Package 'fastreeR'

May 1, 2026

Type Package

Title Phylogenetic, Distance and Other Calculations on VCF and Fasta Files

Version 2.3.0

Backend 2.7.1

biocViews Phylogenetics, Metagenomics, Clustering

Description Calculate distances, build phylogenetic trees or perform hierarchical clustering between the samples of a VCF or FASTA file. Functions are implemented in Java-11 and called via rJava. Parallel implementation that operates directly on the VCF or FASTA file for fast execution.

License GPL-3

Encoding UTF-8

Language en-US

LazyData false

Depends R (>= 4.4)

Imports ape, data.table, dynamicTreeCut, methods, R.utils, rJava, stats, stringr, utils

SystemRequirements Java (>= 11)

RoxygenNote 7.3.3

URL https://github.com/gkanogiannis/fastreeR, https://github.com/gkanogiannis/BioInfoJava-Utils

BugReports https://github.com/gkanogiannis/fastreeR/issues

Suggests BiocFileCache, BiocStyle, ggtree, graphics, knitr, memuse, rmarkdown, spelling, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

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

git_branch devel

git_last_commit a85f8f1

git_last_commit_date 2026-04-30

Repository Bioconductor 3.24

Date/Publication 2026-05-01

Author Anestis Gkanogiannis [aut, cre] (ORCID:
<https://orcid.org/0000-0002-6441-0688>)

Maintainer Anestis Gkanogiannis <anestis@gkanogiannis.com>

Contents

`fastreeR-package` *fastreeR: Phylogenetic, Distance and Other Calculations on VCF and Fasta Files*

Description

Calculate distances, build phylogenetic trees or perform hierarchical clustering between the samples of a VCF or FASTA file. Functions are implemented in Java-11 and called via rJava. Parallel implementation that operates directly on the VCF or FASTA file for fast execution.

Author(s)

Maintainer: Anestis Gkanogiannis <anestis@gkanogiannis.com> (ORCID)

See Also

Useful links:

- https://github.com/gkanogiannis/fastreeR
- https://github.com/gkanogiannis/BioInfoJava-Utils
- Report bugs at https://github.com/gkanogiannis/fastreeR/issues

`dist2clusters` *Perform Hierarchical Clustering and tree pruning on a distance matrix*

Description

Performs Hierarchical Clustering on a distance matrix (i.e. calculated with `vcf2dist` or `fasta2dist`) and generates a phylogenetic tree (complete linkage by default; single, complete, and average linkage are supported by the Java backend), as in `dist2tree`. The phylogenetic tree is then pruned with `cutreeDynamic` to get clusters (as in `tree2clusters`).

Usage

```
dist2clusters(
  inputDist,
  cutHeight = NULL,
  minClusterSize = 1,
  extra = TRUE,
  verbose = FALSE
)
```

Arguments

`inputDist`	Input distances file location (generated with `vcf2dist` or `fasta2dist`). File can be gzip compressed. Or a `dist` distances object.
`cutHeight`	Define at which height to cut tree. Default automatically defined.
`minClusterSize`	Minimum size of clusters. Default 1.
`extra`	Boolean whether to use extra parameters for the `cutreeDynamic`.
`verbose`	Logical. If TRUE, enables verbose output from the Java backend.

Value

A list of :

- `character` `vector` of the generated phylogenetic tree in Newick format
- `character` `vector` of the clusters. Each row contains data for a cluster, separated by space. The id of the cluster, the size of the cluster (number of elements) and the names of its elements, Cluster id 0 contains all the objects not assigned to a cluster (singletons). Example clusters output :

0	3	Sample1	Sample2	Sample3
1	3	Sample4	Sample5	Sample6
2	2	Sample7	Sample8	
3	2	Sample9	Sample0	

Author(s)

Anestis Gkanogiannis, <anestis@gkanogiannis.com>

References

Java implementation: https://github.com/gkanogiannis/BioInfoJava-Utils

Examples

```
my.clust <- dist2clusters(
    inputDist =
        system.file("extdata", "samples.vcf.dist.gz", package = "fastreeR"),
    verbose = TRUE
)
```

`dist2tree` *Generate phylogenetic tree from samples of a distance matrix*

Description

Performs Hierarchical Clustering on a distance matrix (i.e. calculated with `vcf2dist` or `fasta2dist`) and generates a phylogenetic tree (complete linkage by default; single, complete, and average linkage are supported by the Java backend).

Usage

```
dist2tree(inputDist, verbose = FALSE)
```

Arguments

`inputDist`	Input distances file location (generated with `vcf2dist` or `fasta2dist`). File can be gzip compressed. Or a `dist` distances object.
`verbose`	Logical. If TRUE, enables verbose output from the Java backend.

Value

A `character` `vector` of the generated phylogenetic tree in Newick format.

Author(s)

Anestis Gkanogiannis, <anestis@gkanogiannis.com>

References

Java implementation: https://github.com/gkanogiannis/BioInfoJava-Utils

Examples

```
my.tree <- dist2tree(
    inputDist =
    system.file("extdata", "samples.vcf.dist.gz", package = "fastreeR"),
    verbose = TRUE
)
```

`fasta2dist` *Calculate distances between sequences of a FASTA file*

Description

This function calculates a d2_S type dissimilarity measurement between the `n` sequences (which can represent samples) of a FASTA file. See doi:10.1186/s1285901611863 for more details.

Usage

```
fasta2dist(
  ...,
  outputFile = NULL,
  threads = 2,
  kmer = 6,
  normalize = FALSE,
  compress = TRUE,
  verbose = FALSE
)
```

Arguments

`...`	Input fasta files locations (uncompressed or gzip compressed).
`outputFile`	Output distances file location.
`threads`	Number of java threads to use.
`kmer`	Kmer length to use for analyzing fasta sequences.
`normalize`	Normalize on sequences length.
`compress`	Compress output (adds .gz extension).
`verbose`	Logical. If TRUE, enables verbose output from the Java backend.

Value

A `dist` distances object of the calculation.

Author(s)

Anestis Gkanogiannis, <anestis@gkanogiannis.com>

References

Java implementation: https://github.com/gkanogiannis/BioInfoJava-Utils

Examples

```
my.dist <- fasta2dist(
    inputfile = system.file("extdata", "samples.fasta.gz",
        package = "fastreeR"
    )
)
```

`tree2clusters` *Perform Hierarchical Clustering and tree pruning on a phylogenetic tree*

Description

The phylogenetic tree is pruned with `cutreeDynamic` to get clusters.

Usage

```
tree2clusters(
  treeStr,
  treeDistances = NULL,
  treeLabels = NULL,
  cutHeight = NULL,
  minClusterSize = 1,
  extra = TRUE,
  verbose = FALSE
)
```

Arguments

`treeStr`	A `character` `vector` of a phylogenetic tree in Newick format
`treeDistances`	`numeric` `matrix` of distances, that were used to generate the tree. If NULL, it will be inferred from tree branch lengths.
`treeLabels`	A `character` `vector` of tree leaf labels.
`cutHeight`	Define at which height to cut tree. Default automatically defined.
`minClusterSize`	Minimum size of clusters. Default 1.
`extra`	Boolean whether to use extra parameters for the `cutreeDynamic`.
`verbose`	Logical. If TRUE, enables verbose output from the Java backend.

Value

- `character` `vector` of the clusters. Each row contains data for a cluster, separated by space. The id of the cluster, the size of the cluster (number of elements) and the names of its elements, Cluster id 0 contains all the objects not assigned to a cluster (singletons). Example clusters output :

0	3	Sample1	Sample2	Sample3
1	3	Sample4	Sample5	Sample6
2	2	Sample7	Sample8	
3	2	Sample9	Sample0	

Author(s)

Anestis Gkanogiannis, <anestis@gkanogiannis.com>

References

Java implementation: https://github.com/gkanogiannis/BioInfoJava-Utils

Examples

```
my.clust <- tree2clusters(
    treeStr = dist2tree(
        inputDist = system.file("extdata", "samples.vcf.dist.gz",
            package = "fastreeR"
        )
    ),
    verbose = TRUE
)
```

`vcf2clusters` *Perform Hierarchical Clustering and tree pruning on samples of VCF file*

Description

Performs Hierarchical Clustering on a distance matrix calculated as in `vcf2dist` and generates a phylogenetic tree (complete linkage by default; single, complete, and average linkage are supported by the Java backend), as in `dist2tree`. The phylogenetic tree is then pruned with `cutreeDynamic` to get clusters (as in `tree2clusters`).

Usage

```
vcf2clusters(
  inputFile,
  threads = 2,
  cutHeight = NULL,
  minClusterSize = 1,
  extra = TRUE,
  verbose = FALSE
)
```

Arguments

`inputFile`	Input vcf file location (uncompressed or gzip compressed).
`threads`	Number of java threads to use.
`cutHeight`	Define at which height to cut tree. Default automatically defined.
`minClusterSize`	Minimum size of clusters. Default 1.
`extra`	Boolean whether to use extra parameters for the `cutreeDynamic`.
`verbose`	Logical. If TRUE, enables verbose output from the Java backend.

Details

Biallelic or multiallelic (maximum 7 alternate alleles) SNP and/or INDEL variants are considered, phased or not. Some VCF encoding examples are:

- heterozygous variants : `1/0` or `0/1` or `0/2` or `1|0` or `0|1` or `0|2`
- homozygous to the reference allele variants : `0/0` or `0|0`
- homozygous to the first alternate allele variants : `1/1` or `1|1`

If there are n samples and m variants, an nxn zero-diagonal symmetric distance matrix is calculated. The calculated cosine type distance (1-cosine_similarity)/2 is in the range [0,1] where value 0 means completely identical samples (cosine is 1), value 0.5 means perpendicular samples (cosine is 0) and value 1 means completely opposite samples (cosine is -1).

The calculation is performed by a Java back-end implementation, that supports multi-core CPU utilization and can be demanding in terms of memory resources. By default a JVM is launched with a maximum memory allocation of 512 MB. When this amount is not sufficient, the user needs to reserve additional memory resources, before loading the package, by updating the value of the `java.parameters` option. For example in order to allocate 4GB of RAM, the user needs to issue `options(java.parameters="-Xmx4g")` before `library(fastreeR)`.

Value

A list of :

- `dist` distances object.
- `character vector` of the generated phylogenetic tree in Newick format
- `character vector` of the clusters. Each row contains data for a cluster, separated by space. The id of the cluster, the size of the cluster (number of elements) and the names of its elements, Cluster id 0 contains all the objects not assigned to a cluster (singletons). Example clusters output :

0	3	Sample1	Sample2	Sample3
1	3	Sample4	Sample5	Sample6
2	2	Sample7	Sample8	
3	2	Sample9	Sample0	

Author(s)

Anestis Gkanogiannis, <anestis@gkanogiannis.com>

References

Java implementation: https://github.com/gkanogiannis/BioInfoJava-Utils

Examples

```
my.clust <- vcf2clusters(
    inputFile = system.file("extdata", "samples.vcf.gz",
        package = "fastreeR"
    )
)
```

`vcf2dist` *Calculate distances between samples of a VCF file*

Description

This function calculates a cosine type dissimilarity measurement between the n samples of a VCF file.

Usage

```
vcf2dist(
  inputFile,
  outputFile = NULL,
  threads = 2,
  compress = FALSE,
  verbose = FALSE,
  windowBp = NULL,
  windowVariants = NULL,
  windowStep = NULL,
  windowMinVariants = 1L,
  longFormat = FALSE
)
```

Arguments

`inputFile`	Input vcf file location (uncompressed or gzip compressed).
`outputFile`	Output distances file location.
`threads`	Number of java threads to use.
`compress`	Compress output (adds .gz extension).
`verbose`	Logical. If TRUE, enables verbose output from the Java backend.
`windowBp`	Optional positive integer. Emit one matrix per window of N base pairs. Mutually exclusive with `windowVariants`.
`windowVariants`	Optional positive integer. Emit one matrix per N consecutive variants. Mutually exclusive with `windowBp`.
`windowStep`	Optional positive integer. Window step (defaults to window size, i.e. tiled). Sliding windows are not yet implemented and will be rejected by the Java backend.
`windowMinVariants`	Minimum number of variants required to emit a window (default 1; smaller windows are skipped silently).
`longFormat`	Logical. In windowed mode, return a long-form `data.frame` instead of a list of `dist` objects.

Details

Biallelic or multiallelic (maximum 7 alternate alleles) SNP and/or INDEL variants are considered, phased or not. Some VCF encoding examples are:

- heterozygous variants : `1/0` or `0/1` or `0/2` or `1|0` or `0|1` or `0|2`
- homozygous to the reference allele variants : `0/0` or `0|0`
- homozygous to the first alternate allele variants : `1/1` or `1|1`

If there are `n` samples and `m` variants, an `nxn` zero-diagonal symmetric distance matrix is calculated. The calculated cosine type distance (1-cosine_similarity)/2 is in the range [0,1] where value 0 means completely identical samples (cosine is 1), value 0.5 means perpendicular samples (cosine is 0) and value 1 means completely opposite samples (cosine is -1).

The calculation is performed by a Java backend implementation, that supports multi-core CPU utilization and can be demanding in terms of memory resources. By default a JVM is launched

with a maximum memory allocation of 512 MB. When this amount is not sufficient, the user needs to reserve additional memory resources, before loading the package, by updating the value of the `java.parameters` option. For example in order to allocate 4GB of RAM, the user needs to issue `options(java.parameters="-Xmx4g")` before `library(fastreeR)`.

Output file, if provided, will contain `n+1` lines. The first line contains the number `n` of samples and number `m` of variants, separated by space. Each of the subsequent `n` lines contains `n+1` values, separated by space. The first value of each line is a sample name and the rest `n` values are the calculated distances of this sample to all the samples. Example output file of the distances of 3 samples calculated from 1000 variants:

```
3 1000
Sample1  0.0  0.5  0.2
Sample2  0.5  0.0  0.9
Sample3  0.2  0.9  0.0
```

Windowed mode. Setting `windowBp` or `windowVariants` (mutually exclusive) instructs the Java backend to emit one distance matrix per genomic window. Windows never straddle chromosomes. The return value changes accordingly:

- `longFormat = FALSE` (default): a named `list` of `dist` objects, one per window. List names follow the format `"chrom:start-end"`.
- `longFormat = TRUE`: a single `data.frame` with columns `chrom`, `start`, `end`, `sample_i`, `sample_j`, `dist`.

Value

In non-windowed mode, a `dist` distances object. In windowed mode, either a named `list` of `dist` objects (default) or a long-form `data.frame` (when `longFormat = TRUE`).

Author(s)

Anestis Gkanogiannis, <anestis@gkanogiannis.com>

References

Java implementation: https://github.com/gkanogiannis/BioInfoJava-Utils

Examples

```
my.dist <- vcf2dist(
    inputFile = system.file("extdata", "samples.vcf.gz",
        package = "fastreeR"
    )
)
```

vcf2istats *Calculate various per sample statistics from a VCF file*

Description

Only biallelic SNPs are considered. For each sample, the following statistics are calculated :

- INDIV : Sample name
- N_SITES : Total number of SNPs
- N_HET : Number of SNPs with heterozygous call (`0/1` or `0|1` or `1/0` or `1|0`)
- N_ALT : Number of SNPs with alternate homozygous call (`1/1` or `1|1`)
- N_REF : Number of SNPs with reference homozygous call (`0/0` or `0|0`)
- N_MISS : Number of SNPs with missing call (`./.` or `.|.`)
- P_HET : Percentage of heterozygous calls
- P_ALT : Percentage of alternate homozygous calls
- P_REF : Percentage of reference homozygous calls
- P_MISS : Percentage of missing calls (missing rate)

Usage

```
vcf2istats(inputFile, outputFile = NULL)
```

Arguments

`inputFile`	Input vcf file location (uncompressed or gzip compressed).
`outputFile`	Output samples statistics file location.

Value

A `data.frame` of sample statistics.

Author(s)

Anestis Gkanogiannis, <anestis@gkanogiannis.com>

References

Java implementation: https://github.com/gkanogiannis/BioInfoJava-Utils

Examples

```
my.istats <- vcf2istats(
    inputFile =
        system.file("extdata", "samples.vcf.gz", package = "fastreeR")
)
```

`vcf2tree` *Generate phylogenetic tree from samples of a VCF file*

Description

This function calculates a distance matrix between the samples of a VCF file as in `vcf2dist` and performs Hierarchical Clustering on this distance matrix as in `dist2tree`. A phylogenetic tree is calculated by hierarchical clustering of the distance matrix (complete linkage by default; single, complete, and average linkage are supported by the Java backend).

Usage

```
vcf2tree(
  inputFile,
  threads = 1,
  verbose = FALSE,
  bootstrap = 0,
  windowBp = NULL,
  windowVariants = NULL,
  windowStep = NULL,
  windowMinVariants = 1L
)
```

Arguments

`inputFile`	Input vcf file location (uncompressed or gzip compressed).
`threads`	Number of java threads to use (default 1).
`verbose`	Logical. If TRUE, enables verbose output from the Java backend.
`bootstrap`	Number of bootstrap replicates to perform (default 0, no bootstrapping).
`windowBp`	Optional positive integer. Emit one tree per window of N base pairs. Mutually exclusive with `windowVariants`.
`windowVariants`	Optional positive integer. Emit one tree per N consecutive variants. Mutually exclusive with `windowBp`.
`windowStep`	Optional positive integer. Window step (defaults to window size, i.e. tiled). Sliding windows are not yet implemented and will be rejected by the Java backend.
`windowMinVariants`	Minimum number of variants required to emit a window (default 1; smaller windows are skipped silently).

Details

If the `bootstrap` parameter is set to a positive integer, the Java backend performs streaming bootstrap sampling of variants for the requested number of replicates. Bootstrap support values are encoded in the returned Newick string at internal nodes (percent support across replicates). Note that enabling bootstrapping increases runtime and memory usage proportionally to the number of replicates.

Biallelic or multiallelic (maximum 7 alternate alleles) SNP and/or INDEL variants are considered, phased or not. Some VCF encoding examples are:

- heterozygous variants : `1/0` or `0/1` or `0/2` or `1|0` or `0|1` or `0|2`
- homozygous to the reference allele variants : `0/0` or `0|0`
- homozygous to the first alternate allele variants : `1/1` or `1|1`

If there are `n` samples and `m` variants, an `nxn` zero-diagonal symmetric distance matrix is calculated. The calculated cosine type distance (1-cosine_similarity)/2 is in the range [0,1] where value 0 means completely identical samples (cosine is 1), value 0.5 means perpendicular samples (cosine is 0) and value 1 means completely opposite samples (cosine is -1).

The calculation is performed by a Java backend implementation, that supports multi-core CPU utilization and can be demanding in terms of memory resources. By default a JVM is launched with a maximum memory allocation of 512 MB. When this amount is not sufficient, the user needs to reserve additional memory resources, before loading the package, by updating the value of the `java.parameters` option. For example in order to allocate 4GB of RAM, the user needs to issue `options(java.parameters="-Xmx4g")` before `library(fastreeR)`.

Windowed mode. Setting `windowBp` or `windowVariants` (mutually exclusive) instructs the Java backend to emit one Newick tree per genomic window. Windows never straddle chromosomes. Bootstrap is not supported in windowed mode and will raise an error. The return value becomes a `data.frame` with one row per window and columns `chrom`, `start`, `end`, `nvariants`, `newick`.

Value

In non-windowed mode, a `character` vector of the generated phylogenetic tree in Newick format. In windowed mode, a `data.frame` with columns `chrom`, `start`, `end`, `nvariants`, `newick` (one row per window).

Author(s)

Anestis Gkanogiannis, <anestis@gkanogiannis.com>

References

Java implementation: https://github.com/gkanogiannis/BioInfoJava-Utils

Examples

```
my.tree <- vcf2tree(
    inputFile = system.file("extdata", "samples.vcf.gz",
        package = "fastreeR"
    )
)
```

Index