Package 'CGRphylo2'

November 5, 2026

Type Package

Title Chaos Game Representation for Phylogenetic Analysis

Version 0.99.2

Date 2026-06-12

Description An alignment-free phylogenetic analysis method for viral genomes using Chaos Game Representation (CGR), a technique based on statistical physics concepts. Viruses exhibit high mutation rates, facilitating rapid evolution and emergence of new species, subspecies, strains, and recombinant forms. Accurate classification is crucial for understanding viral evolution and therapeutic development. Traditional phylogenetic methods require sequence alignment, which is computationally intensive. CGRphylo2 addresses this by implementing CGR-based whole-genome comparison that is fast, accurate, and computationally efficient. The package successfully classifies closely related viral lineages (demonstrated on SARS-CoV-2 lineages A and B), identifies recombinants (such as the XBB variant), and distinguishes multiple strains simultaneously. It processes sequences 5-13.7x faster than alignment-based methods (Clustal-Omega) with linear computational scaling. As a k-mer based approach, it enables simultaneous comparison of numerous closely-related sequences of different lengths. The package creates frequency matrices for distance calculations and phylogenetic tree construction, with outputs compatible with standard formats (MEGA, PHYLIP, Newick). Methods are based on Thind and Sinha (2023) <doi:10.2174/0113892029264990231013112156>.

License GPL-3

Encoding UTF-8

LazyData false

Depends R (>= 4.5.0)

Imports BiocParallel, Biostrings

Suggests BiocStyle, knitr, rmarkdown, testthat (>= 3.0.0), ape, seqinr, stringr, ggplot2, RColorBrewer, treeio

VignetteBuilder knitr

biocViews Phylogenetics, Genetics, SequenceMatching, Alignment, Clustering, Visualization, Software, MultipleSequenceAlignment, MultipleComparison, Classification

BugReports https://github.com/amarinderthind/CGRphylo2/issues

URL https://github.com/amarinderthind/CGRphylo2

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

git_url https://git.bioconductor.org/packages/CGRphylo2

git_branch devel

git_last_commit 05a7d66

git_last_commit_date 2026-06-11

Repository Bioconductor 3.24

Date/Publication 2026-11-05

Author Amarinder Singh Thind [aut, cre] (ORCID:
<https://orcid.org/0000-0003-4592-0380>)

Maintainer Amarinder Singh Thind <thind.amarinder@gmail.com>

Contents

calculateDistanceMatrix 2
cgrplot 3
create_meta 4
filter_N 5
from_DNAStringSet 6
matrixDistance 6
parallelCGR 7
plot_cgr 8
saveMegaDistance 9
savePhylipDistance 10

Index **12**

`calculateDistanceMatrix` *Calculate pairwise distance matrix for multiple sequences*

Description

Computes a full pairwise distance matrix from a list of CGR frequency matrices.

Usage

```
calculateDistanceMatrix(freq_matrices, distance_type = "Euclidean")
```

Arguments

`freq_matrices`	List. A named list of frequency matrices, one per sequence, as returned by parallelCGR().
`distance_type`	Character. Type of distance to calculate: "Euclidean" (default), "S_Euclidean", or "Manhattan".

Details

This function calculates all pairwise distances between sequences based on their CGR frequency matrices. The resulting distance matrix can be used for phylogenetic tree construction, clustering, or other downstream analyses.

Value

Numeric matrix. A symmetric distance matrix with sequence names as row and column names.

Examples

```
# Build two minimal frequency matrices (4 k-mers each, normalized)
fm1 <- matrix(c(0.4, 0.3, 0.2, 0.1), ncol = 1,
              dimnames = list(c("A", "C", "G", "T"), NULL))
fm2 <- matrix(c(0.1, 0.2, 0.3, 0.4), ncol = 1,
              dimnames = list(c("A", "C", "G", "T"), NULL))
fm3 <- matrix(c(0.25, 0.25, 0.25, 0.25), ncol = 1,
              dimnames = list(c("A", "C", "G", "T"), NULL))

freq_list <- list(Seq1 = fm1, Seq2 = fm2, Seq3 = fm3)

dist_matrix <- calculateDistanceMatrix(freq_list, distance_type = "Euclidean")
print(round(dist_matrix, 4))
```

---

cgrplot *Generate CGR plot coordinates for a DNA sequence*

Description

Creates x and y coordinates for visualizing a Chaos Game Representation (CGR) plot of a DNA sequence. The CGR plot is a 2D representation that reveals patterns and composition of genomic sequences.

Usage

```
cgrplot(seq_index)
```

Arguments

seq_index	Integer. The index of the sequence in the global fasta_filtered object to plot.

Details

The Chaos Game Representation is an iterative mapping technique that creates a 2D representation of DNA sequences. Each base (A, C, G, T) is assigned to a corner of a unit square, and the sequence is plotted by iteratively moving halfway from the current position to the corner corresponding to the next base.

The resulting plot has fractal properties and reveals sequence composition, repeats, and other genomic features. Different sequences create distinct patterns that reflect their underlying genomic structure.

Corner assignments (standard):

- A: (0, 0) - bottom left
- C: (1, 0) - bottom right
- G: (1, 1) - top right
- T: (0, 1) - top left

Value

Matrix with two columns (x and y coordinates) for plotting the CGR. Each row represents the position of one nucleotide in the CGR space.

References

Jeffrey HJ (1990). Chaos game representation of gene structure. Nucleic Acids Research, 18(8):2163-2170.

Examples

```
assign("fasta_filtered", list(seq1 = "ATCGATCGATCGATCGATCG"), envir = .GlobalEnv)
cgr_coords <- cgrplot(1)
plot(cgr_coords[, 1], cgr_coords[, 2],
    main = "CGR Plot", xlab = "", ylab = "", cex = 0.5, pch = 4
)
rm(fasta_filtered, envir = .GlobalEnv)
```

---

create_meta *Create metadata table for sequences*

Description

Extracts and compiles metadata including sequence length, GC content, and N content from FASTA sequences.

Usage

```
create_meta(fastafile, N_filter)
```

Arguments

fastafile	List. A list of DNA sequences read by seqinr::read.fasta()
N_filter	Integer. N filter threshold (for reference in output)

Details

This function provides useful summary statistics for quality control and understanding sequence characteristics before phylogenetic analysis.

Value

data.frame. A data frame with columns: name, length, GC_content, N_content

Examples

```
# Create test sequences
test_seqs <- list(
    seq1 = "ATCGATCGATCG",
    seq2 = "GCTAGCTAGCTA",
    seq3 = "AAAATTTTCCCCGGGG"
)

# Get metadata
meta <- create_meta(test_seqs, N_filter = 50)
print(meta)
```

`filter_N` *Filter sequences by N content*

Description

Filters a list of DNA sequences by removing those with too many ambiguous (N) bases.

Usage

```
filter_N(fastafile, N_filter)
```

Arguments

`fastafile`	List. A named list of DNA sequences (e.g. from `seqinr::read.fasta()`).
`N_filter`	Integer. Maximum number of N bases allowed in a sequence. Sequences with more N's than this threshold will be removed.

Details

This function is useful for quality control before phylogenetic analysis. Sequences with excessive ambiguous bases can affect the accuracy of distance calculations and tree construction.

Value

List. Filtered list of sequences.

Examples

```
test_seqs <- list(
    good_seq = "ATCGATCG",
    bad_seq  = "ATCGNNNNNATCG",
    okay_seq = "ATCGNNATCG"
)
filtered <- filter_N(test_seqs, N_filter = 3)
length(filtered)  # Should be 2
```

from_DNAStringSet *Convert a DNAStringSet to a named list of sequences*

Description

Converts a DNAStringSet object to a named list of character strings suitable for use with CGRphylo2 functions such as filter_N, create_meta, and parallelCGR.

Usage

```
from_DNAStringSet(dna)
```

Arguments

dna	A DNAStringSet object containing one or more DNA sequences.

Details

Bioconductor workflows commonly store DNA sequences as DNAStringSet objects. This function bridges that format with CGRphylo2's internal list representation, allowing seamless use of Bioconductor data structures in CGR-based phylogenetic analysis.

Value

A named list of character strings, one element per sequence.

Examples

```
if (requireNamespace("Biostrings", quietly = TRUE)) {
    dna <- Biostrings::DNAStringSet(c(
        seq1 = "ATCGATCGATCGATCG",
        seq2 = "GCTAGCTAGCTAGCTA"
    ))
    seqs <- from_DNAStringSet(dna)
    length(seqs)   # 2
    nchar(seqs[[1]])  # 16
}
```

matrixDistance *Calculate distance between two CGR frequency matrices*

Description

Computes the distance between two CGR frequency matrices using the specified distance metric.

Usage

```
matrixDistance(matrix1, matrix2, distance_type = "Euclidean")
```

Arguments

matrix1	Numeric matrix. First CGR frequency matrix.
matrix2	Numeric matrix. Second CGR frequency matrix.
distance_type	Character. Type of distance to calculate. Options are: • "Euclidean" (default): Standard Euclidean distance • "S_Euclidean": Squared Euclidean distance • "Manhattan": Manhattan (city block) distance

Details

This function calculates pairwise distances between CGR frequency matrices. The Euclidean distance is most commonly used, but Manhattan and squared Euclidean distances are also available for specific applications.

Value

Numeric. The calculated distance between the two matrices.

Examples

```
# Create two simple frequency matrices
matrix1 <- matrix(c(0.1, 0.2, 0.3, 0.4), ncol = 1)
matrix2 <- matrix(c(0.15, 0.25, 0.25, 0.35), ncol = 1)

# Calculate Euclidean distance
dist_euclidean <- matrixDistance(matrix1, matrix2,
                                 distance_type = "Euclidean")
print(dist_euclidean)

# Calculate Manhattan distance
dist_manhattan <- matrixDistance(matrix1, matrix2,
                                 distance_type = "Manhattan")
print(dist_manhattan)
```

---

parallelCGR *Parallel computation of CGR frequency matrices*

Description

Efficiently calculates CGR frequency matrices for multiple sequences using BiocParallel for cross-platform parallel processing.

Usage

```
parallelCGR(sequences, k_mer, len_trim, BPPARAM = BiocParallel::bpparam())
```

Arguments

sequences	List. A named list of DNA sequences (from filter_N or similar).
k_mer	Integer. The k-mer size for frequency calculation.
len_trim	Integer. Length to trim all sequences to.
BPPARAM	A BiocParallelParam object controlling parallel execution. Defaults to BiocParallel::bpparam() which uses the registered back-end (serial on Windows by default, multicore on Unix/macOS). Pass BiocParallel::SerialParam() to force single-core execution (required inside vignette chunks and unit tests).

Details

This function uses BiocParallel::bplapply to dispatch computation across available cores. The BPPARAM argument lets callers choose the back-end: MulticoreParam on Linux/macOS, SnowParam on Windows, or SerialParam for sequential execution.

Value

Named list. A list of frequency matrices, one per sequence.

Examples

```
seqs <- list(
    Seq1 = "ATCGATCGATCGATCGATCG",
    Seq2 = "GCTAGCTAGCTAGCTAGCTA"
)
freq_mats <- parallelCGR(seqs, k_mer = 2, len_trim = 20,
                         BPPARAM = BiocParallel::SerialParam())
cat("Matrices computed:", length(freq_mats), "\n")
```

---

plot_cgr *Plot CGR with customization options*

Description

A convenience wrapper function to create a CGR plot with sensible defaults.

Usage

```
plot_cgr(seq_index, main = NULL, cex = 0.2, pch = 4, col = "black", ...)
```

Arguments

seq_index	Integer. The index of the sequence to plot.
main	Character. Main title for the plot. If NULL, uses sequence name.
cex	Numeric. Point size (default 0.2 for dense sequences).
pch	Integer. Point character type (default 4 for crosses).
col	Character. Color for points (default "black").
...	Additional arguments passed to plot().

Value

NULL. Creates a plot as a side effect.

Examples

```
assign("fasta_filtered", list(seq1 = "ATCGATCGATCGATCGATCG"), envir = .GlobalEnv)
plot_cgr(1)
plot_cgr(1, main = "My Sequence", col = "blue", cex = 0.3)
rm(fasta_filtered, envir = .GlobalEnv)
```

---

saveMegaDistance *Save distance matrix in MEGA format*

Description

Exports a distance matrix to MEGA format for use with MEGA software for phylogenetic tree visualization and analysis.

Usage

```
saveMegaDistance(filename, distance_matrix)
```

Arguments

`filename`	Character. Output filename (typically with .meg extension).
`distance_matrix`	Numeric matrix. A square distance matrix with row and column names corresponding to sequence identifiers.

Details

MEGA (Molecular Evolutionary Genetics Analysis) is widely used for phylogenetic analysis. This function creates a distance matrix file compatible with MEGA's format specifications.

Value

NULL. Writes file as a side effect.

Examples

```
# Build a small symmetric distance matrix
dist_mat <- matrix(
    c(0.00, 0.12, 0.25,
      0.12, 0.00, 0.18,
      0.25, 0.18, 0.00),
    nrow = 3
)
rownames(dist_mat) <- colnames(dist_mat) <- c("Seq1", "Seq2", "Seq3")

# Save to a temporary file (no permanent files written during checks)
out <- tempfile(fileext = ".meg")
saveMegaDistance(out, dist_mat)
```

```
# Inspect the first few lines of the output
writeLines(readLines(out, n = 8))
```

`savePhylipDistance` *Save distance matrix in PHYLIP format*

Description

Exports a distance matrix to PHYLIP format for phylogenetic analysis with various bioinformatics tools.

Usage

```
savePhylipDistance(filename, distance_matrix, mode = "relaxed")
```

Arguments

`filename`	Character. Output filename (typically .txt or .phy extension).
`distance_matrix`	Numeric matrix. A square distance matrix with row and column names corresponding to sequence identifiers.
`mode`	Character. PHYLIP format mode: • "original": Original PHYLIP format (10 character limit for names) • "relaxed": Relaxed PHYLIP format (allows up to 250 characters)

Details

PHYLIP format is widely supported by phylogenetic software. The original format limits sequence names to 10 characters, while the relaxed format allows longer names. The relaxed format is recommended for modern applications.

Value

NULL. Writes file as a side effect.

References

PHYLIP format specification: http://www.phylo.org/index.php/help/relaxed_phylip

Examples

```
# Build a small symmetric distance matrix
dist_mat <- matrix(
    c(0.00, 0.12, 0.25,
      0.12, 0.00, 0.18,
      0.25, 0.18, 0.00),
    nrow = 3
)
rownames(dist_mat) <- colnames(dist_mat) <- c("Seq1", "Seq2", "Seq3")
```

```
# Save in relaxed PHYLIP format to a temporary file
out <- tempfile(fileext = ".phy")
savePhylipDistance(out, dist_mat, mode = "relaxed")

# Inspect output
writeLines(readLines(out))
```

Index

BiocParallelParam, *8*

calculateDistanceMatrix, 2
cgrplot, 3
create_meta, 4

DNAStringSet, *6*

filter_N, 5
from_DNAStringSet, 6

matrixDistance, 6

parallelCGR, 7
plot_cgr, 8

saveMegaDistance, 9
savePhylipDistance, 10