoClone object containing the modified clone.

See Also

Executes in order maskSeqGaps, maskSeqEnds and collapseDuplicates. Returns a ChangeoClone object which serves as input to buildPhylipLineage.

Examples

```
# Example Change-O data.frame
db <- data.frame(SEQUENCE_ID=LETTERS[1:4],
                 SEQUENCE_IMGT=c("CCCCTGGG", "CCCCTGGN", "NAACTGGN", "NNNCTGNN"),
                 V_CALL="Homsap IGKV1-39*01 F",
                 J_CALL="Homsap IGKJ5*01 F",
                 JUNCTION_LENGTH=2,
                 GERMLINE_IMGT_D_MASK="CCCCAGGG",
                 CLONE=1,
                 TYPE=c("IgM", "IgG", "IgG", "IgA"),
                 COUNT=1:4,
                 stringsAsFactors=FALSE)

# Without end masking
makeChangeoClone(db, text_fields="TYPE", num_fields="COUNT")

# With end masking
makeChangeoClone(db, max_mask=3, text_fields="TYPE", num_fields="COUNT")
```

---

makeTempDir *Create a temporary folder*

Description

`makeTempDir` creates a randomly named temporary folder in the system temp location.

Usage

```
makeTempDir(prefix)
```

Arguments

`prefix`	prefix name for the folder.

Value

The path to the temporary folder.

See Also

This is just a wrapper for tempfile and dir.create.

Examples

```
makeTempDir("Clone50")
```

maskSeqEnds *Masks ragged leading and trailing edges of aligned DNA sequences*

Description

maskSeqEnds takes a vector of DNA sequences, as character strings, and replaces the leading and trailing characters with "N" characters to create a sequence vector with uniformly masked outer sequence segments.

Usage

```
maskSeqEnds(seq, max_mask = NULL, trim = FALSE)
```

Arguments

seq	a character vector of DNA sequence strings.
max_mask	the maximum number of characters to mask. If set to 0 then no masking will be performed. If set to NULL then the upper masking bound will be automatically determined from the maximum number of observed leading or trailing "N" characters amongst all strings in seq.
trim	if TRUE leading and trailing characters will be cut rather than masked with "N" characters.

Value

A modified seq vector with masked (or optionally trimmed) sequences.

See Also

See maskSeqGaps for masking internal gaps.

Examples

```
# Default behavior uniformly masks ragged ends
seq <- c("CCCCTGGG", "NAACTGGN", "NNNCTGNN")
maskSeqEnds(seq)

# Does nothing
maskSeqEnds(seq, max_mask=0)

# Cut ragged sequence ends
maskSeqEnds(seq, trim=TRUE)

# Set max_mask to limit extent of masking and trimming
maskSeqEnds(seq, max_mask=1)
maskSeqEnds(seq, max_mask=1, trim=TRUE)
```

maskSeqGaps *Masks gap characters in DNA sequences*

Description

maskSeqGaps substitutes gap characters, c("-", "."), with "N" in a vector of DNA sequences.

Usage

```
maskSeqGaps(seq, outer_only = FALSE)
```

Arguments

seq	a character vector of DNA sequence strings.
outer_only	if TRUE replace only contiguous leading and trailing gaps; if FALSE replace all gap characters.

Value

A modified seq vector with "N" in place of c("-", ".") characters.

See Also

See maskSeqEnds for masking ragged edges.

Examples

```
maskSeqGaps(c("ATG-C", "CC..C"))
maskSeqGaps("--ATG-C-")
maskSeqGaps("--ATG-C-", outer_only=TRUE)
```

MRCATest-class *S4 class defining edge significance*

Description

MRCATest defines the significance of enrichment for annotations appearing at the MRCA of the tree.

Usage

```
## S4 method for signature 'MRCATest'
print(x)

## S4 method for signature 'MRCATest,missing'
plot(x, y, ...)
```

Arguments

x	MRCATest object.
y	ignored.
...	arguments to pass to plotMRCATest.

Slots

`tests` data.frame describing the significance test results with columns:

- `ANNOTATION`: annotation value.
- `COUNT`: observed count of MRCA positions with the given annotation.
- `EXPECTED`: expected mean count of MRCA occurance for the annotation.
- `PVALUE`: one-sided p-value for the hypothesis that the observed annotation abundance is greater than expected.

`permutations` data.frame containing the raw permutation test data with columns:

- `ANNOTATION`: annotation value.
- `COUNT`: count of MRCA positions with the given annotation.
- `ITER`: numerical index define which permutation realization each observation corresponds to.

`nperm` number of permutation realizations.

`pairwiseDist` *Calculate pairwise distances between sequences*

Description

`pairwiseDist` calculates all pairwise distance between a set of sequences.

Usage

```
pairwiseDist(seq, dist_mat = getDNAMatrix())
```

Arguments

`seq`	character vector containing a DNA sequences.
`dist_mat`	Character distance matrix. Defaults to a Hamming distance matrix returned by getDNAMatrix. If gap characters, `c("-", ".")`, are assigned a value of -1 in `dist_mat` then contiguous gaps of any run length, which are not present in both sequences, will be counted as a distance of 1. Meaning, indels of any length will increase the sequence distance by 1. Gap values other than -1 will return a distance that does not consider indels as a special case.

Value

A matrix of numerical distance between each entry in `seq`. If `seq` is a named vector, row and columns names will be added accordingly.

See Also

Nucleotide distance matrix may be built with getDNAMatrix. Amino acid distance matrix may be built with getAAMatrix. Uses seqDist for calculating distances between pairs. See pairwiseEqual for generating an equivalence matrix.

Examples

```
# Gaps will be treated as Ns with a gap=0 distance matrix
pairwiseDist(c(A="ATGGC", B="ATGGG", C="ATGGG", D="AT--C"),
             dist_mat=getDNAMatrix(gap=0))

# Gaps will be treated as universally non-matching characters with gap=1
pairwiseDist(c(A="ATGGC", B="ATGGG", C="ATGGG", D="AT--C"),
             dist_mat=getDNAMatrix(gap=1))

# Gaps of any length will be treated as single mismatches with a gap=-1 distance matrix
pairwiseDist(c(A="ATGGC", B="ATGGG", C="ATGGG", D="AT--C"),
             dist_mat=getDNAMatrix(gap=-1))
```

---

pairwiseEqual *Calculate pairwise equivalence between sequences*

Description

`pairwiseEqual` determined pairwise equivalence between a pairs in a set of sequences, excluding ambiguous positions (Ns and gaps).

Usage

```
pairwiseEqual(seq)
```

Arguments

`seq`	character vector containing a DNA sequences.

Value

A logical matrix of equivalence between each entry in `seq`. Values are `TRUE` when sequences are equivalent and `FALSE` when they are not.

See Also

Uses seqEqual for testing equivalence between pairs. See pairwiseDist for generating a sequence distance matrix.

Examples

```
# Gaps and Ns will match any character
seq <- c(A="ATGGC", B="ATGGG", C="ATGGG", D="AT--C", E="NTGGG")
d <- pairwiseEqual(seq)
rownames(d) <- colnames(d) <- seq
d
```

`permuteLabels` *Permute the node labels of a tree*

Description

`permuteLabels` permutes the node annotations of a lineage tree.

Usage

```
permuteLabels(graph, field, exclude = c("Germline", NA))
```

Arguments

`graph`	igraph object containing an annotated lineage tree.
`field`	string defining the annotation field to permute.
`exclude`	vector of strings defining `field` values to exclude from permutation.

Value

A modified igraph object with vertex annotations permuted.

See Also

testEdges.

Examples

```
# Define and plot example graph
library(igraph)
graph <- ExampleTrees[[23]]
plot(graph, layout=layout_as_tree, vertex.label=V(graph)$ISOTYPE,
     vertex.size=50, edge.arrow.mode=0, vertex.color="grey80")

# Permute annotations and plot new tree
g <- permuteLabels(graph, "ISOTYPE")
plot(g, layout=layout_as_tree, vertex.label=V(g)$ISOTYPE,
     vertex.size=50, edge.arrow.mode=0, vertex.color="grey80")
```

plotAbundance *Plots a clonal abundance distribution*

Description

plotAbundance plots the results from estimating the complete clonal relative abundance distribution. The distribution is plotted as a log rank abundance distribution.

Usage

```
plotAbundance(data, colors = NULL, main_title = "Rank Abundance",
  legend_title = NULL, xlim = NULL, ylim = NULL, silent = FALSE, ...)
```

Arguments

data	data.frame returned by estimateAbundance.
colors	named character vector whose names are values in the group column of data and whose values are colors to assign to those group names.
main_title	string specifying the plot title.
legend_title	string specifying the legend title.
xlim	numeric vector of two values specifying the c(lower, upper) x-axis limits.
ylim	numeric vector of two values specifying the c(lower, upper) y-axis limits.
silent	if TRUE do not draw the plot and just return the ggplot2 object; if FALSE draw the plot.
...	additional arguments to pass to ggplot2::theme.

Value

A ggplot object defining the plot.

See Also

See estimateAbundance for generating the input abundance distribution. Plotting is performed with ggplot.

Examples

```
# Estimate abundance by sample and plot
abund <- estimateAbundance(ExampleDb, "SAMPLE", nboot=100)
plotAbundance(abund)
```

plotDiversityCurve *Plot the results of rarefyDiversity*

Description

plotDiversityCurve plots a DiversityCurve object.

Usage

```
plotDiversityCurve(data, colors = NULL, main_title = "Diversity",
  legend_title = "Group", log_q = TRUE, log_d = TRUE, xlim = NULL,
  ylim = NULL, annotate = c("none", "depth"), silent = FALSE, ...)
```

Arguments

data	DiversityCurve object returned by rarefyDiversity.
colors	named character vector whose names are values in the group column of the data slot of data, and whose values are colors to assign to those group names.
main_title	string specifying the plot title.
legend_title	string specifying the legend title.
log_q	if TRUE then plot q on a log scale; if FALSE plot on a linear scale.
log_d	if TRUE then plot the diversity scores D on a log scale; if FALSE plot on a linear scale.
xlim	numeric vector of two values specifying the c(lower, upper) x-axis limits.
ylim	numeric vector of two values specifying the c(lower, upper) y-axis limits.
annotate	string defining whether to added values to the group labels of the legend. When "none" (default) is specified no annotations are added. Specifying ("depth") adds sequence counts to the labels.
silent	if TRUE do not draw the plot and just return the ggplot2 object; if FALSE draw the plot.
...	additional arguments to pass to ggplot2::theme.

Value

A ggplot object defining the plot.

See Also

See rarefyDiversity for generating DiversityCurve objects for input. Plotting is performed with ggplot.

Examples

```
# All groups pass default minimum sampling threshold of 10 sequences
div <- rarefyDiversity(ExampleDb, "SAMPLE", step_q=0.1, max_q=10, nboot=100)
plotDiversityCurve(div, legend_title="Sample")
```

plotEdgeTest *Plot the results of an edge permutation test*

Description

plotEdgeTest plots the results of an edge permutation test performed with testEdges as either a histogram or cumulative distribution function.

Usage

```
plotEdgeTest(data, color = "black", main_title = "Edge Test",
  style = c("histogram", "cdf"), silent = FALSE, ...)
```

Arguments

data	EdgeTest object returned by testEdges.
color	color of the histogram or lines.
main_title	string specifying the plot title.
style	type of plot to draw. One of: • "histogram": histogram of the edge count distribution with a red dotted line denoting the observed value. • "cdf": cumulative distribution function of edge counts with a red dotted line denoting the observed value and a blue dotted line indicating the p-value.
silent	if TRUE do not draw the plot and just return the ggplot2 object; if FALSE draw the plot.
...	additional arguments to pass to ggplot2::theme.

Value

A ggplot object defining the plot.

See Also

See testEdges for performing the test.

Examples

```
# Define example tree set
graphs <- ExampleTrees[1-10]

# Perform edge test on isotypes
x <- testEdges(graphs, "ISOTYPE", nperm=10)

# Plot
```

```
plotEdgeTest(x, color="steelblue", style="hist")
plotEdgeTest(x, style="cdf")
```

`plotMRCATest` *Plot the results of a founder permutation test*

Description

`plotMRCATest` plots the results of a founder permutation test performed with `testMRCA`.

Usage

```
plotMRCATest(data, color = "black", main_title = "MRCA Test",
  style = c("histogram", "cdf"), silent = FALSE, ...)
```

Arguments

`data`	MRCATest object returned by testMRCA.
`color`	color of the histogram or lines.
`main_title`	string specifying the plot title.
`style`	type of plot to draw. One of: • `"histogram"`: histogram of the annotation count distribution with a red dotted line denoting the observed value. • `"cdf"`: cumulative distribution function of annotation counts with a red dotted line denoting the observed value and a blue dotted line indicating the p-value.
`silent`	if `TRUE` do not draw the plot and just return the ggplot2 object; if `FALSE` draw the plot.
`...`	additional arguments to pass to ggplot2::theme.

Value

A `ggplot` object defining the plot.

See Also

See testEdges for performing the test.

Examples

```
# Define example tree set
graphs <- ExampleTrees[1-10]

# Perform MRCA test on isotypes
x <- testMRCA(graphs, "ISOTYPE", nperm=10)

# Plot
plotMRCATest(x, color="steelblue", style="hist")
plotMRCATest(x, style="cdf")
```

---

plotSubtrees *Plots subtree statistics for multiple trees*

Description

plotSubtree plots distributions of normalized subtree statistics for a set of lineage trees, broken down by annotation value.

Usage

```
plotSubtrees(graphs, field, stat, root = "Germline", exclude = c("Germline",
  NA), colors = NULL, main_title = "Subtrees",
  legend_title = "Annotation", style = c("box", "violin"), silent = FALSE,
  ...)
```

Arguments

graphs	list of igraph objects containing annotated lineage trees.
field	string defining the annotation field.
stat	string defining the subtree statistic to plot. One of: • outdegree: distribution of normalized node outdegrees. • size: distribution of normalized subtree sizes. • depth: distribution of subtree depths. • pathlength: distribution of maximum pathlength beneath nodes.
root	name of the root (germline) node.
exclude	vector of strings defining field values to exclude from plotting.
colors	named vector of colors for values in field, with names defining annotation names field column and values being colors. Also controls the order in which values appear on the plot. If NULL alphabetical ordering and a default color palette will be used.
main_title	string specifying the plot title.

legend_title	string specifying the legend title.
style	string specifying the style of plot to draw. One of: • "histogram": histogram of the annotation count distribution with a red dotted line denoting the observed value. • "cdf": cumulative distribution function of annotation counts with a red dotted line denoting the observed value and a blue dotted line indicating the p-value.
silent	if TRUE do not draw the plot and just return the ggplot2 object; if FALSE draw the plot.
...	additional arguments to pass to ggplot2::theme.

Value

A ggplot object defining the plot.

See Also

Subtree statistics are calculated with summarizeSubtrees.

Examples

```
# Define example tree set
graphs <- ExampleTrees[1-10]

# Plot violins of outdegree by sample
plotSubtrees(graphs, "SAMPLE", "out", main_title="Node outdegree",
             style="v")

# Plot violins of subtree by sample
plotSubtrees(graphs, "SAMPLE", "size", style="v")

# Plot boxplot of pathlength by isotype
plotSubtrees(graphs, "ISOTYPE", "path", colors=IG_COLORS,
             legend_title="Isotype", style="b")

# Plot boxplot of depth by isotype
plotSubtrees(graphs,  "ISOTYPE", "depth", style="b")
```

---

polar *Calculates the average polarity of amino acid sequences*

Description

polar calculates the average polarity score of amino acid sequences. Non-informative positions are excluded, where non-informative is defined as any character in c("X", "-", ".", "*").

Usage

```
polar(seq, polarity = NULL)
```

Arguments

seq	vector of strings containing amino acid sequences.
polarity	named numerical vector defining polarity scores for each amino acid, where names are single-letter amino acid character codes. If NULL, then the Grantham, 1974 scale is used.

Value

A vector of bulkiness scores for the sequence(s).

References

1. Grantham R. Amino acid difference formula to help explain protein evolution. Science 185, 862-864 (1974).

See Also

For additional size related indices see aaindex.

Examples

```
# Default scale
seq <- c("CARDRSTPWRRGIASTTVRTSW", "XXTQMYVRT")
polar(seq)

# Use the Zimmerman et al, 1968 polarity scale from the seqinr package
library(seqinr)
data(aaindex)
x <- aaindex[["ZIMJ680103"]]$I
# Rename the score vector to use single-letter codes
names(x) <- translateStrings(names(x), ABBREV_AA)
# Calculate polarity
polar(seq, polarity=x)
```

---

rarefyDiversity *Generate a clonal diversity index curve*

Description

rarefyDiversity divides a set of clones by a group annotation, uniformly resamples the sequences from each group, and calculates diversity scores (D) over an interval of diversity orders (q).

Usage

```
rarefyDiversity(data, group, clone = "CLONE", copy = NULL, min_q = 0,
  max_q = 4, step_q = 0.05, min_n = 30, max_n = NULL, ci = 0.95,
  nboot = 2000, progress = FALSE)
```

Arguments

data	data.frame with Change-O style columns containing clonal assignments.
group	name of the data column containing group identifiers.
clone	name of the data column containing clone identifiers.
copy	name of the data column containing copy numbers for each sequence. If copy=NULL (the default), then clone abundance is determined by the number of sequences. If a copy column is specified, then clone abundances is determined by the sum of copy numbers within each clonal group.
min_q	minimum value of q.
max_q	maximum value of q.
step_q	value by which to increment q.
min_n	minimum number of observations to sample. A group with less observations than the minimum is excluded.
max_n	maximum number of observations to sample. If NULL the maximum if automatically determined from the size of the largest group.
ci	confidence interval to calculate; the value must be between 0 and 1.
nboot	number of bootstrap realizations to generate.
progress	if TRUE show a progress bar.

Details

Clonal diversity is calculated using the generalized diversity index (Hill numbers) proposed by Hill (Hill, 1973). See calcDiversity for further details.

Diversity is calculated on the estimated complete clonal abundance distribution. This distribution is inferred by using the Chao1 estimator to estimate the number of seen clones, and applying the relative abundance correction and unseen clone frequency described in Chao et al, 2015.

To generate a smooth curve, D is calculated for each value of q from min_q to max_q incremented by step_q. Variability in total sequence counts across unique values in the group column is corrected by repeated resampling from the estimated complete clonal distribution to a common number of sequences.

The diversity index (D) for each group is the mean value of over all resampling realizations. Confidence intervals are derived using the standard deviation of the resampling realizations, as described in Chao et al, 2015.

Value

A DiversityCurve object summarizing the diversity scores.

References

1. Hill M. Diversity and evenness: a unifying notation and its consequences. Ecology. 1973 54(2):427-32.
2. Chao A. Nonparametric Estimation of the Number of Classes in a Population. Scand J Stat. 1984 11, 265270.
3. Chao A, et al. Rarefaction and extrapolation with Hill numbers: A framework for sampling and estimation in species diversity studies. Ecol Monogr. 2014 84:45-67.
4. Chao A, et al. Unveiling the species-rank abundance distribution by generalizing the Good-Turing sample coverage theory. Ecology. 2015 96, 11891201.

See Also

See calcDiversity for the basic calculation and DiversityCurve for the return object. See testDiversity for significance testing. See plotDiversityCurve for plotting the return object.

Examples

```
# Group by sample identifier
div <- rarefyDiversity(ExampleDb, "SAMPLE", step_q=1, max_q=10, nboot=100)
plotDiversityCurve(div, legend_title="Sample")

# Grouping by isotype rather than sample identifier
div <- rarefyDiversity(ExampleDb, "ISOTYPE", min_n=40, step_q=1, max_q=10,
                       nboot=100)
plotDiversityCurve(div, legend_title="Isotype")
```

---

readChangeoDb *Read a Change-O tab-delimited database file*

Description

`readChangeoDb` reads a tab-delimited database file created by a Change-O tool into a data.frame.

Usage

```
readChangeoDb(file, select = NULL, drop = NULL, seq_upper = TRUE)
```

Arguments

`file`	tab-delimited database file output by a Change-O tool.
`select`	columns to select from database file.
`drop`	columns to drop from database file.
`seq_upper`	if `TRUE` convert sequence columns to upper case; if `FALSE` do not alter sequence columns. See Value for a list of which columns are effected.

Value

A data.frame of the database file. Columns will be imported as is, except for the following columns which will be explicitly converted into character values:

- SEQUENCE_ID
- CLONE
- SAMPLE

And the following sequence columns which will converted to upper case if `seq_upper=TRUE` (default).

- SEQUENCE_INPUT
- SEQUENCE_VDJ
- SEQUENCE_IMGT
- JUNCTION
- GERMLINE_IMGT
- GERMLINE_IMGT_D_MASK

See Also

Wraps read.table and fread. See writeChangeoDb for writing to Change-O files.

Examples

```
## Not run:
    # Read all columns in and convert sequence fields to upper case
    db <- readChangeoDb("changeo.tsv")

    # Subset columns and convert sequence fields to upper case
    db <- readChangeoDb("changeo.tsv", select=c("SEQUENCE_ID", "SEQUENCE_IMGT"))

    # Drop columns and do not alter sequence field case
    db <- readChangeoDb("changeo.tsv", drop=c("D_CALL", "DUPCOUNT"),
                        seq_upper=FALSE)

## End(Not run)
```

---

seqDist *Calculate distance between two sequences*

Description

`seqDist` calculates the distance between two DNA sequences.

Usage

```
seqDist(seq1, seq2, dist_mat = getDNAMatrix())
```

Arguments

seq1	character string containing a DNA sequence.
seq2	character string containing a DNA sequence.
dist_mat	Character distance matrix. Defaults to a Hamming distance matrix returned by getDNAMatrix. If gap characters, c("-", "."), are assigned a value of -1 in dist_mat then contiguous gaps of any run length, which are not present in both sequences, will be counted as a distance of 1. Meaning, indels of any length will increase the sequence distance by 1. Gap values other than -1 will return a distance that does not consider indels as a special case.

Value

Numerical distance between seq1 and seq2.

See Also

Nucleotide distance matrix may be built with getDNAMatrix. Amino acid distance matrix may be built with getAAMatrix. Used by pairwiseDist for generating distance matrices. See seqEqual for testing sequence equivalence.

Examples

```
# Ungapped examples
seqDist("ATGGC", "ATGGG")
seqDist("ATGGC", "ATG??")

# Gaps will be treated as Ns with a gap=0 distance matrix
seqDist("ATGGC", "AT--C", dist_mat=getDNAMatrix(gap=0))

# Gaps will be treated as universally non-matching characters with gap=1
seqDist("ATGGC", "AT--C", dist_mat=getDNAMatrix(gap=1))

# Gaps of any length will be treated as single mismatches with a gap=-1 distance matrix
seqDist("ATGGC", "AT--C", dist_mat=getDNAMatrix(gap=-1))

# Gaps of equivalent run lengths are not counted as gaps
seqDist("ATG-C", "ATG-C", dist_mat=getDNAMatrix(gap=-1))

# Overlapping runs of gap characters are counted as a single gap
seqDist("ATG-C", "AT--C", dist_mat=getDNAMatrix(gap=-1))
seqDist("A-GGC", "AT--C", dist_mat=getDNAMatrix(gap=-1))
seqDist("AT--C", "AT--C", dist_mat=getDNAMatrix(gap=-1))

# Discontiguous runs of gap characters each count as separate gaps
seqDist("-TGGC", "AT--C", dist_mat=getDNAMatrix(gap=-1))
```

seqEqual *Test DNA sequences for equality.*

Description

seqEqual checks if two DNA sequences are identical.

Usage

```
seqEqual(seq1, seq2, ignore = as.character(c("N", "-", ".", "?")))
```

Arguments

seq1	character string containing a DNA sequence.
seq2	character string containing a DNA sequence.
ignore	vector of characters to ignore when testing for equality. Default is to ignore c("N",".","-","?")

Value

Returns TRUE if sequences are equal and FALSE if they are not. Sequences of unequal length will always return FALSE regardless of their character values.

See Also

Used by pairwiseEqual within collapseDuplicates. See seqDist for calculation Hamming distances between sequences.

Examples

```
# Ignore gaps
seqEqual("ATG-C", "AT--C")
seqEqual("ATGGC", "ATGGN")
seqEqual("AT--T", "ATGGC")

# Ignore only Ns
seqEqual("ATG-C", "AT--C", ignore="N")
seqEqual("ATGGC", "ATGGN", ignore="N")
seqEqual("AT--T", "ATGGC", ignore="N")
```

sortGenes *Sort V(D)J genes*

Description

sortGenes sorts a vector of V(D)J gene names by either lexicographic ordering or locus position.

Usage

```
sortGenes(genes, method = c("name", "position"))
```

Arguments

genes	vector of strings respresenting V(D)J gene names.
method	string defining the method to use for sorting genes. One of: • "name": sort in lexicographic order. Order is by family first, then gene, and then allele. • "position": sort by position in the locus, as determined by the final two numbers in the gene name. Non-localized genes are assigned to the highest positions.

Value

A sorted character vector of gene names.

See Also

See getAllele, getGene and getFamily for parsing gene names.

Examples

```
# Create a list of allele names
genes <- c("IGHV1-69D*01","IGHV1-69*01","IGHV4-38-2*01","IGHV1-69-2*01",
           "IGHV2-5*01","IGHV1-NL1*01", "IGHV1-2*01,IGHV1-2*05",
           "IGHV1-2", "IGHV1-2*02", "IGHV1-69*02")

# Sort genes by name
sortGenes(genes)

# Sort genes by position in the locus
sortGenes(genes, method="pos")
```

stoufferMeta *Weighted meta-analysis of p-values via Stouffer's method*

Description

stoufferMeta combines multiple weighted p-values into a meta-analysis p-value using Stouffer's Z-score method.

Usage

```
stoufferMeta(p, w = NULL)
```

Arguments

p	numeric vector of p-values.
w	numeric vector of weights.

Value

A named numeric vector with the combined Z-score and p-value in the form c(Z, pvalue).

Examples

```
# Define p-value and weight vectors
p <- c(0.1, 0.05, 0.3)
w <- c(5, 10, 1)

# Unweighted
stoufferMeta(p)

# Weighted
stoufferMeta(p, w)
```

summarizeSubtrees *Generate subtree summary statistics for a tree*

Description

summarizeSubtrees calculates summary statistics for each node of a tree. Includes both node properties and subtree properties.

Usage

```
summarizeSubtrees(graph, fields = NULL, root = "Germline")
```

Arguments

graph	igraph object containing an annotated lineage tree.
fields	annotation fields to add to the output.
root	name of the root (germline) node.

Value

A data.frame with columns:

- NAME: node name.
- PARENT: name of the parent node.
- OUTDEGREE: number of edges leading from the node.
- SIZE: total number of nodes within the subtree rooted at the node.
- DEPTH: the depth of the subtree that is rooted at the node.
- PATHLENGTH: the maximum pathlength beneath the node.
- OUTDEGREE_NORM: OUTDEGREE normalized by the total number of edges.
- SIZE_NORM: SIZE normalized by the largest subtree size (the germline).
- DEPTH_NORM: DEPTH normalized by the largest subtree depth (the germline).
- PATHLENGTH_NORM: PATHLEGNTH normalized by the largest subtree pathlength (the germline).

An additional column corresponding to the value of field is added when specified.

See Also

See buildPhylipLineage for generating input trees. See getPathLengths for calculating path length to nodes.

Examples

```
# Summarize a tree
graph <- ExampleTrees[[23]]
summarizeSubtrees(graph, fields="ISOTYPE", root="Germline")
```

---

tableEdges *Tabulate the number of edges between annotations within a lineage tree*

Description

tableEdges creates a table of the total number of connections (edges) for each unique pair of annotations within a tree over all nodes.

Usage

```
tableEdges(graph, field, indirect = FALSE, exclude = NULL)
```

Arguments

graph	igraph object containing an annotated lineage tree.
field	string defining the annotation field to count.
indirect	if FALSE count direct connections (edges) only. If TRUE walk through any nodes with annotations specified in the argument to count indirect connections. Specifying indirect=TRUE with exclude=NULL will have no effect.
exclude	vector of strings defining field values to exclude from counts. Edges that either start or end with the specified annotations will not be counted. If NULL count all edges.

Value

A data.frame defining total annotation connections in the tree with columns:

- PARENT: parent annotation
- CHILD: child annotation
- COUNT: count of edges for the parent-child relationship

See Also

See testEdges for performed a permutation test on edge relationships.

Examples

```
# Define example graph
graph <- ExampleTrees[[23]]

# Count direct edges between isotypes including inferred nodes
tableEdges(graph, "ISOTYPE")

# Count direct edges excluding edges to and from germline and inferred nodes
tableEdges(graph, "ISOTYPE", exclude=c("Germline", NA))

# Count indirect edges walking through germline and inferred nodes
tableEdges(graph, "ISOTYPE", indirect=TRUE, exclude=c("Germline", NA))
```

---

testDiversity *Pairwise test of the diversity index*

Description

testDiversity performs pairwise significance tests of the diversity index (D) at a given diversity order (q) for a set of annotation groups via rarefaction and bootstrapping.

Usage

```
testDiversity(data, q, group, clone = "CLONE", copy = NULL, min_n = 30,
  max_n = NULL, nboot = 2000, progress = FALSE)
```

Arguments

data	data.frame with Change-O style columns containing clonal assignments.
q	diversity order to test.
group	name of the data column containing group identifiers.
clone	name of the data column containing clone identifiers.
copy	name of the data column containing copy numbers for each sequence. If copy=NULL (the default), then clone abundance is determined by the number of sequences. If a copy column is specified, then clone abundances is determined by the sum of copy numbers within each clonal group.
min_n	minimum number of observations to sample. A group with less observations than the minimum is excluded.
max_n	maximum number of observations to sample. If NULL the maximum if automatically determined from the size of the largest group.
nboot	number of bootstrap realizations to perform.
progress	if TRUE show a progress bar.

Details

Clonal diversity is calculated using the generalized diversity index proposed by Hill (Hill, 1973). See calcDiversity for further details.

Diversity is calculated on the estimated complete clonal abundance distribution. This distribution is inferred by using the Chao1 estimator to estimate the number of seen clones, and applying the relative abundance correction and unseen clone frequency described in Chao et al, 2014.

Variability in total sequence counts across unique values in the group column is corrected by repeated resampling from the estimated complete clonal distribution to a common number of sequences. The diversity index estimate (D) for each group is the mean value of over all bootstrap realizations.

Significance of the difference in diversity index (D) between groups is tested by constructing a bootstrap delta distribution for each pair of unique values in the group column. The bootstrap delta distribution is built by subtracting the diversity index Da in $group - a$ from the corresponding value Db in $group - b$, for all bootstrap realizations, yeilding a distribution of nboot total deltas; where $group - a$ is the group with the greater mean D. The p-value for hypothesis $Da! = Db$ is the value of $P(0)$ from the empirical cumulative distribution function of the bootstrap delta distribution, multiplied by 2 for the two-tailed correction.

Value

A DiversityTest object containing p-values and summary statistics.

Note

This method may inflate statistical significance when clone sizes are uniformly small, such as when most clones sizes are 1, sample size is small, and `max_n` is near the total count of the smallest data group. Use caution when interpreting the results in such cases. We are currently investigating this potential problem.

References

1. Hill M. Diversity and evenness: a unifying notation and its consequences. Ecology. 1973 54(2):427-32.
2. Chao A. Nonparametric Estimation of the Number of Classes in a Population. Scand J Stat. 1984 11, 265270.
3. Wu Y-CB, et al. Influence of seasonal exposure to grass pollen on local and peripheral blood IgE repertoires in patients with allergic rhinitis. J Allergy Clin Immunol. 2014 134(3):604-12.
4. Chao A, et al. Rarefaction and extrapolation with Hill numbers: A framework for sampling and estimation in species diversity studies. Ecol Monogr. 2014 84:45-67.
5. Chao A, et al. Unveiling the species-rank abundance distribution by generalizing the Good-Turing sample coverage theory. Ecology. 2015 96, 11891201.

See Also

See calcDiversity for the basic calculation and DiversityTest for the return object. See rarefyDiversity for curve generation. See ecdf for computation of the empirical cumulative distribution function.

Examples

```
# Groups under the size threshold are excluded and a warning message is issued.
testDiversity(ExampleDb, "SAMPLE", q=0, min_n=30, nboot=100)
```

`testEdges` *Tests for parent-child annotation enchrichment in lineage trees*

Description

`testEdges` performs a permutation test on a set of lineage trees to determine the significance of an annotation's association with parent-child relationships.

Usage

```
testEdges(graphs, field, indirect = FALSE, exclude = c("Germline", NA),
  nperm = 200, progress = FALSE)
```

Arguments

graphs	list of igraph objects with vertex annotations.
field	string defining the annotation field to permute.
indirect	if FALSE count direct connections (edges) only. If TRUE walk through any nodes with annotations specified in the argument to count indirect connections. Specifying indirect=TRUE with exclude=NULL will have no effect.
exclude	vector of strings defining field values to exclude from permutation.
nperm	number of permutations to perform.
progress	if TRUE show a progress bar.

Value

An EdgeTest object containing the test results and permutation realizations.

See Also

Uses tableEdges and permuteLabels. See plotEdgeTest for plotting the permutation distributions.

Examples

```
# Define example tree set
graphs <- ExampleTrees[1-10]

# Perform edge test on isotypes
x <- testEdges(graphs, "ISOTYPE", nperm=10)
print(x)
```

---

testMRCA *Tests for MRCA annotation enrichment in lineage trees*

Description

testMRCA performs a permutation test on a set of lineage trees to determine the significance of an annotation's association with the MRCA position of the lineage trees.

Usage

```
testMRCA(graphs, field, root = "Germline", exclude = c("Germline", NA),
  nperm = 200, progress = FALSE)
```

Arguments

graphs	list of igraph object containing annotated lineage trees.
field	string defining the annotation field to test.
root	name of the root (germline) node.
exclude	vector of strings defining field values to exclude from the set of potential founder annotations.
nperm	number of permutations to perform.
progress	if TRUE show a progress bar.

Value

An MRCATest object containing the test results and permutation realizations.

See Also

Uses getMRCA and getPathLengths. See plotMRCATest for plotting the permutation distributions.

Examples

```
# Define example tree set
graphs <- ExampleTrees[1-10]

# Perform MRCA test on isotypes
x <- testMRCA(graphs, "ISOTYPE", nperm=10)
print(x)
```

---

translateDNA *Translate nucleotide sequences to amino acids*

Description

translateDNA translates nucleotide sequences to amino acid sequences.

Usage

```
translateDNA(seq, trim = FALSE)
```

Arguments

seq	vector of strings defining DNA sequence(s) to be converted to translated.
trim	boolean flag to remove 3 nts from both ends of seq (converts IMGT junction to CDR3 region).

Value

A vector of translated sequence strings.

See Also

translate.

Examples

```
# Translate a single sequence
translateDNA("ACTGACTCGA")

# Translate a vector of sequences
translateDNA(ExampleDb$JUNCTION[1:3])

# Remove the first and last codon from the translation
translateDNA(ExampleDb$JUNCTION[1:3], trim=TRUE)
```

---

translateStrings *Translate a vector of strings*

Description

translateStrings modifies a character vector by substituting one or more strings with a replacement string.

Usage

```
translateStrings(strings, translation)
```

Arguments

strings	vector of character strings to modify.
translation	named character vector or a list of character vectors specifying the strings to replace (values) and their replacements (names).

Details

Does not perform partial replacements. Each translation value must match a complete strings value or it will not be replaced. Values that do not have a replacement named in the translation parameter will not be modified.

Replacement is accomplished using gsub.

Value

A modified strings vector.

See Also

See gsub for single value replacement in the base package.

Examples

```
# Using a vector translation
strings <- LETTERS[1:5]
translation <- c("POSITION1"="A", "POSITION5"="E")
translateStrings(strings, translation)

# Using a list translation
strings <- LETTERS[1:5]
translation <- list("1-3"=c("A","B","C"), "4-5"=c("D","E"))
translateStrings(strings, translation)
```

---

writeChangeoDb *Write a Change-O tab-delimited database file*

Description

writeChangeoDb is a simple wrapper around write.table with defaults appropriate for writing a Change-O tab-delimited database file from a data.frame.

Usage

```
writeChangeoDb(data, file)
```

Arguments

data	data.frame of Change-O data.
file	output file name.

See Also

Wraps write.table. See readChangeoDb for reading to Change-O files.

Examples

```
## Not run:
  # Write a database
  writeChangeoDb(data, "changeo.tsv")

## End(Not run)
```

Index