Package 'geneClusterPattern'

September 18, 2026

Type Package

Title Plot conserved gene pattern across multiple species

Version 0.99.2

Description An R package for constructing and comparing ortholog gene patterns around a focal gene across species. It enables the exploration of conserved gene neighborhoods (synteny) by organizing orthologous genes within a defined genomic window and aligning their relative positions across multiple genomes. It can retrieve homologous gene directly from Ensembl or it can incorporate user-provided ortholog group assignments.

License MIT + file LICENSE

biocViews Visualization

VignetteBuilder knitr

RoxygenNote 8.0.0

Encoding UTF-8

URL https://github.com/jianhong/geneClusterPattern

BugReports https://github.com/jianhong/geneClusterPattern/issues

Depends R (>= 4.6.0)

Imports BiocGenerics, Biostrings, pwalign, GenomeInfoDb, GenomicRanges, grDevices, IRanges, RANN, biomaRt, grid, methods, Seqinfo, stats, trackViewer, utils

Suggests org.Dr.eg.db, BiocStyle, knitr, rmarkdown, testthat, S4Vectors

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

git_branch devel

git_last_commit a879182

git_last_commit_date 2026-09-15

Repository Bioconductor 3.24

Date/Publication 2026-09-17

Author Jianhong Ou [aut, cre] (ORCID: <https://orcid.org/0000-0002-8652-2488>), Kenneth Poss [aut, fnd]

Maintainer Jianhong Ou <jou@morgridge.org>

Contents

`geneClusterPattern-package`

Plot conserved gene pattern cross multiple species

Description

Understanding the conservation of gene neighborhoods across species provides important insights into genome organization, functional linkage, and evolutionary processes. This package is developed to facilitate the systematic characterization of orthologous gene arrangements surrounding a focal gene across multiple species. Based on the orthology information, geneClusterPattern constructs an ordered representation of genes within a configurable genomic window upstream and downstream of the focal gene. The resulting ortholog gene pattern captures the relative arrangement of ortholog groups across species, enabling direct comparison of local genomic structure.

Author(s)

Maintainer: Jianhong Ou <jou@morgridge.org> (ORCID)

Authors:

- Jianhong Ou <jou@morgridge.org> (ORCID)
- Kenneth Poss <kposs@morgridge.org> [funder]

See Also

Useful links:

- https://github.com/jianhong/geneClusterPattern
- Report bugs at https://github.com/jianhong/geneClusterPattern/issues

`fromOrthogroups` *Read Orthologues results as a list This function is under-development.*

Description

Read Orthologues results as a list This function is under-development.

Usage

```
fromOrthogroups(orthogroups, annoGR_list, query_species = 1, split = ", ")
```

Arguments

`orthogroups`	A dataframe with the gene ids for each orthologues group
`annoGR_list`	A GRangesList object with the gene information. The names of each element should be contain all the gene ids saved in the orthogroups parameter. For each element, the 'gene_name' should be available in the metadata.
`query_species`	The species used as queryGR in getGeneCluster.
`split`	character vector used to split the gene ids in the dataframe of orthogroups parameter.

Value

a list of queryGR and homologsList used by getGeneCluster.

Examples

```
orthogroups <- read.delim(
     system.file('extdata/orthofinder/Orthogroups.tsv.gz',
                 package='geneClusterPattern'),
     row.names=1)
annoGR_list <- readRDS(
     system.file('extdata/orthofinder/grange.obj.rds',
                 package='geneClusterPattern'))
colnames(orthogroups) <- names(annoGR_list)
hres <- fromOrthogroups(orthogroups, annoGR_list)
```

`geneOrderScore` *Calculate the gene order score*

Description

Calculate the gene order score.

Usage

```
geneOrderScore(
  genesList,
  ids,
  ref,
  k = length(ids),
  max_gap = 1e+07,
  method = c("edit distance", "global alignment score", "spearman correlation",
   "kendall correlation", "non-random score", "pairs distance", "pairs direction"),
  output = c("score", "value matrix")
)
```

Arguments

genesList	A list of GRanges.
ids	IDs to be plotted. See getGeneCluster.
ref	The reference species.
k	The maximum number of nearest genes used in getGeneCluster.
max_gap	The maximal gaps from the first ID in 'ids'.
method	The method to calculate the gene order score. 'edit distance', the gene order score is the mean of edit (Levenshtein) distance of given ids from different species. Gene order score = k/length(ids)/(mean(adist)+1). 'global alignment score', the gene order score is the mean of normalized global alignment score. For global alignment score please refer pairwiseAlignment. The alignment score will be normalized by the reference self alignment score. 'spearman correlation', the score is the mean of absolute value of Spearman correlation of the genes appearance order. 'non-random score', the score is the mean of Jaccard index of 2-order and 3-order of ids for paired samples. 'pairs distance', the score is the mean of pairwise coordinates distance. Score = k/length(ids)/(mean(abs(coordinates distance))+1). 'pairs direction', the score is the mean of alignment of pairwise strand information. The higher the gene order score, the higher the conservation of gene order.
output	Output values

Value

the score or values matrix such as adist, or alignment scores.

Examples

```
fish <- readRDS(system.file('extdata', 'fish.rds',
package = 'geneClusterPattern'))
homologs <- readRDS(system.file('extdata', 'homologs.rds',
                                package = 'geneClusterPattern'))
queryGene <- 'inhbaa'
nearest10neighbors <- getGeneCluster(fish, queryGene, homologs, k=10)
genesList <- c(drerio=fish, homologs)[
c("hsapiens", "mmusculus", "drerio", "olatipes", "nfurzeri", "gaculeatus")]
# set zebrafish as reference
geneOrderScore(genesList, nearest10neighbors, ref='drerio')
# default first element is reference
geneOrderScore(genesList, nearest10neighbors, method='edit distance')
```

```
geneOrderScore(genesList, nearest10neighbors, method='global alignment score')
geneOrderScore(genesList, nearest10neighbors, method='spearman correlation')
geneOrderScore(genesList, nearest10neighbors, method='kendall correlation')
geneOrderScore(genesList, nearest10neighbors, method='non-random score')
geneOrderScore(genesList, nearest10neighbors, method='pairs distance')
geneOrderScore(genesList, nearest10neighbors, method='pairs direction')
```

`getGeneCluster` *calculate the gene cluster*

Description

A gene cluster is defined as the k nearest genes surrounding a reference gene according to the rank matrix. The rank matrix is a gene-by-species matrix, where each column contains the genomic ranks of genes for a given species.

Usage

```
getGeneCluster(queryGR, queryGeneName, homologsList, k = 10, radius = 0)
```

Arguments

`queryGR`	The query GRanges outputted by grangesFromEnsemblIDs.
`queryGeneName`	The center gene used to search.
`homologsList`	The homologs list outputted by getHomologGeneList.
`k`	The maximum number of nearest genes.
`radius`	The maximum number of gap genes for the search. If radius is set to greater than 0, k parameter will be ignored.

Value

The ids of the cluster for the center gene.

Examples

```
fish <- readRDS(system.file('extdata', 'fish.rds',
                            package = 'geneClusterPattern'))
homologs <- readRDS(system.file('extdata', 'homologs.rds',
                                package = 'geneClusterPattern'))
queryGene <- 'inhbaa'
nearestNeighbors <- getGeneCluster(fish, queryGene, homologs)
```

getGeneRank *Get the gene position orders*

Description

Get the gene position orders without considering the strand info.

Usage

```
getGeneRank(genes)
```

Arguments

genes	A GRanges object with names.

Value

A list with gene ranks named by the chromosome names.

Examples

```
fish <- readRDS(system.file('extdata', 'fish.rds',
                package = 'geneClusterPattern'))
ggr <- getGeneRank(fish)
```

getHomologGeneList *Get the gene positions for homologs*

Description

Get the gene positions without considering the strand info.

Usage

```
getHomologGeneList(species, mart, ensembl_gene_ids, ...)
```

Arguments

species	The target species for homologs query.
mart	object of class Mart.
ensembl_gene_ids	Ensembl gene ids.
...	Parameters could be used by useEnsembl except 'biomart'.

Value

A list of GRanges objects with gene positions.

Examples

```
if(interactive()){
  ## Ensembl server may not response
  library(biomaRt)
  fish <- readRDS(system.file('extdata', 'fish.rds',
                              package = 'geneClusterPattern'))
  ensembl_gene_ids <- names(fish[seqnames(fish)=='24'])
  species <- c('hsapiens', 'mmusculus')
  fish_mart <- useMart("ENSEMBL_MART_ENSEMBL", "drerio_gene_ensembl")
  homologs <- getHomologGeneList(species, fish_mart, ensembl_gene_ids)
}
```

`getHomologListForOrthoFinder` *Get the gene position orders for output of OrthoFinder*

Description

Get the gene position orders without considering the strand info.

Usage

```
getHomologListForOrthoFinder(orthologs, marts)
```

Arguments

`orthologs`	The output of orthologPairsFromOrthoFinder.
`marts`	a list of object of class Mart.

Value

A list of list with gene ranks named by the chromosome names.

Examples

```
path <- system.file('extdata/orthofinder/Danio_rerio.GRCz11.pep.all.tsv.gz',
                    package='geneClusterPattern')
orthologs <- orthologPairsFromOrthoFinder(path)
annoGR_list <- readRDS(
     system.file('extdata/orthofinder/grange.obj.rds',
                    package='geneClusterPattern'))
homologs <- getHomologListForOrthoFinder(orthologs, annoGR_list[-1])
```

grangesFromEnsemblIDs *create a GRanges object by given ensembl gene ids*

Description

Create a GRanges object by given ensembl gene ids

Usage

```
grangesFromEnsemblIDs(mart, ensembl_gene_ids)
```

Arguments

mart	object of class Mart.
ensembl_gene_ids	Ensembl gene ids.

Value

An object of GRanges

Examples

```
if(interactive()){
  library(biomaRt)
  mart <- useMart('ensembl', 'drerio_gene_ensembl')
  grangesFromEnsemblIDs(mart, c('ENSDARG00000008803', 'ENSDARG00000017038'))
}
```

guessSpecies *Guess the species*

Description

Guess the species for a given string.

Usage

```
guessSpecies(
  species,
  output = c("abbr", "scientific name", "mart", "taxid", "common name"),
  ...
)
```

Arguments

species	A character(1L) to guess.
output	Output type.
...	Parameter will be passed to useEnsembl.

Value

A character or Mart object.

Examples

```
if(interactive()){
  guessSpecies('zebrafish')
  guessSpecies('Japanese medaka')
  guessSpecies('stickleback')
  guessSpecies('hsapiens', output='scientific name')
  guessSpecies('human', output='taxid')
  guessSpecies('hg38', output='scientific name')
  guessSpecies('hg38', output='taxid')
  guessSpecies('GRCh38', output='scientific name')
  guessSpecies('GRCh38', output='taxid')
}
```

`homologsFromEnsemblIDs` *retrieve a homologs by given ensembl gene ids*

Description

This function retrieves the user specified homologs for given ensembl gene ids.

Usage

```
homologsFromEnsemblIDs(species, mart, ensembl_gene_ids, ...)
```

Arguments

`species`	The target species for homologs query.
`mart`	object of class Mart.
`ensembl_gene_ids`	Ensembl gene ids.
`...`	Parameters could be used by useEnsembl except 'biomart'.

Value

An object of GRanges

Examples

```
if(interactive()){
  library(biomaRt)
  mart <- useMart('ensembl', 'drerio_gene_ensembl')
  homologsFromEnsemblIDs(
    'human',
    mart,
   c('ENSDARG00000008803', 'ENSDARG00000017038'))
}
```

`orthologPairsFromOrthoFinder` *Read orthoFinder Orthologues results as a list Read orthoFinder Orthologues results as a list for each pair of orthologs.*

Description

Read orthoFinder Orthologues results as a list Read orthoFinder Orthologues results as a list for each pair of orthologs.

Usage

```
orthologPairsFromOrthoFinder(path)
```

Arguments

`path`	The path of the orthologues result file.

Value

a list of matrix with the paired orthologues.

Examples

```
path <- system.file('extdata/orthofinder/Danio_rerio.GRCz11.pep.all.tsv.gz',
                    package='geneClusterPattern')
orthologs <- orthologPairsFromOrthoFinder(path)
```

`plotGeneClusterPatterns` *plot gene patters for given ids*

Description

Plot the gene patters for given ids.

Usage

```
plotGeneClusterPatterns(
  genesList,
  ids,
  additionalID,
  max_gap = 1e+07,
  colors,
  maxNonEssential = 5,
  alignType = c("local", "global"),
  label_size = 0.5
)
```

Arguments

genesList	A list of GRanges.
ids	IDs to be plotted. See getGeneCluster.
additionalID	Other IDs should be included.
max_gap	The maximal gaps from the first ID in 'ids'.
colors	The colors for genes
maxNonEssential	The maximal number for non-essential genes between two essential genes.
alignType	Align the gene pattern by query gene ('local') or by the clusters ('global').
label_size	The label cex value. default is 0.5.

Value

invisible list of plot data.

Examples

```
fish <- readRDS(system.file('extdata', 'fish.rds',
                            package = 'geneClusterPattern'))
homologs <- readRDS(system.file('extdata', 'homologs.rds',
                                package = 'geneClusterPattern'))
queryGene <- 'inhbaa'
nearestNeighbors <- getGeneCluster(fish, queryGene, homologs)
genesList <- c(drerio=fish, homologs)
pgp <- plotGeneClusterPatterns(genesList, nearestNeighbors)
```

---

pruningSequences *Drop non-standard chromosomes Remove the unwanted chromosomes from a list of GRanges object.*

Description

Drop non-standard chromosomes Remove the unwanted chromosomes from a list of GRanges object.

Usage

```
pruningSequences(genesList, negativePattern = "[_.M]")
```

Arguments

genesList	A list of GRanges object
negativePattern	The pattern to match the seqnames that does not need.

Value

A list of filtered GRanges

Examples

```
library(GenomicRanges)
gr <- GRanges(rep(c("chr2", "chr3", "chrM"), 2), IRanges(1:6, 10))
pruningSequences(list(gr))
```

Index