Package 'LymphoSeq'

October 16, 2019

Title Analyze high-throughput sequencing of T and B cell receptors

Version 1.12.0

Author David Coffey <dcoffey@fredhutch.org>

Maintainer David Coffey <dcoffey@fredhutch.org>

Description This R package analyzes high-throughput sequencing of T and B cell receptor complementarity determining region 3 (CDR3) sequences generated by Adaptive Biotechnologies' ImmunoSEQ assay. Its input comes from tab-separated value (.tsv) files exported from the ImmunoSEQ analyzer.

Depends R (>= 3.3), LymphoSeqDB

Imports data.table, plyr, dplyr, reshape, VennDiagram, ggplot2, ineq, RColorBrewer, circlize, grid, utils, stats, ggtree, msa, Biostrings, phangorn, stringdist, UpSetR

License Artistic-2.0

LazyData true

RoxygenNote 6.0.1

Suggests knitr, pheatmap, wordcloud, rmarkdown

VignetteBuilder knitr

biocViews Software, Technology, Sequencing, TargetedResequencing, Alignment, MultipleSequenceAlignment

NeedsCompilation no

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

git_branch RELEASE_3_9

git_last_commit ba3e8d5

git_last_commit_date 2019-05-02

Date/Publication 2019-10-15

R topics documented:

`alignSeq` *Align mutliple sequences*

Description

Perform multiple sequence alignment using one of three methods and output results to the console or as a pdf file. One may perform the alignment of all amino acid or nucleotide sequences in a single sample. Alternatively, one may search for a given sequence within a list of samples using an edit distance threshold.

Usage

```
alignSeq(list, sample = NULL, sequence = NULL, editDistance = 15,
  output = "console", type = "nucleotide", method = "ClustalOmega")
```

Arguments

`list`	A list of data frames consisting of antigen receptor sequences imported by the LymphoSeq function readImmunoSeq.
`sample`	A character vector indicating the name of the sample in the productive sequence list.
`sequence`	A character vector of one ore more amino acid or nucleotide CDR3 sequences to search.

editDistance	An integer giving the minimum edit distance that the sequence must be less than or equal to. See details below.
output	A character vector indicating where the multiple sequence alignemnt should be printed. Options include "console" or "pdf". If "pdf" is selected, the file is saved to the working directory. For "pdf" to work, Texshade must be installed. Refer to the Bioconductor package msa installation instructions for more details.
type	A character vector indicating whether "aminoAcid" or "nucleotide" sequences should be aligned. If "aminoAcid" is specified, then run productiveSeqs first.
method	A character vector indicating the multiple sequence alignment method to be used. Refer to the Bioconductor msa package for more details. Options incude "ClustalW", "ClustalOmega", and "Muscle".

Details

Edit distance is a way of quantifying how dissimilar two sequences are to one another by counting the minimum number of operations required to transform one sequence into the other. For example, an edit distance of 0 means the sequences are identical and an edit distance of 1 indicates that the sequences different by a single amino acid or nucleotide.

Value

Performs a multiple sequence alignemnt and prints to the console or saves a pdf to the working directory.

See Also

If having trouble saving pdf files, refer to Biconductor package msa for installation instructions http://bioconductor.org/packages/release/bioc/vignettes/msa/inst/doc/msa.pdf

Examples

```
file.path <- system.file("extdata", "IGH_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.nt <- productiveSeq(file.list = file.list, aggregate = "nucleotide")

alignSeq(list = productive.nt, sample = "IGH_MVQ92552A_BL", type = "nucleotide",
         method = "ClustalW", output = "console")
```

---

bhattacharyyaCoefficient

Bhattacharyya coefficient

Description

Calculates the Bhattacharyya coefficient of two samples.

Usage

```
bhattacharyyaCoefficient(sample1, sample2)
```

Arguments

sample1	A data frame consisting of frequencies of antigen receptor sequences. "frequencyCount" is a required column.
sample2	A data frame consisting of frequencies of antigen receptor sequences. "frequencyCount" is a required column.

Value

Returns the Bhattacharyya coefficient, a measure of the amount of overlap between two samples. The value ranges from 0 to 1 where 1 indicates the sequence frequencies are identical in the two samples and 0 indicates no shared frequencies.

See Also

bhattacharyyaMatrix

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.aa <- productiveSeq(file.list, aggregate = "aminoAcid")

bhattacharyyaCoefficient(productive.aa[["TRB_Unsorted_32"]],
   productive.aa[["TRB_Unsorted_83"]])
```

---

bhattacharyyaMatrix *Bhattacharyya matrix*

Description

Calculates the Bhattacharyya coefficient of all pairwise comparison from a list of data frames.

Usage

```
bhattacharyyaMatrix(productive.seqs)
```

Arguments

productive.seqs	A list data frames of productive sequences generated by the LymphoSeq function productiveSeq. "frequencyCount" and "aminoAcid" are a required columns.

Value

A data frame of Bhattacharyya coefficients calculated from all pairwise comparisons from a list of sample data frames. The Bhattacharyya coefficient is a measure of the amount of overlap between two samples. The value ranges from 0 to 1 where 1 indicates the sequence frequencies are identical in the two samples and 0 indicates no shared frequencies.

See Also

pairwisePlot for plotting results as a heat map.

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.aa <- productiveSeq(file.list, aggregate = "aminoAcid")

bhattacharyyaMatrix(productive.seqs = productive.aa)
```

---

chordDiagramVDJ *Chord diagram of VJ or DJ gene associations*

Description

Creates a chord diagram showing VJ or DJ gene associations from one or more samples.

Usage

```
chordDiagramVDJ(sample, association = "VJ", colors = c("red", "blue"))
```

Arguments

sample	A data frame consisting of frequencies of antigen receptor sequences. "vFamilyName", "jFamilyName", and if applicable, "dFamilyName" are a required columns. Using output from the LymphoSeq function topSeqs is recommended.
association	A character vector of gene familes to associate. Options include "VJ" or "DJ".
colors	A character vector of 2 colors corresponding to the V/D and J gene colors respectively.

Details

The size of the ribbons connecting VJ or DJ genes correspond to the number of samples or number of sequences that make up that recombination event. The thicker the ribbon, the higher the frequency of the recombination.

Value

Returns a chord diagram showing VJ or DJ gene associations from one or more samples.

See Also

topSeqs

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.nt <- productiveSeq(file.list = file.list, aggregate = "nucleotide")

top.seqs <- topSeqs(productive.seqs = productive.nt, top = 1)

chordDiagramVDJ(sample = top.seqs, association = "VJ", colors = c("red", "blue"))

# Remove "TCRB" from gene family name
top.seqs <- as.data.frame(apply(top.seqs, 2, function(x) gsub("TCRB", "", x)))

chordDiagramVDJ(sample = top.seqs, association = "VJ", colors = c("red", "blue"))
```

`clonality` *Clonality*

Description

Creates a data frame giving the total number of sequences, number of unique productive sequences, number of genomes, entropy, clonality, Gini coefficient, and the frequency (%) of the top productive sequences in a list of sample data frames.

Usage

```
clonality(file.list)
```

Arguments

`file.list`	A list of data frames consisting of antigen receptor sequencing imported by the LymphoSeq function readImmunoSeq. "aminoAcid", "count", and "frequencyCount" are required columns. "estimatedNumberGenomes" is optional. Note that clonality is usually calculated from productive nucleotide sequences. Therefore, it is not recommended to run this function using a productive sequence list aggregated by amino acids.

Details

Clonality is derived from the Shannon entropy, which is calculated from the frequencies of all productive sequences divided by the logarithm of the total number of unique productive sequences. This normalized entropy value is then inverted (1 - normalized entropy) to produce the clonality metric.

The Gini coefficient is an alternative metric used to calculate repertoire diversity and is derived from the Lorenz curve. The Lorenz curve is drawn such that x-axis represents the cumulative percentage of unique sequences and the y-axis represents the cumulative percentage of reads. A line passing through the origin with a slope of 1 reflects equal frequencies of all clones. The Gini coefficient is the ratio of the area between the line of equality and the observed Lorenz curve over the total area under the line of equality. Both Gini coefficient and clonality are reported on a scale from 0 to 1 where 0 indicates all sequences have the same frequency and 1 indicates the repertoire is dominated by a single sequence.

Value

Returns a data frame giving the total number of sequences, number of unique productive sequences, number of genomes, clonality, Gini coefficient, and the frequency (%) of the top productive sequence in each sample.

See Also

lorenzCurve

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

clonality(file.list = file.list)
```

---

clonalRelatedness *Clonal relatedness*

Description

Calculates the clonal relatedness for each sample in a list of data frames.

Usage

```
clonalRelatedness(list, editDistance = 10)
```

Arguments

list	A list data frames of unproductive or productive nucleotide sequences or productive nucleotide sequences. Nucleotide and count are required columns.
editDistance	An integer giving the minimum edit distance that the sequence must be less than or equal to. See details below.

Details

Clonal relatedness is the proportion of nucleotide sequences that are related by a defined edit distance threshold. The value ranges from 0 to 1 where 0 indicates no sequences are related and 1 indicates all sequences are related.

Edit distance is a way of quantifying how dissimilar two sequences are to one another by counting the minimum number of operations required to transform one sequence into the other. For example, an edit distance of 0 means the sequences are identical and an edit distance of 1 indicates that the sequences different by a single amino acid or nucleotide.

Value

Returns a data frame with the calculated clonal relatedness for each sample.

Examples

```
file.path <- system.file("extdata", "IGH_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

clonal.relatedness <- clonalRelatedness(list = file.list, editDistance = 10)

# Merge results with clonality table
clonality <- clonality(file.list = file.list)
merged <- merge(clonality, clonal.relatedness)
```

---

cloneTrack *Clone tracking plot*

Description

Creates line plot tracking amino acid frequencies across multiple samples

Usage

```
cloneTrack(sequence.matrix, map = "none", productive.aa, label = "none",
  track = "none", unassigned = TRUE)
```

Arguments

sequence.matrix	A sequence matrix generated from the LymphoSeq function seqMatrix.
map	An optional character vector of one or more sample names contained in the productive.aa list. If the same sequence appears in multiple mapped samples, then it will be assigned the label of the first listed sample only.
productive.aa	A list of data frames of productive amino acid sequences containing the samples to be mapped. This parameter is only required if sequence mapping is performed.
label	An optional character vector of one or more labels used to annotate the mapped sequences. The order of the labels must match the order of the samples listed in map.
track	An optional character vector of one or more amino acid sequences to track.
unassigned	A Boolean value indicating whether or not to draw the lines of sequences not being mapped or tracked. If TRUE then the unassigned sequences are drawn. If FALSE, then the unassigned sequences are not drawn.

Details

The plot is made using the package ggplot2 and can be reformatted using ggplot2 functions. See examples below.

Value

Returns a line plot showing the amino acid frequencies across multiple samples in the sequence matrix where each line represents one unique sequence.

See Also

An excellent resource for examples on how to reformat a ggplot can be found in the R Graphics Cookbook online (http://www.cookbook-r.com/Graphs/).

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.aa <- productiveSeq(file.list = file.list, aggregate = "aminoAcid")

top.freq <- topFreq(productive.aa = productive.aa, percent = 0.1)

sequence.matrix <- seqMatrix(productive.aa = productive.aa, sequences = top.freq$aminoAcid)

# Track clones without mapping or tracking specific sequences
cloneTrack(sequence.matrix = sequence.matrix)

# Track top 20 clones mapping to the CD4 and CD8 samples
cloneTrack(sequence.matrix = sequence.matrix, productive.aa = productive.aa,
   map = c("TRB_CD4_949", "TRB_CD8_949"), label = c("CD4", "CD8"),
   track = top.freq$aminoAcid[1:20], unassigned = TRUE)

# Track the top 10 clones from top.freq
cloneTrack(sequence.matrix = sequence.matrix, productive.aa = productive.aa,
   track = top.freq$aminoAcid[1:10], unassigned = FALSE)

# Track clones mapping to the CD4 and CD8 samples while ignoring all others
cloneTrack(sequence.matrix = sequence.matrix, productive.aa = productive.aa,
   map = c("TRB_CD4_949", "TRB_CD8_949"), label = c("CD4", "CD8"),
   unassigned = FALSE)

# Track clones mapping to the CD4 and CD8 samples and track 2 specific sequences
cloneTrack(sequence.matrix = sequence.matrix, productive.aa = productive.aa,
   map = c("TRB_CD4_949", "TRB_CD8_949"), label = c("CD4", "CD8"),
   track = c("CASSPPTGERDTQYF", "CASSQDRTGQYGYTF"), unassigned = FALSE)

# Reorder the x axis, change the axis labels, convert to log scale, and add title
x.limits <- c("TRB_Unsorted_0", "TRB_Unsorted_32",
   "TRB_Unsorted_83", "TRB_Unsorted_949", "TRB_Unsorted_1320")

sequence.matrix <- sequence.matrix[ ,c("aminoAcid", x.limits)]

clone.track <- cloneTrack(sequence.matrix = sequence.matrix,
   productive.aa = productive.aa, track = top.freq$aminoAcid[1:10], unassigned = FALSE)

x.labels <- c("Day 0", "Day 32", "Day 83", "Day 949", "Day 1320")

clone.track +
   ggplot2::scale_x_discrete(expand = c(0,0), labels = x.labels) +
   ggplot2::scale_y_log10() + ggplot2::annotation_logticks(sides = "l") +
   ggplot2::ggtitle("Figure Title")
```

commonSeqs *Common sequences in two or more samples*

Description

Creates a data frame of the common sequences in two or more samples, reporting their frequencies in each.

Usage

```
commonSeqs(samples, productive.aa)
```

Arguments

samples	A character vector of two or more sample names in productive.aa.
productive.aa	A list of productive amino acid sequences generated by the LymphoSeq function productiveSeq where aggregate = "aminoAcid".

Value

Returns a data frame of the common sequences between two or more files displaying their frequencies in each.

See Also

commonSeqsVenn

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.aa <- productiveSeq(file.list = file.list, aggregate = "aminoAcid")

commonSeqs(samples = c("TRB_Unsorted_0", "TRB_Unsorted_32"),
   productive.aa = productive.aa)
```

commonSeqsBar *Common sequences bar plot*

Description

Creates an UpSetR bar plot showing the number of intersecting sequences across multiple samples. This function is useful when more than 3 samples are being compared.

Usage

```
commonSeqsBar(productive.aa, samples, color.sample = NULL,
  color.intersection = NULL, color = "#377eb8", labels = "no")
```

Arguments

productive.aa	A list data frames of of productive amino acid sequences generated by LymphoSeq function productiveSeq where the aggregate parameter was set to "aminoAcid".
samples	The names of two or more samples in the productive.aa list whose intersections will shown.
color.sample	The name of a single sample in the productive.aa list whose sequences will be colored in all samples that they appear in.
color.intersection	The names of two or more samples in the productive.aa list whose intersections will be colored.
color	A character vector of a color name that will be used highlight a selected sample or multiple sample intersections.
labels	A character vector indicating whether the number of intersecting sequences should be shown on the tops of the bars. Options include "yes" or "no".

Value

Returns an UpSetR bar plot showing the number of intersecting sequences across multiple samples.

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.aa <- productiveSeq(file.list = file.list, aggregate = "aminoAcid")

commonSeqsBar(productive.aa = productive.aa, samples = c("TRB_CD4_949", "TRB_CD8_949",
"TRB_Unsorted_949", "TRB_Unsorted_1320"), color.sample = "TRB_CD8_949")
```

---

commonSeqsPlot *Common sequences plot*

Description

Creates a scatter plot of just the sequences in common between two samples.

Usage

```
commonSeqsPlot(sample1, sample2, productive.aa, show = "common")
```

Arguments

sample1	A name of a sample in a list of data frames generated by the LymphoSeq function productiveSeq.
sample2	A name of a sample in a list of data frames generated by the LymphoSeq function productiveSeq.
productive.aa	A list of data frames of productive amino acid sequences produced by the LymphoSeq function productiveSeq containing the samples to be compared.
show	A character vector specifying whether only the common sequences should be shown or all sequences. Available options are "common" or "all".

Details

The plot is made using the package ggplot2 and can be reformatted using ggplot2 functions. See examples below.

Value

Returns a frequency scatter plot of two samples showing only the shared sequences.

See Also

An excellent resource for examples on how to reformat a ggplot can be found in the R Graphics Cookbook online (http://www.cookbook-r.com/Graphs/).

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.aa <- productiveSeq(file.list = file.list, aggregate = "aminoAcid")

commonSeqsPlot("TRB_Unsorted_32", "TRB_Unsorted_83",
   productive.aa = productive.aa)

# Change the X and Y axises to log-10 scale
commonSeqsPlot("TRB_Unsorted_32", "TRB_Unsorted_83",
   productive.aa = productive.aa) +
   ggplot2::scale_x_log10() +
   ggplot2::scale_y_log10() +
   ggplot2::annotation_logticks(sides = "bl")
```

---

commonSeqsVenn *Common sequences Venn diagram*

Description

Creates a Venn diagram comparing the number of common sequences in two or three samples.

Usage

```
commonSeqsVenn(samples, productive.seqs)
```

Arguments

samples	A character vector of two or three names of samples in productive.seqs to compare.
productive.seqs	A list of productive amino acid sequences generated by the LymphoSeq function productiveSeq.

Value

Returns a a Venn diagram of the number of common sequences between two or three samples.

See Also

commonSeqs

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.aa <- productiveSeq(file.list = file.list, aggregate = "aminoAcid")

# Plot a triple Venn diagram
commonSeqsVenn(samples = c("TRB_Unsorted_0",
   "TRB_Unsorted_32", "TRB_Unsorted_83"),
   productive.seqs = productive.aa)

# Plot a double Venn diagram
commonSeqsVenn(samples = c("TRB_Unsorted_0",
   "TRB_Unsorted_32"), productive.seqs = productive.aa)

# Save Venn diagram as a .png file to working directory
png(filename = "Venn diagram.png", res = 300, units = "in", height = 5, width = 5)

commonSeqsVenn(samples = c("TRB_Unsorted_0", "TRB_Unsorted_32"),
   productive.seqs = productive.aa)

dev.off()
```

---

differentialAbundance *Differential abundance analysis*

Description

Use a Fisher exact test to calculate differential abdunance of each sequence in two samples and reports the log2 transformed fold change, P value and adjusted P value.

Usage

```
differentialAbundance(sample1, sample2, list,
  abundance = "estimatedNumberGenomes", type = "aminoAcid", q = 1,
  zero = 0.001, parallel = FALSE)
```

Arguments

sample1	A character vector indicating the name of the first sample in the list to be compared.
sample2	A character vector indicating the name of the second sample in the list to be compared.
list	A list of data frames consisting of antigen receptor sequences imported by the LymphoSeq function readImmunoSeq.
abundance	The input value for the Fisher exact test. "estimatedNumberGenomes" is recommend but "count" may also be used.

type	A character vector indicating whether "aminoAcid" or "nucleotide" sequences should be used. If "aminoAcid" is specified, then run productiveSeqs first.
q	A numeric value between 0.0 and 1.0 indicating the threshold Holms adjusted P value (also knowns as the false discovery rate or q value) to subset the results with. Any sequences with a q value greater than this value will not be shown.
zero	A numeric value to set all zero values to when calculating the log2 transformed fold change between samples 1 and 2. This does not apply to the p and q value calculations.
parallel	A boolean indicating wheter parallel processing should be used or not.

Value

Returns a data frame with columns corresponding to the frequency of the abudance measure in samples 1 and 2, the P value, Q value (Holms adjusted P value, also knowns as the false discovery rate), and log2 transformed fold change.

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.aa <- productiveSeq(file.list = file.list, aggregate = "aminoAcid")

differentialAbundance(list = productive.aa, sample1 = "TRB_Unsorted_949",
                      sample2 = "TRB_Unsorted_1320", type = "aminoAcid", q = 0.01,
                      zero = 0.001)
```

---

exportFasta *Export sequences in fasta format*

Description

Export nucleotide or amino acid sequences in fasta format.

Usage

```
exportFasta(list, type = "nucleotide", names = c("rank", "aminoAcid",
  "count"))
```

Arguments

list	A list of data frames consisting of antigen receptor sequences imported by the LymphoSeq function readImmunoSeq.
type	A character vector indicating whether "aminoAcid" or "nucleotide" sequences should be exported. If "aminoAcid" is specified, then run productiveSeqs first.
names	A character vector of one or more column names to name the sequences. If "rank" is specified, then the rank order of the sequences by frequency is used.

Value

Exports fasta files to the working directory.

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

exportFasta(list = file.list, type = "nucleotide", names = c("rank", "aminoAcid", "count"))

productive.aa <- productiveSeq(file.list = file.list, aggregate = "aminoAcid")

exportFasta(list = productive.aa, type = "aminoAcid", names = "frequencyCount")
```

`geneFreq` *Gene frequencies*

Description

Creates a data frame of VDJ gene counts and frequencies.

Usage

```
geneFreq(productive.nt, locus = "VDJ", family = FALSE)
```

Arguments

`productive.nt`	A list of one or more data frames of productive sequences generated by the LymphoSeq function productiveSeq where the parameter aggregate is set to "nucleotide".
`locus`	A character vector indicating which VDJ genes to include in the output. Available options include "VDJ", "DJ", "VJ", "DJ", "V", "D", or "J".
`family`	A Boolean value indicating whether or not family names instead of gene names are used. If TRUE, then family names are used and if FALSE, gene names are used.

Value

Returns a data frame with the sample names, VDJ gene name, count, and % frequency of the V, D, or J genes (each gene frequency should add to 100% for each sample).

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.nt <- productiveSeq(file.list = file.list, aggregate = "nucleotide")

geneFreq(productive.nt, locus = "VDJ", family = FALSE)

# Create a heat map of V gene usage
vfamilies <- geneFreq(productive.nt, locus = "V", family = TRUE)

require(reshape)
```

```
vfamilies <- reshape::cast(vfamilies, familyName ~ samples, value = "frequencyGene", sum)

rownames(vfamilies) <- as.character(vfamilies$familyName)

vfamilies$familyName <- NULL

RedBlue <- grDevices::colorRampPalette(rev(RColorBrewer::brewer.pal(11, "RdBu")))(256)

require(pheatmap)

pheatmap::pheatmap(vfamilies, color = RedBlue, scale = "row")

# Create a word cloud of V gene usage
vgenes <- geneFreq(productive.nt, locus = "V", family = FALSE)

require(wordcloud)

wordcloud::wordcloud(words = vgenes[vgenes$samples == "TRB_Unsorted_83", "geneName"],
   freq = vgenes[vgenes$samples == "TRB_Unsorted_83", "frequencyGene"],
  colors = RedBlue)

# Create a cumulative frequency bar plot of V gene usage
vgenes <- geneFreq(productive.nt, locus = "V", family = FALSE)

require(ggplot2)

ggplot2::ggplot(vgenes, aes(x = samples, y = frequencyGene, fill = geneName)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  scale_y_continuous(expand = c(0, 0)) +
  guides(fill = guide_legend(ncol = 3)) +
  labs(y = "Frequency (%)", x = "", fill = "") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```

`lorenzCurve` *Lorenz curve*

Description

Plots a Lorenz curve derived from the frequency of the amino acid sequences.

Usage

```
lorenzCurve(samples, list)
```

Arguments

`samples`	A character vector of sample names in list.
`list`	A list data frames generated using the LymphoSeq function readImmunoSeq or productiveSeq. "frequencyCount" is a required column.

Details

The Gini coefficient is an alternative metric used to calculate repertoire diversity and is derived from the Lorenz curve. The Lorenz curve is drawn such that x-axis represents the cumulative percentage of unique sequences and the y-axis represents the cumulative percentage of reads. A line passing through the origin with a slope of 1 reflects equal frequencies of all sequences. The Gini coefficient is the ratio of the area between the line of equality and the observed Lorenz curve over the total area under the line of equality.

The plot is made using the package ggplot2 and can be reformatted using ggplot2 functions. See examples below.

Value

Returns a Lorenz curve.

See Also

An excellent resource for examples on how to reformat a ggplot can be found in the R Graphics Cookbook online (http://www.cookbook-r.com/Graphs/).

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

lorenzCurve(samples = names(file.list), list = file.list)

productive.aa <- productiveSeq(file.list = file.list, aggregate = "aminoAcid")

lorenzCurve(samples = names(productive.aa), list = productive.aa)

# Change the legend labels, line colors, and add a title
samples <- c("TRB_Unsorted_0", "TRB_Unsorted_32",
   "TRB_Unsorted_83", "TRB_Unsorted_949", "TRB_Unsorted_1320")

lorenz.curve <- lorenzCurve(samples = samples, list = productive.aa)

labels <- c("Day 0", "Day 32", "Day 83", "Day 949", "Day 1320")

colors <- c("navyblue", "red", "darkgreen", "orange", "purple")

lorenz.curve + ggplot2::scale_color_manual(name = "Samples", breaks = samples,
   labels = labels, values = colors) + ggplot2::ggtitle("Figure Title")
```

---

mergeFiles *Merge files*

Description

Merges two or more sample data frames into a single data frame and aggregates count, frequency-Count, and estimatedNumberGenomes.

Usage

```
mergeFiles(samples, file.list)
```

Arguments

samples	A character vector of the names of the samples that are to be merged in file.list.
file.list	A list of data frames imported using the LymphoSeq function readImmunoSeq that contain the sample names that are to be merged. "aminoAcid", "nucleotide", "count" and "frequencyCount" are required columns.

Value

Returns a data frame of the merged samples. The values of count, frequencyCount, and estimatedNumberGenomes are aggregated. That is, the sum of count and estimatedNumberGenomes columns of the merged data frame should equal the sum of the columns from the unmerged samples. Likewise, the frequencyCount of the merged data frame should add up to 100%. All other columns in the unmerged data frames are included in the merge data frame.

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

TCRB_Day949_Merged <- mergeFiles(samples = c("TRB_CD4_949",
   "TRB_CD8_949"), file.list)

# To combine the merged data frames with file.list
file.list <- c(list("TCRB_Day949_Merged" = TCRB_Day949_Merged), file.list)
```

---

pairwisePlot *Pairwise comparison plot*

Description

Creates a heat map from a similarity or Bhattacharyya matrix.

Usage

```
pairwisePlot(matrix)
```

Arguments

matrix	A similarity or Bhattacharyya matrix produced by the LymphoSeq functions similarityMatrix or bhattacharyyaMatrix.

Details

The plot is made using the package ggplot2 and can be reformatted using ggplot2 functions. See examples below.

Value

A pairwise comparison heat map.

See Also

An excellent resource for examples on how to reformat a ggplot can be found in the R Graphics Cookbook online (http://www.cookbook-r.com/Graphs/). The functions to create the similarity or Bhattacharyya matrix can be found here: similarityMatrix and bhattacharyyaMatrix

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.aa <- productiveSeq(file.list = file.list, aggregate = "aminoAcid")

similarity.matrix <- similarityMatrix(productive.seqs = productive.aa)

pairwisePlot(matrix = similarity.matrix)

bhattacharyya.matrix <- bhattacharyyaMatrix(productive.seqs = productive.aa)

pairwisePlot(matrix = bhattacharyya.matrix)

# Change plot color, title legend, and add title
pairwisePlot(matrix = similarity.matrix) +
   ggplot2::scale_fill_gradient(low = "#deebf7", high = "#3182bd") +
   ggplot2::labs(fill = "Similarity score") + ggplot2::ggtitle("Figure Title")
```

---

phyloTree *Create phylogenetic tree*

Description

Create a phylogenetic tree using neighbor joining tree estimation for amino acid or nucleotide CDR3 sequences in a list of data frames.

Usage

```
phyloTree(list, sample, type = "nucleotide", layout = "rectangular",
  label = TRUE)
```

Arguments

list	A list data frames of unproductive nucleotide sequences or productive nucleotide sequences generated by the LymphoSeq function productiveSeq. vFamilyName, dFamilyName, jFamilyName, and count are required columns.
sample	A character vector indicating the name of the sample in the productive sequence list.
type	A character vector indicating whether "aminoAcid" or "nucleotide" sequences should be compared.

layout	A character vector indicating the tree layout. Options include "rectangular", "slanted", "fan", "circular", "radial" and "unrooted".
label	A Boolean indicating if the sequencing count should be shown next to the leaves.

Value

Returns a phylogenetic tree where each leaf represents a sequence color coded by the V, D, and J gene usage. The number next to each leaf refers to the sequence count. A triangle shaped leaf indicates the dominant sequence. Refer to the ggtree Bioconductor package documentation for details on how to manipulate the tree.

Examples

```
file.path <- system.file("extdata", "IGH_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.nt <- productiveSeq(file.list = file.list, aggregate = "nucleotide")

phyloTree(list = productive.nt, sample = "IGH_MVQ92552A_BL", type = "nucleotide",
         layout = "rectangular")

phyloTree(list = productive.nt, sample = "IGH_MVQ92552A_BL", type = "aminoAcid",
         layout = "circular")

# Add scale and title to figure
library(ggtree)
library(ggplot2)
phyloTree(list = productive.nt, sample = "IGH_MVQ92552A_BL", type = "aminoAcid",
         layout = "rectangular") +
         ggtree::theme_tree2() +
      ggplot2::theme(legend.position = "right", legend.key = element_rect(colour = "white")) +
         ggplot2::ggtitle("Title")

# Hide legend and leaf labels
phyloTree(list = productive.nt, sample = "IGH_MVQ92552A_BL", type = "nucleotide",
         layout = "rectangular", label = FALSE) +
         ggplot2::theme(legend.position="none")
```

---

productive *Productive sequences*

Description

Remove unproductive CDR3 sequences from a single data frame.

Usage

```
productive(sample, aggregate = "aminoAcid")
```

Arguments

sample	A data frame consisting of antigen receptor sequencing data. "aminoAcid", "count", and "frequencyCount" are required columns.
aggregate	Indicates whether the values of "count", "frequencyCount", and "esimatedNumberGenomes" should be aggregated by amino acid or nucleotide sequence. Acceptable values are "aminoAcid" or "nucleotide". If "aminoAcid" is selected, then the resulting data frame will have columns corresponding to "aminoAcid", "count", "frequnecyCount", and "estimatedNumberGenomes" (if this column is available). If "nucleotide" is selected then all columns in the original data frame will be present in the outputted data frame. The difference in output is due to the fact that the same amino acid CDR3 sequence may be encoded by multiple unique nucleotide sequences with differing V, D, and J genes.

Value

Returns a data frame of productive amino acid sequences with recomputed values for "count", "frequencyCount", and "esimatedNumberGenomes". A productive sequences is defined as a sequence that is in frame and does not have an early stop codon.

See Also

productiveSeq

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive <- productive(sample = file.list[["TRB_Unsorted_32"]], aggregate = "aminoAcid")
```

---

productiveSeq *Productive sequences*

Description

Remove unproductive CDR3 sequences from a list of data frames.

Usage

```
productiveSeq(file.list, aggregate = "aminoAcid", prevalence = FALSE)
```

Arguments

file.list	A list of data frames consisting antigen receptor sequencing data imported by the LymphoSeq function readImmunoSeq. "aminoAcid", "count", and "frequencyCount" are required columns.

aggregate — Indicates whether the values of "count", "frequencyCount", and "esimatedNumberGenomes" should be aggregated by amino acid or nucleotide sequence. Acceptable values are "aminoAcid" or "nucleotide". If "aminoAcid" is selected, then resulting data frame will have columns corresponding to aminoAcid, count, frequencyCount, and estimatedNumberGenomes (if this column is available). If "nucleotide" is selected then all columns in the original list will be present in the outputted list. The difference in output is due to the fact that the same amino acid CDR3 sequence may be encoded by multiple unique nucleotide sequences with differing V, D, and J genes.

prevalence — A Boolean value indicating if a new column should be added to each of the data frames giving the prevalence of each CDR3 amino acid sequence in 55 healthy donor peripheral blood samples. TRUE means the column is added and FALSE means it is not. Values range from 0 to 100% where 100% means the sequence appeared in the blood of all 55 individuals. The prevalenceTRB database is located in a separate package called LymphoSeqDB that should be loaded automatically.

Value

Returns a list of data frames of productive amino acid sequences with recomputed values for "count", "frequencyCount", and "esimatedNumberGenomes". A productive sequences is defined as a sequences that is in frame and does not have an early stop codon.

See Also

Refer to the LymphoSeqDB package for details regarding the prevalenceTRB database.

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.nt <- productiveSeq(file.list = file.list,
   aggregate = "nucleotide", prevalence = FALSE)

productive.aa <- productiveSeq(file.list = file.list,
  aggregate = "aminoAcid", prevalence = TRUE)
```

---

readImmunoSeq *Read ImmunoSeq files*

Description

Imports tab-separated value (.tsv) files exported by the Adaptive Biotechnologies ImmunoSEQ analyzer and stores them as a list of data frames.

Usage

```
readImmunoSeq(path, columns = c("aminoAcid", "nucleotide", "count",
  "count (templates)", "count (reads)", "count (templates/reads)",
  "frequencyCount", "frequencyCount (%)", "estimatedNumberGenomes",
  "vFamilyName", "dFamilyName", "jFamilyName", "vGeneName", "dGeneName",
  "jGeneName"), recursive = FALSE)
```

Arguments

path	Path to the directory containing tab-delimited files. Only files with the extension .tsv are imported. The names of the data frames are the same as names of the files.
columns	Column names from the tab-delimited files that you desire to import, all others will be disregarded. May use "all" to import all columns. A warning may be called if columns contain no data or must be coereced to a different class. Usually this warning can be ignored.
recursive	A Boolean value indicating whether tab-delimited files in subdirectories should be imported. If TRUE, then all files in the parent as well as the subdirectory are imported. If FALSE, then only files in the parent directory are imported.

Details

May import tab-delimited files containing antigen receptor sequencing from other sources (e.g. miTCR or miXCR) as long as the column names are the same as used by ImmunoSEQ files. Available column headings in ImmunoSEQ files are: "nucleotide", "aminoAcid", "count", "count (templates)", "count (reads)", "count (templates/reads)", "frequencyCount", "frequencyCount (%)", "cdr3Length", "vMaxResolved", "vFamilyName", "vGeneName", "vGeneAllele", "vFamilyTies", "vGeneNameTies", "vGeneAlleleTies", "dMaxResolved", "dFamilyName", "dGeneName", "dGeneAllele", "dFamilyTies", "dGeneNameTies", "dGeneAlleleTies", "jMaxResolved", "jFamilyName", "jGeneName", "jGeneAllele", "jFamilyTies", "jGeneNameTies", "jGeneAlleleTies", "vDeletion", "d5Deletion", "d3Deletion", "jDeletion", "n2Insertion", "n1Insertion", "vIndex", "n2Index", "dIndex", "n1Index", "jIndex", "estimatedNumberGenomes", "sequenceStatus", "cloneResolved", "vOrphon", "dOrphon", "jOrphon", "vFunction", "dFunction", "jFunction", "fractionNucleated".

IMPORTANT: be aware that Adaptive has changed the column names of their files over time and if the headings of your files are inconsistent, then specify column = "all" or include all variations of the headings you want to important. For example, column = c("count", "count (templates)", "count (reads)"). Also be aware that the "count" column previously reported the number of sequencing reads in earlier versions of ImmunoSEQ but now is equivalent to the "estimatedNumberGenomes" column.

Value

Returns a list of data frames. The names of each data frame are assigned according to the original ImmunoSEQ file names.

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path,
                           columns = c("aminoAcid", "nucleotide", "count",
                                       "count (templates)", "count (reads)",
                                       "count (templates/reads)",
                                       "frequencyCount", "frequencyCount (%)",
                                       "estimatedNumberGenomes"),
                           recursive = FALSE)
```

removeSeq *Remove sequence*

Description

Removes an amino acid sequence and associated data from all instances within a list of data frames and then recomputes the frequencyCount.

Usage

```
removeSeq(file.list, sequence)
```

Arguments

file.list	A list of data frames imported using the LymphoSeq function readImmunoSeq. "aminoAcid", "count", and "frequencyCount" are required columns.
sequence	A character vector of one or more amino acid sequences to remove from the list of data frames.

Value

Returns a list of data frames like the one imported except all rows with the specified amino acid sequence are removed. The frequencyCount is recalculated.

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

searchSeq(list = file.list, sequence = "CASSDLIGNGKLFF")

cleansed <- removeSeq(file.list = file.list, sequence = "CASSDLIGNGKLFF")

searchSeq(list = cleansed, sequence = "CASSDLIGNGKLFF")
```

searchPublished *Search for T cell receptor beta CDR3 amino acid sequences with known antigen specificity*

Description

Search for published T cell receptor beta CDR3 amino acid sequences with known antigen specificity in a list of data frames.

Usage

```
searchPublished(list)
```

Arguments

list	A list of data frames generated by the LymphoSeq functions readImmunoSeq or productiveSeq. "aminoAcid", "frequencyCount", and "count" are required columns.

Value

Returns a data frame of each sample name and instance in the sample that the published TCR sequence appeared along with additional information including antigen specificity, epitope, HLA type, and PubMed ID (PMID) for the reference where the sequence was characterized. The publishedTRB database is located in a separate package called LymphoSeqDB that should be loaded automatically.

See Also

Refer to the LymphoSeqDB package for details regarding the publishedTRB database.

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.aa <- productiveSeq(file.list = file.list, aggregate = "aminoAcid")

searchPublished(list = productive.aa)
```

---

searchSeq *Search for a sequence*

Description

Search for one or more amino acid or nucleotide CDR3 sequences in a list of data frames.

Usage

```
searchSeq(list, sequence, type = "aminoAcid", match = "global",
  editDistance = 0)
```

Arguments

list	A list of data frames generated by the LymphoSeq functions readImmunoSeq or productiveSeq. "aminoAcid" or "nucleotide", "frequencyCount", and "count" are required columns.
sequence	A character vector of one ore more amino acid or nucleotide CDR3 sequences to search.
type	A character vector specifying the type of sequence(s) to be searched. Available options are "aminoAcid" or "nucleotide".
match	A character vector specifying whether an exact partial or exact global match of the searched sequence(s) is desired. Available options are "partial" and "global".
editDistance	An integer giving the minimum edit distance that the sequence must be less than or equal to. See details below.

Details

An exact partial match means the searched sequence is contained within target sequence. An exact global match means the searched sequence is identical to the target sequence.

Edit distance is a way of quantifying how dissimilar two sequences are to one another by counting the minimum number of operations required to transform one sequence into the other. For example, an edit distance of 0 means the sequences are identical and an edit distance of 1 indicates that the sequences different by a single amino acid or nucleotide.

Value

Returns the rows for every instance in the list of data frames where the searched sequence(s) appeared.

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

aa1 <- "CASSPVSNEQFF"

aa2 <- "CASSQEVPPYQAFF"

searchSeq(list = file.list, sequence = aa1, type = "aminoAcid",
   match = "global", editDistance = 0)

searchSeq(list = file.list, sequence = c(aa1, aa2),
   type = "aminoAcid", match = "global", editDistance = 0)

searchSeq(list = file.list, sequence = aa1, type = "aminoAcid", editDistance = 1)

nt <- "CTGATTCTGGAGTCCGCCAGCACCAACCAGACATCTATGTACCTCTGTGCCAGCAGTCCGGTAAGCAATGAGCAGTTCTTCGGGCCA"

searchSeq(list = file.list, sequence = nt, type = "nucleotide", editDistance = 3)

searchSeq(list = file.list, sequence = "CASSPVS", type = "aminoAcid",
   match = "partial", editDistance = 0)

searchSeq(list = file.list, sequence = nt, type = "nucleotide", editDistance = 0)
```

---

seqMatrix *Sequence matrix*

Description

Creates a data frame with unique, productive amino acid sequences as rows and sample names as headers. Each value in the data frame represents the frequency that the sequence appeared in the sample.

Usage

```
seqMatrix(productive.aa, sequences)
```

Arguments

productive.aa	A list data frames of of productive amino acid sequences generated by LymphoSeq function productiveSeq where the aggregate parameter was set to "aminoAcid".
sequences	A character vector of amino acid sequences of interest. It is useful to specify the output from the LymphoSeq functions uniqueSeqs or topSeqs and subsetting the "aminoAcid" column. See examples below.

Value

Returns a data frame of unique, productive amino acid sequences as rows and the % frequency it appears in each sample as columns.

See Also

topSeqs and uniqueSeqs

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.aa <- productiveSeq(file.list = file.list, aggregate = "aminoAcid")

top.seqs <- topSeqs(productive.seqs = productive.aa, top = 0.1)

sequence.matrix <- seqMatrix(productive.aa = productive.aa,
   sequences = top.seqs$aminoAcid)

unique.seqs <- uniqueSeqs(productive.aa = productive.aa)

sequence.matrix <- seqMatrix(productive.aa = productive.aa,
   sequences = unique.seqs$aminoAcid)

# It can be helpful to combine top.freq and sequence.matrix
top.freq <- topFreq(productive.aa = productive.aa, percent = 0)

sequence.matrix <- seqMatrix(productive.aa = productive.aa, sequences = top.freq$aminoAcid)

top.freq.matrix <- merge(top.freq, sequence.matrix)
```

---

similarityMatrix *Similarity score matrix*

Description

Calculates the similarity score of all pairwise comparison from a list of data frames.

Usage

```
similarityMatrix(productive.seqs)
```

Arguments

productive.seqs
: A list data frames of productive sequences generated by the LymphoSeq function productiveSeq. "count" and "aminoAcid" are a required columns.

Value

A data frame of similarity scores calculated from all pairwise comparisons. The similarity scores is a measure of the amount of overlap between two samples. The value ranges from 0 to 1 where 1 indicates the sequence frequencies are identical in the two samples and 0 indicates no shared frequencies.

See Also

pairwisePlot for plotting results as a heat map.

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.aa <- productiveSeq(file.list = file.list, aggregate = "aminoAcid")

similarityMatrix(productive.seqs = productive.aa)
```

---

similarityScore *Similarity score*

Description

Calculates the similarity score of two samples.

Usage

```
similarityScore(sample1, sample2)
```

Arguments

sample1
: A data frame consisting of frequencies of antigen receptor sequences. "aminoAcid" and "count" are a required columns.

sample2
: A data frame consisting of frequencies of antigen receptor sequences. "aminoAcid" and "count" are a required columns.

Value

Returns the similarity score, a measure of the amount of overlap between two samples. The value ranges from 0 to 1 where 1 indicates the sequence frequencies are identical in the two samples and 0 indicates no shared frequencies.

See Also

similarityMatrix

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.aa <- productiveSeq(file.list, aggregate = "aminoAcid")

similarityScore(productive.aa[["TRB_Unsorted_32"]], productive.aa[["TRB_Unsorted_83"]])
```

`topFreq` *Top frequencies*

Description

Creates a data frame of the top productive amino acid sequences that have a specified minimum frequency threshold and reports the number of samples that the sequence appears in along with the minimum, maximum, and mean frequency across all samples. For T cell receptor beta sequences, the % prevalence and antigen specificity of that sequence is also provided.

Usage

```
topFreq(productive.aa, percent = 0.1)
```

Arguments

`productive.aa`	A list data frames of of productive amino acid sequences imported using the function LymphoSeq function productiveSeq where the aggregate parameter was set to "aminoAcid".
`percent`	The minimum % frequency that the sequence appears in any of the listed samples.

Value

A data frame of amino acid sequences and the number of samples that the sequence appears in along with the minimum, maximum, and mean frequency across all samples. For T cell receptor beta sequences, additionally reported is the % prevalence that the sequence appears in 55 healthy donor blood samples. Also provided is the antigen specificity of that sequence if known by comparing it to a database of previously reported sequences in the literature. The prevalenceTRB and publishedTRB databases are located in a separate package called LymphoSeqDB that should be loaded automatically.

See Also

Refer to the LymphoSeqDB package for details regarding the prevalenceTRB and publishedTRB database.

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.aa <- productiveSeq(file.list = file.list, aggregate = "aminoAcid")

top.freq <- topFreq(productive.aa = productive.aa, percent = 0.1)
```

---

topSeqs *Top sequences*

Description

Creates a data frame of a selected number of top productive sequences from a list of data frames.

Usage

```
topSeqs(productive.seqs, top = 1)
```

Arguments

productive.seqs	A list data frames of productive sequences generated by the LymphoSeq function productiveSeq. "frequencyCount" and "aminoAcid" are a required columns.
top	The number of top productive sequences in each data frame to subset by their frequencies.

Value

Returns a data frame of a selected number of top productive sequences from a list of data frames.

See Also

chordDiagramVDJ

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.aa <- productiveSeq(file.list = file.list, aggregate = "aminoAcid")

top.seqs <- topSeqs(productive.seqs = productive.aa, top = 1)
```

topSeqsPlot *Cumulative frequency bar plot of top sequences*

Description

Create a cumulative frequency bar plot of a specified number of top sequences.

Usage

```
topSeqsPlot(list, top = 10)
```

Arguments

list	A list data frames imported using the LymphoSeq function readImmunoSeq or productiveSeq.
top	The number of top sequences to be colored in the bar plot. All other, less frequent sequences are colored violet.

Details

The plot is made using the package ggplot2 and can be reformatted using ggplot2 functions. See examples below.

Value

Returns a cumulative frequency bar plot of the top sequences.

See Also

An excellent resource for examples on how to reformat a ggplot can be found in the R Graphics Cookbook online (http://www.cookbook-r.com/Graphs/).

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

topSeqsPlot(list = file.list, top = 10)

# Display the number of sequences at the top of bar plot and add a title
n <- as.character(lapply(file.list, nrow))

topSeqsPlot(list = file.list, top = 10) +
  ggplot2::annotate("text", x = 1:length(file.list), y = 105, label = n, color = "black") +
   ggplot2::expand_limits(y = c(0, 110)) + ggplot2::ggtitle("Figure Title") +
   ggplot2::scale_x_discrete(limits = names(file.list))
```

uniqueSeqs *Unique sequences*

Description

Aggregates all productive sequences within a list of data frames by count.

Usage

```
uniqueSeqs(productive.aa)
```

Arguments

productive.aa	A list data frames of of productive amino acid sequences imported using the function LymphoSeq function productiveSeq where the aggregate parameter was set to "aminoAcid".

Value

A data frame of unique amino acid sequences from the list of data frames aggregated by count.

Examples

```
file.path <- system.file("extdata", "TCRB_sequencing", package = "LymphoSeq")

file.list <- readImmunoSeq(path = file.path)

productive.aa <- productiveSeq(file.list = file.list, aggregate = "aminoAcid")

unique.seqs <- uniqueSeqs(productive.aa = productive.aa)
```

Index