Package 'RBiotools'

March 18, 2022

Type Package

Title Support for Basic Comparative Microbial Genomics

Version 0.5.3

Date 2022-03-18

Author Michael R. Leuze

Maintainer Michael R. Leuze <RBiotools@gmail.com>

Description Supports microbial comparative genomics with functions parameterized by a list of GenBank accession numbers and R implementations of Prodigal, RNAmmer, and Linclust.

License GPL (>= 2) | Artistic-2.0

Imports ape, Biostrings, data.table, fmsb, ggplot2, gplots, graphics, grid, grImport, gridExtra, methods, msa, pheatmap, Rcpp, RCurl, rentrez, seqinr, stats, stringr, utils

LinkingTo Rcpp

NeedsCompilation yes

LazyData true

RoxygenNote 7.1.1

Roxygen list(markdown = TRUE)

Encoding UTF-8

R topics documented:

---

addGenome *addGenome*

Description

Adds a user-supplied genome to the set of genomes in an RBiotools session

Usage

```
addGenome()
```

Details

The primary use of the addGenome function is to add a genome nucleotide sequence in FASTA format to the GenomeSeqList in the RBiotools global data infrastructure. This is necessary when a genome of interest is not available from GenBank. A typical application of addGenome is to incorporate a *newly sequenced genome* into RBiotools so that it may be included in a set of known genomes for comparative genomics analyses.

The user is interactively queried for

- **Unique Identifier** – A unique identifier that serves as an *accession number* for the genome, such as: XX000001 — Note that "XX" is not currently used as an accession prefix by GenBank, EMBL, or DDBJ.
- **FASTA file Name** – The name of a file in FASTA format *in the current working directory* that contains one or more nucleotide sequences, such as: XX000001.fasta
- **General Identifier** – An identifier that corresponds to a *species*-level identifier, such as: Clinical Isolates from Sample ABC
- **Specific Identifier** – An identifier that corresponds to a *strain*-level identifier, such as: ABC.01

Side Effects of addGenome:

- Converts the sequence(s) in the FASTA file into a DNAStringSet
- Appends the DNAStringSet to the global list of genome sequences GenomeSeqList
- Adds sequence identifiers to the global data frame orgName.df
- Adds sequence identifiers, length, and AT content to the global data frame orgData.df

Value

None

Development Notes

Questions

- Should the user be prompted for identifiers and FASTA file name (as currently implemented), or should these be supplied as function parameters?
- Would an additional third-level identifier be useful to clinical and experimental biologists?

Examples

```
## Not run:

# Requires a nucleotide FASTA file (such as XX000001.fasta) in the current working directory
# Possible user input is shown after each prompt.

addGenome()
Enter the unique identifier you will use to reference this genome: XX000001
Enter the FASTA file name for genomic sequence(s): XX000001.fasta
Enter a general identifier (e.g., species) for these genome sequence(s): Clinical Isolates from Sample ABC
Enter a specific identifier (e.g., strain) for these genome sequence(s): ABC.01

## End(Not run)
```

`checkIdentifier` *checkIdentifier*

Description

Makes the use of Whole Genome Shotgun (WGS) sequencing project identifers (such as Prefix, Project, Caption, Locus, GenBank_assembly, and RefSeq_assembly) seamlessly interchangable when calling RBiotools functions.

Usage

```
checkIdentifier(givenID)
```

Arguments

`givenID`	character WGS sequencing project identifier

Details

For WGS identifiers, `checkIdentifier` examines the contents of the `wgsName.df` table to determine whether the WGS project is known to the current R session. If not, `checkIdentifier` uses `rentrez` functions that work with the NCBI Eutils API to search and download summary data for WGS projects from the GenBank Nucleotide and Assembly databases. These data are then parsed and the WGS identifers extracted are added as a new entry in `wgsName.df`. The "Locus" identifier is returned to the calling function, since it is the identifier that is used, when sequence integers are incorporated, to download contigs from a WGS project.

Note: It is likely that there are some situations in which `checkIdentifier` will fail. Prior to 2019, a WGS contig was identified by a 4-letter alphabetic prefix and a two-digit version for the WGS and then a six-digit sequence for the contig. From NCBI documentation, *"In January 2019 GenBank began assigning accessions with a stable 6-letter WGS accession prefix and a* **minimum** *of 9 digits, e.g., XXXXXX000000000, XXXXXX010000000, and XXXXXX010000001, for the wgs project, its first version, and its first sequence, respectively."* The current implementation of `checkIdentifier` assumes that the complete accession will be the 6-letter prefix followed by exactly 9 digits, 2 for version and 7 for sequence.

Note: In general, `checkIdentifier` finds the identifiers for the *latest* version of a WGS sequencing project. It is possible, however, due to inconsistencies in the GenBank databases, that unexpected behavior will result, as in the example below, in which, there is a discrepancy in the identifiers extracted from GenBank. The "1" in WGS_Project and WGS_Locus indicates that the latest version of the WGS project is version 1, but the ".2" in GenBank_Assembly and RefSeq_Assembly indicates that the latest version of the WGS project is version 2. A search of the GenBank Assembly database for the term "GCF_000446905" finds two versions of a WGS sequencing project, an earlier version with project ID AIBS01 and 121 contigs and a later version, also with project ID AIBS01, and 120 contigs. A further search of the GenBank Nucleotide database finds contigs with identifiers AIBS01000001 - AIBS01000121, but no contigs with identifiers beginning with AIBS02. Calling the function `downloadGenBank` with the WGS_Locus "AIBS01000000" will return 121 contigs.

Value

character WGS sequencing project identifier

Examples

```
## Not run:
gbID <- "GCF_000446905"            # A RefSeq assembly identifier for a WGS project
wgsID <- checkIdentifier(gbID)     # Returns the WGS_Locus identifier "AIBS01000000" ...
   # ... and also writes the following into the wgsName.df data frame:
   #   WGS_Prefix         AIBS
   #   WGS_Project        AIBS01
   #   WGS_Caption        AIBS00000000
   #   WGS_Locus          AIBS01000000
   #   GenBank_Assembly   GCA_000446905.2
   #   RefSeq_Assembly    GCF_000446905.2
   #   contigs            120

## End(Not run)
```

`createAtlas` *createAtlas*

Description

Creates an SVG file in the working directory containing a Genome Atlas figure for a specified organism

Usage

```
createAtlas(organismAccession)
```

Arguments

organismAccession

Details

createAtlas creates a Genome Atlas for an organism specified by a GenBank accession number. Required data include the genome sequence, protein gene coordinates and strand, and rRNA gene coordinates and strand. If these data are not found in the RBiotools global data structures, they will automatically be downloaded or derived.

Value

None

Lanes currently implemented

- Genomic sequence coordinate scale
- Percent AT
- GC skew
- Annotations:
  - Protein coding sequences on forward and reverse strands
  - Ribosomal RNA sequences on forward and reverse strands
- Position preference
- Stacking energy

Examples

```
## Not run:
createAtlas("CP001859") # Acidaminococcus fermentans DSM 20731

## End(Not run)
```

---

dendrogram16S *dendrogram16S*

Description

A function for constructing and plotting a dendrogram using ribosomal RNA 16S sequences

Usage

```
dendrogram16S(
  accessionList,
  treeType = "BEST",
  leafLabel = "shortName",
  plotIdentifier = NULL
)
```

Arguments

accessionList	character vector containing GenBank accession numbers
treeType	character vector with values "FULL" or "BEST" indicating whether the full set of ribosomal RNA 16S sequences will be use to construct a dendrogram, or only the highest scoring 16S sequence from each organism.
leafLabel	character vector indicating how the leaves of the dendrogram are to be labeled. Options are "fullName" or "shortName". "fullName" is recommended for a diverse, multi-species set of genomes. "shortName" is recommended for a single-species set of genomes.
plotIdentifier	character vector containing a title for the plot

Details

This function plots a dendrogram based on ribosomal RNA 16S sequences.

The function dendrogram16S

1. extracts the specified 16S sequences from the RBiotools global storage,
2. aligns the sequences using the msa – MultiSequence Alignment – function from the msa R package,
3. computes the pairwise distances between sequences using the "raw" evolutionary model – the proportion or the number of sites that differ between each pair of sequences – of the dist.dna function from the ape R package, and
4. constructs a dendrogram using the bionj function, which performs the BIONJ algorithm of Gascuel, from the ape package. Note that clustering using the Neighbor Joining algorithm results in an *unrooted* tree.
5. It then plots the dendrogram using plot from the R Graphics Package.

Possible future enhancements:

- Scale the size of labels to accomodate for the size of the dendrogram
- Specify the maximum number of 16S sequences from a single genome to be used in constructing the dendrogram, allowing gradations between "BEST" and "FULL"
- Construct a representative 16S sequence for each organism from its set of 16S sequences, and use representative sequences to construct the dendrogram
- Use more sophisticated methods for constructing trees, such as those found in the *Phylogenetic Inference* packages from the CRAN *Phylogenetics* project

Value

None

References

O. Gascuel (1997) *BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data* https://doi.org/10.1093/oxfordjournals.molbev.a025808

Examples

```
## Not run:
myGenomes <- c("CP018228", "CP017405", "HG530068", "CP002031", "CP002332") # Proteobacteria
dendrogram16S(myGenomes)

## End(Not run)
```

---

downloadGenBank *downloadGenBank*

Description

Downloads genomic information for a set of genomes into an RBiotools session

Usage

```
downloadGenBank(accessionList)
```

Arguments

accessionList	character vector containing GenBank accession numbers

Details

This is a convenience function that determines whether each accession number in `accessionList` is associated with an organism or with a Whole Genome Shotgun (WGS) project and then calls either `downloadGenome` or `downloadWGS`. It uses the `rentrez` function `entrez_summary`, which works with the NCBI Eutils API to search the GenBank nucleotide database, to determine the type of download. It catches errors – both from the user, such as invalid accession numbers, and from GenBank, such as violations of access policies (not expected) – and skips any accession number that causes an error. `downloadGenBank` currently recognizes GenBank IDs for finished genomes and GenBank IDs, GenBank WGS assembly IDs, and RefSeq WGS assembly IDs for Whole Genome Shotgun projects.

Note: `downloadGenBank` *will seldom, if ever, need to be called directly by the* `RBiotools` *user.* `RBiotools` *functions will automatically initiate downloads from GenBank, if necessary.*

Value

None

Examples

```
## Not run:
accessionIDs <- c(
  "REQD01000000",  # Listeria monocytogenes strain PNUSAL004150 - a WGS project
  "CP015831"       # Escherichia coli O157 strain 644-PT8       - a finished genome
)

downloadGenBank(accessionIDs)

## End(Not run)
```

downloadGenome *downloadGenome*

Description

Downloads genomic information for a set of genomes into an RBiotools session

Usage

```
downloadGenome(accessionList)
```

Arguments

accessionList	character vector containing GenBank accession numbers

Details

This function fetches information for a set of genomes specified by accession numbers. It uses rentrez functions that work with the NCBI Eutils API to search and download data from the GenBank nucleotide database. It catches errors – both from the user, such as invalid accession numbers, and from GenBank, such as violations of access policies (not expected) – and skips any accession number that causes an error. For each accession number, entrez_fetch downloads a FASTA file of the genome sequence and entrez_summary downloads genomic information, including identifiers, taxonomy, and counts of various types of genes – protein encoding, pseudo, and RNA.

Side effects of downloadGenBank:

- Converts the sequence(s) in the FASTA file into a DNAStringSet
- Appends the DNAStringSet to the global list of genome sequences GenomeSeqList
- Adds sequence identifiers and taxonomy information, **when avaiable via** Entrez, to the global data frame orgName.df
- Adds sequence identifiers, length, AT content, and gene counts to the global data frame orgData.df

Note: The gene counts in orgData.df are those downloaded from GenBank by entrez_summary. These counts may differ from the number of genes called by Prodigal or RNAmmer.

Identifiers include organism, strain, and substrain names, if available. Additional identifiers are constructed, primarily to give the user options in labeling plots. These additional identifiers include "fullName", which contains organism, strain, and substrain identifers. The "fullName" identifier is necessary because of inconsistencies in organism naming in GenBank. In particular, organism names may include or omit strain and substrain identifiers. A short identifier that is useful for labeling trees for sets of organisms from the same species is the "shortName" identifier, which includes only strain and substrain identifiers.

Note: downloadGenome *will seldom, if ever, need to be called directly by the* RBiotools *user.* RBiotools *functions will automatically initiate downloads from GenBank, if necessary.*

Future enhancements: Construct a download function for genomes with multiple contigs. These genomes include genomes with multiple chromosomes, such as *Burkholderia mallei ATCC 23344* with chromosome accession numbers CP000010 and CP000011.

Value

None

Examples

```
## Not run:
E_coli <- c(
  "AP012306",  # Escherichia coli str. K-12 substr. MDS42 DNA
  "KK583188",  # Escherichia coli DSM 30083 = JCM 1649 = ATCC 11775
  "U00096",    # Escherichia coli str. K-12 substr. MG1655
  "CP000802",  # Escherichia coli HS
  "CP000800",  # Escherichia coli E24377A
  "AP009378",  # Escherichia coli SE15
  "FM180568",  # Escherichia coli 0127:H6 E2348/69
  "CU928163",  # Escherichia coli UMN026
  "CP008957",  # Escherichia coli O157:H7 str. EDL933
  "CP027027",  # Shigella dysenteriae strain E670/74
  "CP026802",  # Shigella sonnei strain ATCC 29930
  "CP026877",  # Shigella boydii strain ATCC 35964
  "CP026793",  # Shigella flexneri strain 74-1170
  "CP015831"   # Escherichia coli O157 strain 644-PT8
)

downloadGenome(E_coli)

## End(Not run)
```

`downloadWGS` *downloadWGS*

Description

Downloads genomic information for a set of Whole Genome Shotgun sequencing projects into an RBiotools session

Usage

```
downloadWGS(accessionList)
```

Arguments

`accessionList`	character vector containing GenBank accession numbers

Details

This function fetches information for a set of WGS (Whole Genome Shotgun) sequencing projects specified by accession numbers. If an accession number corresponds to a contig in a WGS sequencing project, it is **replaced with the accession number for the entire project**, and all project sequences (scaffolds or contigs) are downloaded. `downloadWGS` uses `rentrez` functions that work with the NCBI Eutils API to search and download data from the GenBank nucleotide database. It catches errors – both from the user, such as invalid accession numbers or accession numbers that do not correspond to WGS projects, and from GenBank, such as violations of access policies (not expected) – and skips any accession number that causes an error. For each WGS contig accession number, `entrez_fetch` downloads a FASTA file of the contig sequence and `entrez_summary` downloads genomic information, including identifiers, taxonomy, and counts of various types of

genes – protein encoding, pseudo, and RNA – which are summed for entry into the orgData.df data frame.

Note: The gene counts in orgData.df are those downloaded from GenBank by entrez_summary. These counts may differ from the number of genes called by Prodigal or RNAmmer.

Note: downloadWGS *will seldom, if ever, need to be called directly by the* RBiotools *user.* RBiotools *functions will automatically initiate downloads from GenBank, if necessary.*

Value

None

Examples

```
## Not run:
accessionIDs <- c(
  "REQD01000000",  # Listeria monocytogenes strain PNUSAL004150  (9 sequences)
  "PJKH01000000"   # Akkermansia muciniphila strain GP11  (42 sequences)
)

downloadWGS(accessionIDs)

## End(Not run)
```

---

fetchProdigalCalls *fetchProdigalCalls*

Description

Extracts Prodigal gene calls from the RBiotools global data store

Usage

```
fetchProdigalCalls(accessionList)
```

Arguments

accessionList	character vector containing GenBank accession numbers – only the first vector element is used

Details

This function fetches Prodigal gene calls for a genome specified by an accession number. If the accessionList character vector parameter contains multiple accession numbers, only the first is used. It fetches the Prodigal gene calls by subsetting the global data frame ProdigalCalls. Entries where the value in ProdigalCalls$accession matches the specified accession number are selected, and the results are returned in a data frame. If the genome specified by the accession number has not been downloaded or the protein genes have not been called, RBiotools will perform these tasks automatically.

Note: fetchProdigalCalls *will seldom, if ever, need to be called directly by the* RBiotools *user. It is included in* RBiotools *primarily as a tool for future development of proteomics functionality.*

Value

genes.df

Examples

```
## Not run:
geneCalls <- fetchProdigalCalls("U00096")

## End(Not run)
```

`fetchRNAmmerCalls` *fetchRNAmmerCalls*

Description

Extracts RNAmmer gene calls from the RBiotools global data store

Usage

```
fetchRNAmmerCalls(accessionList)
```

Arguments

`accessionList`	character vector containing GenBank accession numbers – only the first vector element is used

Details

This function fetches RNAmmer gene calls for a genome specified by an accession number. If the `accessionList` character vector parameter contains multiple accession numbers, only the first is used. It fetches the RNAmmer gene calls by subsetting the global data frame `RNAmmerCalls`. Entries where the value in `RNAmmerCalls$accession` matches the specified accession number are selected, and the results are returned in a data frame. If the genome specified by the accession number has not been downloaded or if all types of rRNA genes have not been called, `RBiotools` will perform these tasks automatically.

Note: `fetchRNAmmerCalls` *will seldom, if ever, need to be called directly by the* `RBiotools` *user. It is included in* `RBiotools` *primarily as a tool for future development of functionality.*

Value

genes.df

Examples

```
## Not run:
geneCalls <- fetchRNAmmerCalls("U00096")

## End(Not run)
```

findAccessionIDs *findAccessionIDs*

Description

Searches for complete chromosome GenBank sequence assession numbers

Usage

```
findAccessionIDs(accessionID, includeStrain = FALSE, maxReturn = 50)
```

Arguments

accessionID	character GenBank accession number
includeStrain	logical indicates whether strain is to be included in the search, default is FALSE

Details

findAccessionIDs is potentially useful for constructing sets of genomes for analysis.

findAccessionIDs searches for complete GenBank chromosomal sequences associated with a specified GenBank accession number. It uses rentrez functions that work with the NCBI Eutils API to search and download summary data from the GenBank nucleotide database.

Note: There are some situations in which findAccessionIDs will fail, in particular, when the name of the organism is **not** included in the title of the GenBank record. An example of this is accession number CP000800, for which the organism name is *Escherichia coli O139:H28 str. E24377A* and the title of the GenBank record is *Escherichia coli E24377A, complete genome*. A more subtle mismatch occurs for accession number FM180568, where the organism name contains *str.* preceding the strain, but the title of the GenBank record does not. A more sophisticated parsing of organism names and record titles may improve the performance of this function, but it is unlikely that all inconsistencies will be identified and resolved.

Value

None

Examples

```
## Not run:
accessionID <- "NZ_JVDK01001145"  # Burkholderia cenocepacia strain 530_BMUL (WGS)
findAccessionIDs(accessionID)     # This call returns (as of May 2019)
                                  # 57 IDs for complete chromosomal sequences from:
                                  # * 20 Burkholderia cenocepacia strains and
                                  # *  4 complete phage sequences

## End(Not run)
```

initializeRBiotools *initializeRBiotools*

Description

Initialize the global data store for an RBiotools session

Usage

```
initializeRBiotools()
```

Arguments

None

Details

This function initializes the global data structures that collectively constitute the global data store for an RBiotools session. This global data store makes the writing and reading of temporary files unnecessary.

RBiotools global data structures:

- GenomeSeqList – an R list of named DNAStringSets, each of which is a genomic sequence identified by an accession number
- orgName.df – an R data frame containing genome identifiers
- orgData.df – an R data frame containing genome data, including sequence length, AT content, and gene counts
- wgsName.df – an R data frame containing alternative names for WGS projects
- ProdigalCalls – an R data frame containing Prodigal called genes, locations, scores, and nucleotide and protein sequences for a set of genomes
- RNAmmerCalls – an R data frame containing RNAmmer called genes, locations, scores, and nucleotide sequences for a set of genomes
- rRNAcalled.df – an R data frame that tracks which types of rRNA genes have been called by RNAmmer for a set of genomes

initializeRBiotools also downloads and installs all RBiotools dependent packages and loads them into the current R session.

Note: initializeRBiotools *will seldom, if ever, need to be called directly by the* RBiotools *user.* RBiotools *functions will automatically initialize the global data store, if necessary.*

Value

None

Examples

```
## Not run:
initializeRBiotools()

## End(Not run)
```

plotBlastMatrix *plotBlastMatrix*

Description

Creates an SVG file in the working directory for a BLAST matrix for a set of genomes

Usage

```
plotBlastMatrix(proteinGrouping, organismIdentifier = "shortName", digits = 0)
```

Arguments

proteinGrouping	a homology matrix data frame constructed by runLinclust containing protein grouping information
organismIdentifier	character vector indicating the name to be used for organisms; options are "accessionNumber", "fullName" or "shortName"; "fullName" is recommended for a diverse, multi-species set of genomes; "shortName" is recommended for a single-species set of genomes.
digits	integer indicating the number of decimal digits to be displayed in homology percentage; values 0, 1, and 2 are supported.

Details

plotBlastMatrix creates a SVG file containing a figure that displays, for a set of genomes, homology between pairs of genomes and homology within genomes. plotBlastMatrix uses a "protein grouping" constructed by runLinclust. The protein grouping is returned from runLinclust as a homology matrix, with a column for each genome and a row for each protein group. Each homology matrix entry is a count of proteins for a genome / group combination.

Value

None

Examples

```
## Not run:
myGenomes <- c("AP012306", "KK583188", "U00096", "CP000802", "CP000800") # E. coli genomes
proteinGrouping <- runLinclust(myGenomes)
plotBlastMatrix(proteinGrouping)

## End(Not run)
```

plotHeatMapAA *plotHeatMapAA*

Description

A function for plotting heatmaps for amino acid usage across a set of genomes

Usage

```
plotHeatMapAA(
  accessionList,
  organismIdentifier = "shortName",
  plotIdentifier = NULL,
  colorPalette = "cm",
  plotMethod = "heatmap.2"
)
```

Arguments

accessionList	character vector containing GenBank accession numbers
organismIdentifier	character vector indicating the name to be used for organisms; options are "fullName" or "shortName"; "fullName" is recommended for a diverse, multi-species set of genomes; "shortName" is recommended for a single-species set of genomes
plotIdentifier	character vector containing a title for the plot
colorPalette	character vector indicating the color palette to be used in the heatmap; options are the color palettes from the grDevices package: "cm", "topo", "terrain", "heat", "rainbow"
plotMethod	character vector indicating the R function to use for plotting the heatmap; options are "heatmap" from the stats package, "heatmap.2" from the gplots package, and "pheatmap" from the "pheatmap" package. This parameter is included during the development process. (See **Development Notes** below.) When the best function for amino acid usage heatmaps is determined, this parameter will likely be removed.

Details

A heatmap is a graphical representation of data in a matrix that uses color-coding to represent different data values. plotHeatMapAA plots heatmaps for amino acid usage across a set of genomes (one or more) using a variety of methods from several R packages. A PNG file is created in the working directory.

Development Notes

General questions

- Which will be simpler for clinical / experimental biologists to use: 1) Splitting tasks into different functions – separate heatmap functions for amino acids and for codons, for example, or 2) combining related tasks into a single function – plotUsage, for example, which plots usage for amino acids or codons or plots position bias using either a radar plot or a circular bar plot?

- Should users provide names for the .PNG plot files? Should the heatmap plotting functions add sequence numbers to the plot file names to distinguish them and, in doing so, not overwrite earlier plot file? What are the tradeoffs?

Heatmaps can be challenging to configure. Some are based on the `ggplot2` R package by Hadley Wickham, a system for declaratively creating graphics, based on *The Grammar of Graphics* by Leland Wilkinson. `ggplot2` is documented in *ggplot2: Elegant Graphics for Data Analysis*, a 257-page Springer publication in the *Use R!* series. Complications arise when a heatmap function built on `ggplot2` alters access to the `ggplot2` controls, and, in some cases, renames `ggplot2` functions. This results in situations where modifications to heatmaps must be made by either resetting the heatmap function parameters or `ggplot2` parameters – sometimes either will work, sometimes neither. A rather complicated case in point: `ggplot2` rotates text labels by specifying `angle`. `angle` does not work with `pheatmap`, which uses `rot` in the `draw_columns` function, which is not directly accessible to the user. The `draw_columns` function, therefore, must be overwritten in the `pheatmap` namespace to rotate column labels. A couple of auxiliary functions have been added to `plotHeatMapAA` to overwrite the `pheatmap` namespace so that amino acid codes, which label the columns, will be displayed horizontally rather than vertically. So the bottom line is ... there is a lot of fine tuning of the heatmaps that can be done, but it's complicated. Development of heatmap codes for `RBiotools` is currently at a level where return for effort is significantly diminished.

Heatmaps are displayed in a .PNG format in a window external to R Studio, rather than in the R Studio plot pane. This approach allows precise control over the size of the plot, which can vary significantly if visualized in the R Studio plot pane. `plotHeatMapAA` writes a file named `HeatMapAA.png` in the current working directory.

Heatmap functions currently implemented are

- `heatmap` from the `stats` package
  - A square plot. The aspect ratio can be changed with `asp`, but dendrograms do not resize and are not, therefore, aligned with the heatmap.
  - The title (specified with `main`) overlaps the top margin of the plot unless prepended with a newline (`"\n"`) to lower it into the display. The title appears in the space for the column dendrogram. Question: How to create a margin at the top of the display for a title?
  - Bottom and right margins for the column and row names are set with `margins`, a numeric vector of length 2. These margins **do not** adjust to the length of the names.
  - Column labels are -90 degrees from a horizontal text line. Question: How to rotate the column labels (single character amino acid codes) so they are upright?
- `heatmap.2` from the `gplots` package
  - *Improvements over* `heatmap`: 1) Column labels can be rotated with the `srtCol` parameter. 2) Margin for row labels can be easily adjusted to accommodate short or full (long) names for organisms. Currently done by setting the margin to pre-determined widths, depending on whether short or full names are displayed. This could be a problem if an organism has a very long name. 3) A color-key with histogram (indicating the number of matrix entries within a certain range) is plotted.
  - The title appears in the space for the column dendrogram. A less-than-optimal solution is to use the color-key as a substitute for the main title.
- `pheatmap` from the `pheatmap` package
  - *Improvements over* `heatmap` *and* `heatmap.2`: 1) Margin for row labels is *automatically* adjusted to accommodate short or full (long) names for organisms. 2) Numbers can be displayed inside a heatmap matrix entry to show the usage of an amino acid (relative to the maximum for that organism). 3) The upper margin of the plot is wide enough to accomodate a two-line title above the column dendrogram.

- A simple color-key is plotted in the upper right corner of the display. It is simpler than the `heatmap.2`, and, unfortunately, it seems to use the same point size as the main title of the plot. The color-key is currently supressed because of the point size issue and because it is redundant if relative amino acid usage is plotted inside each heatmap matrix entry.
- Column labels can be rotated, but, as noted above, *it's complicated!*
- Question: How can the line width of the dendrograms be changed. They are currently too narrow.

Note that all of the currently implemented methods construct dendrograms use the `distfun` function to compute distances between vectors and the `hclust` function for clustering. Both of these functions are from the `stats` R package. `distfun` defaults to the `"euclidian"` method, and `hclust` defaults to the `"complete"` linkage method. These defaults may be changed (see the documentation for `distfun` and `hclust`) but are used in all of the heatmap functions currently implemented in `RBiotools`.

Value

None

Examples

```
## Not run:
myGenomes <- c("AP012306", "KK583188", "U00096", "CP000802", "CP000800")  # E. coli genomes
plotHeatMapAA(myGenomes)

## End(Not run)
```

`plotHeatMapBias` *plotHeatMapBias*

Description

A function for plotting heatmaps for position bias across a set of genomes

Usage

```
plotHeatMapBias(
  accessionList,
  organismIdentifier = "shortName",
  plotIdentifier = NULL,
  colorPalette = "cm",
  plotMethod = "heatmap.2"
)
```

Arguments

`accessionList`	character vector containing GenBank accession numbers
`organismIdentifier`	character vector indicating the name to be used for organisms; options are `"fullName"` or `"shortName"`; `"fullName"` is recommended for a diverse, multi-species set of genomes; `"shortName"` is recommended for a single-species set of genomes

plotIdentifier	character vector containing a title for the plot
colorPalette	character vector indicating the color palette to be used in the heatmap; options are the color palettes from the grDevices package: "cm", "topo", "terrain", "heat", "rainbow"
plotMethod	character vector indicating the R function to use for plotting the heatmap; options are "heatmap" from the stats package, "heatmap.2" from the gplots package, and "pheatmap" from the "pheatmap" package. This parameter is included during the development process. (See **Development Notes** in the documentation for plotHeatMapsAA.) When the best function for postion bias heatmaps is determined, this parameter will likely be removed.

Details

A heatmap is a graphical representation of data in a matrix that uses color-coding to represent different data values. plotHeatMapBias plots heatmaps for postion bias across a set of genomes (one or more) using a variety of methods from several R packages.

Development Notes

See notes in plotHeatMapAA documentation

Heatmaps are displayed in a .PNG format in a window external to R Studio, rather than in the R Studio plot pane. This approach allows precise control over the size of the plot, which can vary significantly if visualized in the R Studio plot pane. plotHeatMapBias writes a file named HeatMapBias.png in the current working directory.

Value

None

Examples

```
## Not run:
myGenomes <- c("AP012306", "KK583188", "U00096", "CP000802", "CP000800")  # E. coli genomes
plotHeatMapBias(myGenomes)

## End(Not run)
```

---

plotHeatMapCodon *plotHeatMapCodon*

Description

A function for plotting heatmaps for codon usage across a set of genomes

Usage

```
plotHeatMapCodon(
  accessionList,
  organismIdentifier = "shortName",
  plotIdentifier = NULL,
  colorPalette = "cm",
  plotMethod = "heatmap.2"
)
```

Arguments

accessionList	character vector containing GenBank accession numbers
organismIdentifier	character vector indicating the name to be used for organisms; options are "fullName" or "shortName"; "fullName" is recommended for a diverse, multi-species set of genomes; "shortName" is recommended for a single-species set of genomes
plotIdentifier	character vector containing a title for the plot
colorPalette	character vector indicating the color palette to be used in the heatmap; options are the color palettes from the grDevices package: "cm", "topo", "terrain", "heat", "rainbow"
plotMethod	character vector indicating the R function to use for plotting the heatmap; options are "heatmap" from the stats package, "heatmap.2" from the gplots package, and "pheatmap" from the "pheatmap" package. This parameter is included during the development process. (See **Development Notes** in the documentation for plotHeatMapAA.) When the best function for codon usage heatmaps is determined, this parameter will likely be removed.

Details

A heatmap is a graphical representation of data in a matrix that uses color-coding to represent different data values. plotHeatMapCodon plots heatmaps for codon usage across a set of genomes (one or more) using a variety of methods from several R packages.

Development Notes

See notes in plotHeatMapAA documentation

Heatmaps are displayed in a .PNG format in a window external to R Studio, rather than in the R Studio plot pane. This approach allows precise control over the size of the plot, which can vary significantly if visualized in the R Studio plot pane. plotHeatMapCodon writes a file named HeatMapCodon.png in the current working directory.

Value

None

Examples

```
## Not run:
myGenomes <- c("AP012306", "KK583188", "U00096", "CP000802", "CP000800")  # E. coli genomes
plotHeatMapCodon(myGenomes)

## End(Not run)
```

---

plotPanCore *plotPanCore*

Description

Creates a pan-core figure for a pan-genome

Usage

```
plotPanCore(homologyMatrix, xAxis = "accession", title = "")
```

Arguments

homologyMatrix	a dataframe constructed by runLinclust containing protein grouping information
xAxis	character vector indicating the labels to be used for organisms; options are "accession" (default), "fullName", or "shortName"
title	character vector containing the title for the Pan-Core figure

Details

plotPanCore creates a pan-core plot using gene-grouping information in a homology matrix contructed using runLinclust.

Value

None

Examples

```
## Not run:
myGenomes <- c("AP012306", "KK583188", "U00096", "CP000802", "CP000800") # E. coli genomes
homologyMatrix <- runLinclust(myGenomes)
plotPanCore(homologyMatrix)

## End(Not run)
```

---

plotUsage *plotUsage*

Description

A function for plotting the usage of amino acids or codons or the position bias for a genome or set of genomes

Usage

```
plotUsage(
  accessionList,
  plotData = "AA",
  codonOrder = "thirdPosition",
  plotType = "radar",
  plotColor = "darkblue",
  plotIdentifier = NULL
)
```

Arguments

accessionList	character vector containing GenBank accession numbers
plotData	character vector with values "AA" (amino acid), "codon" (codon), or "positionBias" (position bias)
codonOrder	character vector with values "firstPosition" or "thirdPosition" (default)
plotType	character vector with values "radar" or "circleBar" indicating the type of plot
plotColor	character vector with an R-recognized color name or an RGB hex code, such as "#FF0000" for red
plotIdentifier	character vector that is used in the chart title

Details

plotUsage creates "radar" charts (also known as "spider" charts) and "circle bar" charts (bar charts in polar coordinates) that show the usage of amino acids or codons or the position bias for a genome or set of genomes. In the specification of chart color, plotUsage recognizes 657 named colors known to R and all RGB hex codes, such as "#FF0000" for "red". If the user specifies a color that is not recognized, plotUsage outputs a message, choses a random color from the set of named colors, and creates the plot. Note that some of these colors are not particularly suitable for plotting. If the user specifies a plot identifier using the plotIdentifier parameter, the identifier will appear in the chart title. If a plot identifier is not specified and the accessionList parameter contains a single accession number, the full name of the organism is used as the identifer. If a plot identifier is not specified and the accessionList contains multiple accession numbers, "Multiple Organisms" is used as the identifier.

Value

None

Examples

```
## Not run:
E_coli <- c("AP012306", "KK583188", "U00096", "CP000802", "CP000800")

# Create a radar plot for amino acid usage across the set of genomes in the E_coli list
plotUsage(E_coli)

# Create a customized plot for the E_coli list
plotUsage(E_coli, plotData = "codon",
                  plotType = "circleBar",
                  plotColor = "darkred",
                  plotIdentifier = "Set of Escherichia coli Genomes")

## End(Not run)
```

prodigal *prodigal*

Description

An R implementation of the widely-used protein-coding gene prediction software tool for bacterial and archaeal genomes. The acronym stands for PROkaryotic DYnamic Programming Genefinding ALgorithm.

Usage

```
prodigal(genomeSequence, verbose = FALSE)
```

Arguments

genomeSequence	DNAString, DNAStringSet, or character name of a FASTA file containing a genomic sequence
verbose	logical, indicating whether to report algorithm progress consistent with the original C implementation

Details

Prodigal is a gene-finding program for microbial for genome annotation of either draft or finished microbial sequence. It was developed to predict translation initiation sites more accurately. It was designed to minimize the number of false positive predictions. This implementation exists entirely in the R layer, with code written in R and C, integrated using the Rcpp package.

Note: prodigal *will seldom, if ever, need to be called directly by the* RBiotools *user.* RBiotools *functions will automatically call* prodigal *when necessary.*

Value

allGenes.df – a data frame with called genes, genomic coordinates, scores, and nucleotide and protein product sequences

References

Hyall et al. *Prodigal: prokaryotic gene recognition and translation initiation site identification:* https://doi.org/10.1186/1471-2105-11-119

Examples

```
## Not run:
organism <- "U00096" # Escherichia coli str. K-12 substr. MG1655

# fetch sequence from list of genomic sequences
genomeSeq <- GenomeSeqList[organism][[organism]]

# Call prodigal with a DNAStringSet ...
geneCalls <- prodigal(genomeSeq)

# ... or use the name of a FASTA file in the working directory
geneCalls <- prodigal("U00096.fasta")
```

```
## End(Not run)
```

RBiotools *RBiotools: An R package for comparative microbial genomic analysis*

Description

The RBiotools package is a collection of object-oriented functions that work with sets of genome identifiers, such as GenBank accession numbers.

Details

RBiotools is a collection of functions and data structures that support comparative microbial genomics analysis. It is intended for use by clinical researchers, experimental biologists, and students in microbial genomics courses. Design goals include to be simple and intuitive and to handle data conversion and manipulation seamlessly. The large majority of data is stored internally in global data structures, and reliance on external files is minimized. Most of the currently implemented functions require a single parameter, a set of one or more GenBank accession numbers. Additional parameters for the plotting functions are available to users to select alternative methods, to add custom titles, or to choose color palettes. RBiotools functions fall into five categories:

1. **Initialization** – Initializing RBiotools global data structures prior to any RBiotools session
2. **Data Importation** – Importing of genomic data, specifically genomic sequences and gene counts, downloaded to RBiotools from GenBank or uploaded to RBiotools from FASTA files
3. **Data Extraction** – Fetching data from RBiotools global data structures for viewing or analysis
4. **Data Analysis** – Analyzing genomic or derived data
5. **Data Visualization** – Creating various plots and charts of genomic and derived data

None of these functions must be called explicitly by users except the functions for creating plots and charts and the function for uploading user-supplied genomic data. All other functions will be called automatically if initialization of global data structures is required, if data needs to be downloaded from GenBank, or if analysis of genomic data, such as gene calling, is necessary.

Future Development

- The current version of RBiotools implements the functionality of CMG-Biotools. The CMG-Biotools system, usually installed on a virtual machine, contains the GeneWiz browser to create and visualize genome atlases. The RBiotools function createAtlas, written entirely in R, creates an SVG file that can be opened with a browser to visualize the genome atlas. createAtlas currently produces basic genome atlases that display coding sequences and rRNA genes on the forward and reverse strands, as well as stacking energy, position preference, and GC-skew. Future versions of createAtlas will support additional analyses of genomic data and improved visualization capabilities.
- Future versions of RBiotools will integrate **pbdR** (Programming with Big Data in R) packages to support utilization of high-performance computing (HPC) resources.

The RBiotools developers welcome

- Reports of unexpected or faulty behavior in RBiotools
- Suggestions for streamlining and simplifying the use of RBiotools, for making RBiotools more intuitive, or for any other enhancements or modifications
- Suggestions for clarifying or improving RBiotools documentation
- Suggestions for additional RBiotools functionality, the highest priority being ***what is most useful to clinical researchers and experimental biologists***

References

Vesth et al. *CMG-Biotools, a Free Workbench for Basic Comparative Microbial Genomics* http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0060120)

Hallin, Binnewies, and Ussery: *The genome BLASTatlas—a GeneWiz extension for visualization of whole-genome homology* http://pubs.rsc.org/en/content/articlelanding/2008/mb/b717118h/

Ostrouchov, Chen, Schmidt, and Patel: *Programming with Big Data in R*. 6th Extremely Large Databases Conference (XLDB), Stanford, CA, USA, September 2012.

Schmidt, Chen, Matheson, and Ostrouchov: *Programming with BIG Data in R: Scaling Analytics from One to Thousands of Nodes*. Big Data Research, Volume 8, July 2017. https://doi.org/10.1016/j.bdr.2016.10.002

---

removeGenomes *removeGenomes*

Description

Removes a set of genomes from an RBiotools session

Usage

```
removeGenomes(accessionList)
```

Arguments

accessionList	character vector containing accession numbers of genomes to be removed from the RBiotools global data infrastructure

Details

The primary function of this function is to reclaim storage.

All data for each organism specified in the accessionList parameter is removed from all RBiotools global data structuress, specifically:

- GenomeSeqList
- orgName.df
- orgData.df
- wgsName.df
- ProdigalCalls
- RNAmmerCalls
- rRNAcalled.df

Value

None

`rnammer` *rnammer*

Description

An R implementation of the widely-used ribosomal RNA gene prediction software tool for bacterial, archaeal, eukaryotic genomes.

Usage

```
rnammer(seqSet, dom, mol)
```

Arguments

`seqSet`	DNAStringSet containing DNA sequence(s) to search for ribosomal RNA genes
`dom`	character Domain: "ARC" for Archaea, "BAC" for Bacteria, "EUK" for Eukarya
`mol`	character Molecule: "TSU" for 5/8s, "SSU" for 16/18s, "LSU" for 23/28s

Details

RNAmmer implements computational predictors for the major rRNA species from all kingdoms of life. The software uses hidden Markov models trained on data from the 5S ribosomal RNA database and the European ribosomal RNA database project. A pre-screening step makes the method fast with little loss of sensitivity. This implementation exists entirely in the R layer, including the hidden Markov models, with code written in R and C, integrated using the `Rcpp` package.

Note: `rnammer` *will seldom, if ever, need to be called directly by the* `RBiotools` *user.* `RBiotools` *functions will automatically call* `rnammer` *when necessary.*

Value

allGenes.df – a data frame with called genes, genomic coordinates, scores, and nucleotide sequences

References

Lagesen et al. *RNAmmer: consistent and rapid annotation of ribosomal RNA genes:* https://doi.org/10.1093/nar/gkm160

Examples

```
## Not run:
org <- "U00096"                         # Escherichia coli str. K-12 substr. MG1655
genomeSeq <- GenomeSeqList[org][[org]] # fetch sequence from list of genomic sequences
geneCalls <- rnammer(genomeSeq)         # Call rnammer with a DNAStringSet

## End(Not run)
```

runLinclust *runLinclust*

Description

Cluster proteins from a set of organisms

Usage

```
runLinclust(accessionList, mode = "SLOW", verbose = FALSE)
```

Arguments

accessionList	character vector containing GenBank accession numbers

Details

This function groups all proteins from all organisms in the accessionList using an R and C++ implementation of the Linclust method of Steinegger and Söding. Minor differences in protein groupings from the Steinegger and Söding implementation and the RBiotools implementation may be observed. Among other differences, Steinegger and Söding's code employs an SIMD Smith-Waterman C++ library for local, gapped protein alignments, while RBiotools uses the `pairwiseAlignment` function from the R `Biostrings` package.

Value

homologyMatrix.df, a data frame with a column for each organism and a row for each protein group, with data frame elements containing counts of proteins for each organism / protein group combination.

References

Steinegger and Söding *Clustering huge protein sequence sets in linear time:* https://doi.org/10.1038/s41467-018-04964-5

Examples

```
## Not run:
accessionIDs <- c(
  "AP012306",  # Escherichia coli str. K-12 substr. MDS42 DNA
  "KK583188",  # Escherichia coli DSM 30083 = JCM 1649 = ATCC 11775
  "U00096",    # Escherichia coli str. K-12 substr. MG1655
  "CP000802",  # Escherichia coli HS
  "CP000800",  # Escherichia coli E24377A
  "AP009378",  # Escherichia coli SE15
  "FM180568",  # Escherichia coli 0127:H6 E2348/69
  "CU928163",  # Escherichia coli UMN026
  "CP008957",  # Escherichia coli O157:H7 str. EDL933
  "CP027027",  # Shigella dysenteriae strain E670/74
  "CP026802",  # Shigella sonnei strain ATCC 29930
  "CP026877",  # Shigella boydii strain ATCC 35964
  "CP026793",  # Shigella flexneri strain 74-1170
  "CP015831"   # Escherichia coli O157 strain 644-PT8
```

```
)

runLinclust(accessionIDs)

## End(Not run)
```

---

runProdigal *runProdigal*

Description

A convenience function that calls the prodigal gene calling function for a set of genomes and adds the results to the RBiotools global data store

Usage

```
runProdigal(accessionList, verbose = TRUE)
```

Arguments

accessionList	character vector containing GenBank accession numbers
verbose	logical, indicating whether to report algorithm progress consistent with the original C implementation

Details

This convenience function calls the prodigal gene calling function for a set of genomes specified by accession numbers. For each genome in the set, runProdigal adds the called genes to ProdigalCalls, an RBiotools global data frame containing called genes, genomic coordinates, scores, and nucleotide and protein product sequences. The function fetchProdigalCalls can then be used to extract a data frame from ProdigalCalls with gene calls for a specified genome.

Note: runProdigal *will seldom, if ever, need to be called directly by the* RBiotools *user.* RBiotools *functions will automatically initiate calling of protein genes, if required for use by another function.*

Value

None

Examples

```
## Not run:
myGenomes <- c("CP003284",
               "CP003597",
               "NONSENSE", # skipped, no corresponding genomic sequence
               "CP003289",
               "CP006632"
              )
runProdigal(myGenomes)
calledGenes <- fetchProdigalCalls("CP003284")

## End(Not run)
```

runRNAmmer *runRNAmmer*

Description

A convenience function that calls the rnammer gene calling function for a set of genomes and adds the results to the RBiotools global data store

Usage

```
runRNAmmer(accessionList, scope = "ALL")
```

Arguments

accessionList	character vector containing GenBank accession numbers

Details

This convenience function calls the rnammer gene calling function for a set of genomes specified by accession numbers, using genomic sequences found in the global data list GenomeSeqList. If a genomic sequence is not found in the global data store for a specified accession number, the user is prompted to download genomic information using the function downloadGenBank. For each genome for which a genomic sequence is found, runRNAmmer adds the called genes to RNAmmerCalls, an RBiotools global data frame containing called genes, genomic coordinates, scores, and nucleotide sequences. The function fetchRNAmmerCalls can then be used to extract a data frame from RNAmmerCalls with gene calls for a specified genome.

Value

None

Examples

```
## Not run:
myGenomes <- c("CP003284",
               "CP003597",
          "NONSENSE", # This entry will be skipped, since a corresponding genomic sequence will not be found
               "CP003289",
               "CP006632"
              )
runRNAmmer(myGenomes) #' calledGenes <- fetchRNAmmerCalls("CP003284") #'
## End(Not run)
```

writeProteinFAA *writeProteinFAA*

Description

A function for writing protein FASTA files to the current working directory for a set of genomes

Usage

```
writeProteinFAA(accessionList)
```

Arguments

accessionList	character vector containing GenBank accession numbers

Details

writeProteinFAA writes protein FASTA files to the current working directory for a set of genomes. The genomes are specified in a list of GenBank (or GenBank-like) accession numbers. The protein FASTA files use Prodigal formatting for the headers; each protein translation is written on a single line and terminated with a "*" character.

Value

None

References

Hyall et al. *Prodigal: prokaryotic gene recognition and translation initiation site identification:* https://doi.org/10.1186/1471-2105-11-119

Hyatt *Understanding the Prodigal Output:* https://github.com/hyattpd/Prodigal/wiki/Understanding-the-Pro

Examples

```
## Not run:
genomeSet <- c(
  "AP012306",  # Escherichia coli str. K-12 substr. MDS42 DNA
  "CP018295"   # Bacillus subtilis subsp. subtilis strain J-5
)

runProdigal(genomeSet)

writeProteinFAA(genomeSet)

## End(Not run)
```

Index