Package 'geneviewer'

January 10, 2025

Type Package

Title Gene Cluster Visualizations

Version 0.1.10

Maintainer Niels van der Velden <n.s.j.vandervelden@gmail.com>

Description Provides tools for plotting gene clusters and transcripts by importing data from GenBank, FASTA, and GFF files. It performs BLASTP and MUMmer alignments [Altschul et al. (1990) <doi:10.1016/S0022-2836(05)80360-2>; Delcher et al. (1999) <doi:10.1093/nar/27.11.2369>] and displays results on gene arrow maps. Extensive customization options are available, including legends, labels, annotations, scales, colors, tooltips, and more.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Language en-US

RoxygenNote 7.3.1

URL https://github.com/nvelden/geneviewer

BugReports https://github.com/nvelden/geneviewer/issues

Depends R (>= 3.5.0)

Imports htmlwidgets, fontawesome, magrittr, dplyr, tidyr, rlang

Suggests knitr, rmarkdown, shiny, biomaRt, pwalign, Biostrings, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Niels van der Velden [aut, cre]

Repository CRAN

Date/Publication 2025-01-10 14:40:02 UTC

Contents

BRCA1_splice_variants *BRCA1 Splice Variants*

Description

This dataset contains detailed information on five different splice variants of the BRCA1 gene. It includes data on the types of transcript features (exon, intron, UTR), chromosomal positions, strand orientations, and transcript lengths.

Usage

```
BRCA1_splice_variants
```

Format

A data frame with 119 observations and 6 variables:

ensembl_transcript_id Unique Ensembl transcript identifiers. A character vector.

type Type of the transcript feature, indicating whether it is an "exon", or "UTR". A character vector.

start The start position of the feature on the chromosome. A numeric vector.

end The end position of the feature on the chromosome. A numeric vector.

strand The strand orientation of the feature, with -1 indicating the reverse strand. An integer vector.

transcript_length The total length of the transcript in base pairs. An integer vector.

Source

Derived from the Ensembl website

erythromycin_BlastP *Erythromycin BlastP results*

Description

This dataset contains detailed information on genes involved in the biosynthesis of Erythromycin, an antibiotic produced by the bacterium Saccharopolyspora erythraea and several homologous gene clusters identified by antiSMASH. It includes gene identifiers, chromosomal positions, orientations, and annotations regarding their products and functions, as well as similarity and identity scores from BlastP analysis.

Usage

```
erythromycin_BlastP
```

Format

A data frame with 148 observations and 16 variables:

protein_id Unique protein identifiers. A character vector.

region The chromosomal region of the gene, indicating start and end positions and strand. A character vector.

translation Amino acid sequence of the protein encoded by the gene. A character vector.

cluster Identifier of the gene cluster to which the gene belongs. A character vector.

strand The strand orientation ("forward" or "complement") of the gene. A character vector.

start The start position of the gene on the chromosome. A numeric vector.

end The end position of the gene on the chromosome. A numeric vector.

rowID A unique identifier for each row in the dataset. An integer vector.

identity The identity score from BlastP analysis, representing the percentage of identical matches. A numeric vector.

similarity The similarity score from BlastP analysis, often reflecting the functional or structural similarity. A numeric vector.

BlastP Reference to the protein_id after BlastP comparison, or NA if not applicable. A character vector.

score Score assigned based on the BlastP analysis, quantifying the match quality. A numeric vector.

Gene Gene name or identifier if available, otherwise NA. A character vector.

Position Formatted string indicating the gene's position and orientation on the chromosome. A character vector.

Product Description of the gene product. A character vector.

Functions Functional categorization of the gene. A character vector.

Source

Derived from BlastP analysis of Saccharopolyspora erythraea genes involved in Erythromycin production.

`erythromycin_cluster` *Erythromycin Gene Cluster Data*

Description

This dataset contains information about various genes involved in the biosynthesis of Erythromycin, an antibiotic produced by the bacterium Saccharopolyspora erythraea. It includes details such as gene identifiers, start and end positions of genes, and their associated functions and products.

Usage

```
erythromycin_cluster
```

Format

A data frame with 23 rows and 6 columns:

Identifiers Unique identifiers for the genes. A character vector.

Start Start positions of the genes on the chromosome. A numeric vector.

End End positions of the genes on the chromosome. A numeric vector.

Strand Strand orientation of the gene, either "+" or "-". A character vector.

Product Description of the gene products. A character vector.

Functions Functional categorization of the genes. A character vector.

---

gbk_features_to_df *Convert GenBank Features to Data Frame*

Description

This function processes a list of GenBank features (loaded by read_gbk()) and converts selected features into a data frame. It supports processing multiple gene clusters.

Usage

```
gbk_features_to_df(
  gbk_list,
  feature = "CDS",
  keys = NULL,
  process_region = TRUE
)
```

Arguments

gbk_list	A list of lists where each sub-list contains GenBank features for a specific gene cluster. Each sub-list is expected to have a named list of features, with each feature being a character vector.
feature	A string specifying the feature type to extract from each gene cluster's FEATURE list (e.g., "CDS" or "gene"). Defaults to "CDS".
keys	An optional vector of strings representing specific keys within the feature to retain in the final data frame. If 'NULL' (the default), all keys within the specified feature are included.
process_region	A boolean flag; when set to 'TRUE' (the default), special processing is performed on the 'region' key (if present) to extract 'strand', 'start', and 'end' information.

Value

A data frame where each row corresponds to a feature from the input list. The data frame includes a 'cluster' column indicating the source gbk file.

Examples

```
## Not run:
gbk <- read_gbk("path/to/genbank_file.gbk")
df <- gbk_features_to_df(gbk)

# To extract only specific keys within the "CDS" feature
df <- gbk_features_to_df(gbk, feature = "CDS", keys = c("gene", "region"))

# To disable special processing of the 'region' key
df <- gbk_features_to_df(gbk, process_region = FALSE)

## End(Not run)
```

`GC_align` *Align gene clusters*

Description

This function aligns clusters based on a specified gene id.

Usage

```
GC_align(GC_chart, id_column, id, align = "left")
```

Arguments

`GC_chart`	A chart object containing genomic data along with clustering information.
`id_column`	The name of the column that contains the gene identifiers.
`id`	The specific identifier of the gene to be aligned.
`align`	The alignment method for the gene. Valid values are "left", "right", or "center". Defaults to "left".

Value

The modified ‘GC_chart‘ object with updated genomic coordinates.

Examples

```
genes_data <- data.frame(
  start = c(10, 90, 130, 170, 210),
  end = c(40, 120, 160, 200, 240),
  name = c('Gene 1', 'Gene 2', 'Gene 3', 'Gene 4', 'Gene 5'),
  group = c('A', 'B', 'C', 'C', 'A'),
  cluster =  c(1, 1, 1, 2, 2)
)


GC_chart(genes_data, group = "group", cluster = "cluster", height = "150px") %>%
```

```
  GC_align("group", "A", align = "left") %>%
  GC_legend(FALSE)
```

`GC_annotation` *Add Annotations to a GC_chart*

Description

This function adds annotations to specified clusters within a GC chart. Annotations can be of various types and are positioned based on provided coordinates. The types of annotations available are: text, textMarker, line, arrow, symbol, rectangle, promoter, and terminator.

Usage

```
GC_annotation(GC_chart, type = "textAnnotation", cluster = NULL, ...)
```

Arguments

`GC_chart`	A GC chart object to which the annotations will be added.
`type`	Character vector specifying the type of annotations to add. The default is "text".
`cluster`	Numeric or character vector specifying the clusters to which annotations should be added.
`...`	Additional parameters for customization of annotations, depending on the type.

Value

Updated GC chart object with added annotations.

See Also

`GC_trackMouse`

Examples

```
genes_data <- data.frame(
  start = c(10, 50, 90, 130, 170, 210),
  end = c(40, 80, 120, 160, 200, 240),
  name = c('Gene 1', 'Gene 2', 'Gene 3', 'Gene 4', 'Gene 5', 'Gene 6'),
  group = c('A', 'A', 'B', 'B', 'A', 'C'),
  cluster = c(1, 1, 1, 2, 2, 2)
)

# Adding annotations to a GC chart
GC_chart(genes_data, cluster = "cluster", group = "group", height = "220px") %>%
  GC_annotation(
    type = "textMarker",
    cluster = 1,
```

```
  position = 24,
  text = "Gene 1",
  arrowSize = 8
) %>%
GC_annotation(
  type = "text",
  text = "feature 1",
  x = 91,
  y = 71
) %>%
GC_annotation(
  type = "symbol",
  symbol = "triangle",
  x = 95,
  y = 64,
  size = 10,
  rotation = 180
) %>%
GC_annotation(
  type = "terminator",
  x = 81
) %>%
GC_annotation(
  type = "promoter",
  x = 49
) %>%
# Convenience function to track mouse position on hoover
GC_trackMouse()
```

GC_chart *Create a GC Chart Visualization*

Description

Generates an interactive GC chart for genomic data.

Usage

```
GC_chart(
  data,
  start = "start",
  end = "end",
  cluster = NULL,
  group = NULL,
  strand = NULL,
  width = "100%",
  height = "400px",
  style = list(),
```

```
  elementId = NULL,
  save_button = TRUE
)
```

Arguments

data	Data frame containing genomic information or the file path to a folder containing '.gbk' files. When providing a file path, the data is loaded and processed into a data frame internally.
start	Column name that indicates start positions. Default is "start".
end	Column name that indicates end positions. Default is "end".
cluster	Optional column name used for clustering purposes. Default is NULL.
group	Column name used for gene grouping to influence color aesthetics.
strand	Optional column name indicating strand orientation. Acceptable values include 1, 'forward', 'sense', or '+' to represent the forward strand, and -1, 0, 'reverse', 'antisense', "complement" or '-' to represent the reverse strand. Default is NULL, meaning strand information is not used.
width	Width specification for the chart, such as '100%' or 500. Default is unspecified.
height	Height specification for the chart, such as '400px' or 300. Default is unspecified.
style	A list of CSS styles to be applied to the chart container. Each element of the list should be a valid CSS property-value pair. For example, list(backgroundColor = "white", border = "2px solid black"). Default is an empty list.
elementId	Optional identifier string for the widget. Default is NULL.
save_button	Logical, whether to include a save button. Default is TRUE.

Value

A GC chart widget.

Examples

```
genes_data <- data.frame(
  start = c(10, 50, 90, 130, 170, 210),
  end = c(40, 80, 120, 160, 200, 240),
  name = c('Gene 1', 'Gene 2', 'Gene 3', 'Gene 4', 'Gene 5', 'Gene 6'),
  group = c('A', 'A', 'B', 'B', 'A', 'C'),
  cluster = c(1, 1, 1, 2, 2, 2)
)
# Load from data.frame
GC_chart(genes_data, group = "group", cluster = "cluster", height = "200px") %>%
GC_labels("name")

# Load from folder containing .gbk files
# file_path <- "~/path/to/folder/"
# GC_chart(file_path) %>%
```

GC_cluster *Modify Cluster Settings*

Description

This function can switch prevention of gene overlap on, adjust the spacing between tracks and alter the styling of specified clusters.

Usage

```
GC_cluster(
  GC_chart,
  separate_strands = NULL,
  strand_spacing = NULL,
  prevent_gene_overlap = NULL,
  overlap_spacing = NULL,
  cluster = NULL,
  style = list(),
  ...
)
```

Arguments

GC_chart	The gene chart object to be modified.
separate_strands	Logical, indicating whether to vertically separate forward and reverse genes.
strand_spacing	Numeric, specifies the spacing between genes on different strands. Used only if 'separate_strands' is TRUE.
prevent_gene_overlap	Logical, indicating whether to vertically separate overlapping genes.
overlap_spacing	Numeric, specifies the spacing between overlapping genes Used only if 'prevent_gene_overlap' is TRUE.
cluster	Optional; used to specify which clusters in the chart should have tooltips.
style	A list of CSS styles to be applied to the gene track. Each element of the list should be a valid CSS property-value pair. For example, list(backgroundColor = "red", color = "white").
...	Additional arguments to be passed to the underlying functions.

Value

Returns the modified gene chart object.

Examples

```
genes_data <- data.frame(
  start = c(1, 10, 200, 220, 600),
  end = c(10, 150, 180, 400, 400),
  name = c('Gene 1', 'Gene 2', 'Gene 3', 'Gene 4', 'Gene 5'),
  group = c('A', 'A', 'A', 'A', 'A')
)


GC_chart(genes_data, group = "group", height = "150px") %>%
  GC_cluster(separate_strands=TRUE, strand_spacing = 0) %>%
  GC_legend(FALSE)
```

GC_clusterFooter *Add a Footer to Each Cluster in a GC Chart*

Description

This function allows you to add a footer to all or specific clusters within a GC chart. You can specify titles, subtitles, and control the display and styling.

Usage

```
GC_clusterFooter(
  GC_chart,
  title = NULL,
  subtitle = NULL,
  show = TRUE,
  cluster = NULL,
  subtitleFont = list(),
  titleFont = list(),
  ...
)
```

Arguments

GC_chart	A GC chart object that the footers will be added to.
title	Character vector or NULL. The title to be displayed in the footer. Multiple titles can be provided for different clusters, and they will be recycled if there are more clusters than titles. Default is NULL.
subtitle	Character vector or NULL. Subtitles to accompany the main titles. Default is NULL.
show	Logical vector. Controls the visibility of each footer. Default is TRUE for all clusters.
cluster	Numeric or character vector specifying which clusters should have footers added or updated. If NULL, all clusters will be updated. Default is NULL.

subtitleFont	List, styling options for the subtitle.
titleFont	List, styling options for the title.
...	Additional arguments for further customization of the footers.

Value

A GC chart object with updated footer settings for each specified cluster.

Examples

```
genes_data <- data.frame(
  start = c(10, 90, 130, 170, 210),
  end = c(40, 120, 160, 200, 240),
  name = c('Gene 1', 'Gene 3', 'Gene 4', 'Gene 5', 'Gene 6'),
  group = c('A', 'B', 'B', 'A', 'C'),
  cluster = c(1, 1, 2, 2, 2)
)

# Add a simple footer with subtitle to all clusters
GC_chart(genes_data, cluster = "cluster", group = "group") %>%
  GC_clusterFooter(
    title = "Cluster Footer",
    subtitle = "Cluster subtitle"
  )
# Add styling to the title and subtitle
GC_chart(genes_data, cluster = "cluster", group = "group") %>%
  GC_clusterFooter(
    title = "This is a footer",
    subtitle = "Subtitle for the footer",
    spacing = 15,
    show = TRUE,
    cluster = 1,
    x = 6,
    y = -20,
    align = "center", # left / right
    spacing = 12,
    titleFont = list(
      fontSize = "12px",
      fontWeight = "bold",
      fontFamily = "sans-serif",
      fill = "black",
      cursor = "default"
    ),
    subtitleFont = list(
      fill = "grey",
      fontSize = "10px",
      fontStyle = "normal",
      fontFamily = "sans-serif",
      cursor = "default"
    )
  )
```

`GC_clusterLabel` *Set or Update Cluster Labels for a GC Chart*

Description

This function allows you to set or update the labels for specified clusters within a GC chart. It provides flexibility in terms of the title, visibility, width, position, and other additional customization options.

Usage

```
GC_clusterLabel(
  GC_chart,
  title = NULL,
  show = TRUE,
  width = "100px",
  cluster = NULL,
  position = "left",
  wrapLabel = TRUE,
  wrapOptions = list(),
  ...
)
```

Arguments

`GC_chart`	A GC chart object.
`title`	Character vector. The title for the cluster label. Default is NULL.
`show`	Logical. Whether to show the cluster label. Default is TRUE.
`width`	Character. The width of the cluster label. Default is "100px".
`cluster`	Numeric or character vector. Clusters in the GC chart to update. Default is NULL.
`position`	Character. Position of the label, either "left" or "right". Default is "left".
`wrapLabel`	Logical. Indicates whether the label should be wrapped. Default is TRUE.
`wrapOptions`	List. Specifies the wrapping options. Default is an empty List.
`...`	Additional customization arguments for the cluster label, such as 'fontSize', 'fontStyle', 'fontWeight', 'fontFamily', 'cursor', etc.

Value

Updated GC chart with new or modified cluster labels.

Examples

```
genes_data <- data.frame(
  start = c(10, 90, 130, 170, 210),
  end = c(40, 120, 160, 200, 240),
  name = c('Gene 1', 'Gene 3', 'Gene 4', 'Gene 5', 'Gene 6'),
  group = c('A', 'B', 'B', 'A', 'C'),
  cluster = c(1, 1, 2, 2, 2)
)

# Set cluster labels
GC_chart(genes_data, cluster ="cluster", group = "group", height = "200px") %>%
GC_clusterLabel(title = unique(genes_data$cluster))

# Set label for a specific cluster
GC_chart(genes_data, cluster ="cluster", group = "group", height = "200px") %>%
GC_clusterLabel(title = "Cluster 1", cluster = 1)

# Style labels
GC_chart(genes_data, cluster ="cluster", group = "group", height = "200px") %>%
  GC_clusterLabel(
    title = c("Cluster 1", "Cluster 2"),
    width = "100px",
    x = 0,
    y = 0,
    position = "left",
    wrapLabel = TRUE,
    wrapOptions = list(
      dyAdjust = 0,
      lineHeightEms = 1.05,
      lineHeightSquishFactor =  1,
      splitOnHyphen =  TRUE,
      centreVertically = TRUE
     ),
    fontSize = "12px",
    fontStyle = "normal",
    fontWeight = "bold",
    fontFamily = "sans-serif",
    cursor = "default"
  )
```

GC_clusterTitle *Update cluster Title of a GC Chart Cluster*

Description

Modify the cluster title and subtitle of specified clusters within a GC chart and adjust the display settings.

Usage

```
GC_clusterTitle(
  GC_chart,
  title = NULL,
  subtitle = NULL,
  subtitleFont = list(),
  titleFont = list(),
  show = TRUE,
  height = NULL,
  cluster = NULL,
  ...
)
```

Arguments

GC_chart	A GC chart object.
title	Character vector. Titles to set for the clusters.
subtitle	Character vector. Subtitles to set for the clusters.
subtitleFont	List. Settings for the subtitle font.
titleFont	List. Settings for the title font.
show	Logical. Whether to display the title. Default is TRUE.
height	Character. Height for the title (e.g., "40px").
cluster	Numeric or character vector. Clusters in the GC chart to update.
...	Additional customization arguments for title and subtitle.

Value

Updated GC chart with new title settings.

Examples

```
genes_data <- data.frame(
  start = c(10, 50, 90, 130, 170, 210),
  end = c(40, 80, 120, 160, 200, 240),
  name = c('Gene 1', 'Gene 2', 'Gene 3', 'Gene 4', 'Gene 5', 'Gene 6'),
  group = c('A', 'A', 'B', 'B', 'A', 'C'),
  cluster = c(1, 1, 1, 2, 2, 2)
)

# Basic usage
GC_chart(genes_data, cluster ="cluster", group = "group", height = "400px") %>%
GC_labels("name") %>%
GC_clusterTitle(
  title = "Cluster 1 Data",
  subtitle = "Detailed View",
  show = TRUE,
  cluster = 1
)
```

```
# Customizing title style
GC_chart(genes_data, cluster = "cluster", group = "group", height = "400px") %>%
  GC_labels("name") %>%
  GC_clusterTitle(
    title = "Cluster 1 Data",
    subtitle = "Detailed View",
    show = TRUE,
    cluster = 1,
    x = 0,
    y = 5,
    align = "center",
    spacing = 20,
    titleFont = list(
      fontSize = "16px",
      fontStyle = "normal",
      fontWeight = "bold",
      textDecoration = "normal",
      fontFamily = "sans-serif",
      cursor = "default",
      fill = "black"
      # Any other CSS styles
    ),
    subtitleFont = list(
      fontSize = "14px",
      fontStyle = "normal",
      fontWeight = "bold",
      textDecoration = "normal",
      fontFamily = "sans-serif",
      cursor = "default",
      fill = "black"
      # Any other CSS styles
    )
  )
```

GC_color *Update Color Scheme in Gene Chart*

Description

This function updates the color scheme of the legend and genes in a gene chart.

Usage

```
GC_color(GC_chart, colorScheme = NULL, customColors = NULL)
```

Arguments

GC_chart	The gene chart object to be modified.

colorScheme	Optional; character or NULL, the name of a predefined color scheme to apply to the genes.Acceptable values include D3.js's built-in color schemes like "schemeCategory10", "schemeAccent", "schemeTableau10".
customColors	Either NULL, a list of color values, or a named list of color values.

Value

Returns the gene chart object with updated color settings for the genes.

Examples

```
genes_data <- data.frame(
  start = c(10, 50, 90, 130, 170, 210),
  end = c(40, 80, 120, 160, 200, 240),
  name = c('Gene 1', 'Gene 2', 'Gene 3', 'Gene 4', 'Gene 5', 'Gene 6'),
  group = c('A', 'A', 'B', 'B', 'A', 'C'),
  cluster = c(1, 1, 1, 2, 2, 2)
)

GC_chart(genes_data, cluster = "cluster", group = "group", height = "100px") %>%
  GC_color(colorScheme = "schemeCategory10")

GC_chart(genes_data, cluster = "cluster", group = "group", height = "100px") %>%
  GC_color(customColors = c("red", "orange", "green"))

GC_chart(genes_data, cluster = "cluster", group = "group", height = "100px") %>%
  GC_color(customColors = list(A = "yellow", B = "pink", C =  "purple"))
```

GC_coordinates *Modify Coordinates in a GC Chart*

Description

This function updates a GC chart by modifying the coordinates settings. It allows for showing or hiding tick values, applying custom tick values for the top and bottom axes, and supports several other customizations for specific or all clusters in the chart.

Usage

```
GC_coordinates(
  GC_chart,
  show = TRUE,
  tickValuesTop = NULL,
  tickValuesBottom = NULL,
  ticksFormat = NULL,
  tickStyle = list(),
  textStyle = list(),
```

```
  cluster = NULL,
  ...
)
```

Arguments

GC_chart	The GC chart object to be modified.
show	Logical, whether to show the tick values or not. Can be a single value or a vector.
tickValuesTop	Numeric vector or NULL, custom tick values to be used at the top of the cluster. If NULL, the default tick values are used.
tickValuesBottom	Numeric vector or NULL, custom tick values to be used at the bottom of the cluster. If NULL, the default tick values are used.
ticksFormat	Format for tick labels. Default is ",.0f".
tickStyle	List, styling options for the ticks.
textStyle	List, styling options for the text. the coordinate modifications. If NULL, applies to all clusters.
cluster	Numeric or character vector or NULL; specifies which clusters to generate coordinates for. If NULL, labels will be applied to all clusters. Default is NULL.
...	Additional arguments to be passed to the coordinate options.

Value

Returns the GC chart object with updated coordinates.

Examples

```
genes_data <- data.frame(
  start = c(10, 90, 130, 170, 210),
  end = c(40, 120, 160, 200, 240),
  name = c('Gene 1', 'Gene 2', 'Gene 3', 'Gene 4', 'Gene 5'),
  cluster = c(1, 1, 2, 2, 2)
)

# Add coordinates to all clusters
GC_chart(genes_data, cluster = "cluster", group = "name", height = "200px") %>%
GC_coordinates()

# Modify coordinates of a specific cluster
GC_chart(genes_data, cluster = "cluster", group = "name", height = "200px") %>%
GC_coordinates() %>%
GC_coordinates(
  cluster = 2,
  show = TRUE,
  tickValuesTop = c(130, 170, 210, 240),
  tickValuesBottom = c(160, 200),
  ticksFormat = ",.0f", # ".2s",
  rotate = -45,
  yPositionTop = 55,
```

```
  yPositionBottom = 45,
  overlapThreshold = 20,
  tickStyle = list(
  stroke = "black",
  strokeWidth = 1,
  lineLength = 6
  ),
  textStyle = list(
  fill = "black",
  fontSize = "12px",
  fontFamily = "Arial",
  cursor = "default"
  )
)
```

GC_genes *Modify Gene Characteristics within a Chart*

Description

This function updates a gene chart with specific characteristics for genes based on the given parameters. It can show/hide genes, apply a color scheme, assign custom colors, filter by cluster, and accept additional options.

Usage

```
GC_genes(
  GC_chart,
  group = NULL,
  marker = NULL,
  marker_size = NULL,
  show = TRUE,
  colorScheme = NULL,
  customColors = NULL,
  cluster = NULL,
  itemStyle = list(),
  ...
)
```

Arguments

GC_chart	The gene chart object to be modified.
group	Column name used for gene grouping to influence color aesthetics.
marker	Character or NULL, type of marker to represent genes on the chart. Allowed values are 'arrow', 'boxarrow', 'box', 'cbox', and 'rbox'.
marker_size	Character or NULL, size category of the marker ('small', 'medium', 'large').

show	Logical, whether to show the genes or not.
colorScheme	Character or NULL, the name of the color scheme to use.
customColors	List or NULL, custom colors to apply to the genes.
cluster	Numeric or character, the specific cluster to filter genes by.
itemStyle	List, a list of styles to apply to individual items in the chart.
...	Additional arguments to be passed to the gene options.

Value

Returns the modified gene chart object.

Examples

```
genes_data <- data.frame(
  start = c(10, 90, 130, 170, 210),
  end = c(40, 120, 160, 200, 240),
  name = c('Gene 1', 'Gene 3', 'Gene 4', 'Gene 5', 'Gene 6'),
  group = c('A', 'B', 'B', 'A', 'C'),
  cluster = c(1, 1, 2, 2, 2)
)

# Change the appearance of a specific gene cluster
GC_chart(genes_data, cluster = "cluster", group = "group", height = "200px") %>%
  GC_genes(
    group = "group",
    show = TRUE,
    marker = "arrow",
    marker_size = "medium",
    colorScheme = NULL, # One of D3.js build in colorSchemes
                        # (eg. "schemeCategory10",
                        # "schemeAccent", "schemeTableau10")
    customColors = NULL, # A vector of color names
    prevent_overlap = FALSE,
    gene_overlap_spacing = 40,
    cluster = 1, # Specify a specific cluster
    x = 1,
    y = 50,
    stroke = "black",
    strokeWidth = 1,
    arrowheadWidth = NULL,
    arrowheadHeight = NULL,
    arrowHeight = NULL,
    markerHeight = NULL # overwrites marker_size
   )

# Change the appearance of a specific gene
GC_chart(genes_data, cluster = "cluster", group = "group", height = "200px") %>%
GC_genes(
  cluster = 2,
  itemStyle = list(list(index = 2, fill = "red", stroke = "blue")
  )
```

```
)
```

GC_grid *Update Grid Display of a GC Chart Cluster*

Description

Modify the grid display of specified clusters within a GC chart. This function allows users to adjust the margins, width, and height of the grid for each cluster.

Usage

```
GC_grid(
  GC_chart,
  margin = NULL,
  width = NULL,
  height = NULL,
  direction = "column",
  cluster = NULL
)
```

Arguments

GC_chart	A GC chart object.
margin	A list specifying top, right, bottom, and left margins.
width	Numeric or character. Width of the grid. If numeric, will be considered as percentage.
height	Numeric. Height of the grid.
direction	Character. Layout direction of the grid, either "column" (default) for vertical or "row" for horizontal.
cluster	Numeric or character vector. Clusters in the GC chart to update.

Value

Updated GC chart with new grid display settings.

Examples

```
genes_data <- data.frame(
  start = c(10, 90, 130, 170, 210),
  end = c(40, 120, 160, 200, 240),
  name = c('Gene 1', 'Gene 3', 'Gene 4', 'Gene 5', 'Gene 6'),
  group = c('A', 'B', 'B', 'A', 'C'),
  cluster = c(1, 1, 2, 2, 2)
)
```

```
# Set Margin of clusters
GC_chart(genes_data, cluster ="cluster", group = "group", height = "200px") %>%
GC_grid(margin = list(left = "50px", right = "0px"))

# Set height of a specific cluster
GC_chart(genes_data, cluster ="cluster", group = "group", height = "200px") %>%
GC_grid(height = "120px", cluster = 2)
```

`GC_labels` *Add Labels to Each Cluster in a GC Chart*

Description

This function adds labels to each cluster within a GC chart. It provides the option to show or hide the labels and supports customization of label properties. It can automatically pick up group names as labels from the 'GC_chart' object if not provided.

Usage

```
GC_labels(
  GC_chart,
  label = NULL,
  show = TRUE,
  cluster = NULL,
  itemStyle = list(),
  ...
)
```

Arguments

`GC_chart`	A 'GC chart' object to which labels will be added.
`label`	Character vector, logical, or NULL. Specific labels for the clusters. If NULL and 'GC_chart' has group names, those will be used as labels.
`show`	Logical; controls the visibility of labels. Default is 'TRUE'.
`cluster`	Numeric or character vector or NULL; specifies which clusters should be labeled. If NULL, labels will be applied to all clusters. Default is NULL.
`itemStyle`	List, a list of styles to apply to individual items in the chart.
`...`	Additional arguments for further customization of the labels.

Value

A 'GC chart' object with updated label settings for each specified cluster.

Examples

```
genes_data <- data.frame(
  start = c(10, 90, 130, 170, 210),
  end = c(40, 120, 160, 200, 240),
  name = c('Gene 1', 'Gene 3', 'Gene 4', 'Gene 5', 'Gene 6'),
  group = c('A', 'B', 'B', 'A', 'C'),
  cluster = c(1, 1, 2, 2, 2)
)

# Add labels to all clusters
GC_chart(genes_data, cluster = "cluster", group = "group", height = "200px") %>%
GC_labels()

# Add labels and styling
GC_chart(genes_data, cluster = "cluster", group = "group", height = "200px") %>%
GC_labels(
 label = "group",
 show = TRUE,
 x = 0,
 y = 50,
 dy = "-1.2em",
 dx = "0em",
 rotate = 0,
 adjustLabels = TRUE, # Rotate labels to prevent overlap
 fontSize = "12px",
 fontStyle = "italic",
 fill = "black",
 fontFamily = "sans-serif",
 textAnchor = "middle",
 cursor = "default",
)

# Alter style of a specific label
GC_chart(genes_data, cluster = "cluster", group = "group", height = "200px") %>%
GC_labels(label = "group") %>%
GC_labels(
  cluster = 1,
  itemStyle = list(
    list(index = 0, fill = "red", fontSize = "14px", fontWeight = "bold")
  )
)
```

GC_legend *Set Legend for a Gene Chart*

Description

This function configures the legend for a gene chart. It allows toggling the legend's visibility, setting a background color, and assigning custom labels for the legend entries. The function can also handle additional customizations through various options.

Usage

```
GC_legend(
  GC_chart,
  group = NULL,
  show = TRUE,
  backgroundColor = "#0000",
  order = list(),
  position = "bottom",
  style = list(),
  legendOptions = list(),
  legendTextOptions = list(),
  ...
)
```

Arguments

GC_chart	The gene chart object to be modified.
group	Optional; character or NULL, specifies the groups to include in the legend. If NULL, groups are taken from the 'group' attribute of the 'GC_chart' object.
show	Logical, specifies whether to display the legend.
backgroundColor	String, the background color of the legend.
order	Optional; list, specifies the order of the legend labels.
position	Character. Position of the legend, either "top" or "bottom". Default is "bottom".
style	A list of CSS styles to be applied to the chart container. Each element of the list should be a valid CSS property-value pair. For example, list(backgroundColor = "white", border = "2px solid black"). Default is an empty list.
legendOptions	List, additional options for the legend.
legendTextOptions	List, additional text options for the legend.
...	Additional arguments to be passed to the legend configuration.

Value

Returns the modified gene chart object with the legend configured.

Examples

```
genes_data <- data.frame(
  start = c(10, 90, 130, 170, 210),
  end = c(40, 120, 160, 200, 240),
  name = c('Gene 1', 'Gene 3', 'Gene 4', 'Gene 5', 'Gene 6'),
  group = c('A', 'B', 'B', 'A', 'C'),
  cluster = c('A', 'A', 'A', 'B', 'B')
)

# Customize legend
```

```
GC_chart(genes_data, cluster = "cluster", group = "group", height = "200px") %>%
  GC_legend(
    position = "top", #bottom
    orientation = "horizontal", #vertical
    x = 0,
    y = 0,
    adjustHeight = TRUE,
    backgroundColor = "#0000",
    order = list(),
    positions = "bottom",
    style = list(
      backgroundColor = "#0000"
      # Any other CSS styles
    ),
    legendOptions = list(
      cursor = "pointer",
      colorScheme = NULL,
      customColors = NULL # c("red", "green", "orange")
      # Additional styles
    ),
    legendTextOptions = list(
      cursor = "pointer",
      textAnchor = "start",
      dy = ".35em",
      fontSize = "12px",
      fontFamily = "sans-serif",
      fill = "black"
      # Additional styles
    )
  )
```

GC_links *Add Links to GC Chart*

Description

Add links generated by ‘get_links‘ to a ‘GC_chart‘ object. Links are added to the graph by their respective rowIDs.

Usage

```
GC_links(
  GC_chart,
  group = NULL,
  data = NULL,
  value1 = NULL,
  value2 = NULL,
  cluster = NULL,
  curve = TRUE,
```

```
  measure = "identity",
  show_links = TRUE,
  label = TRUE,
  normal_color = "#969696",
  inverted_color = "#d62728",
  use_group_colors = FALSE,
  color_bar = TRUE,
  colorBarOptions = list(),
  linkWidth = NULL,
  linkStyle = list(),
  labelStyle = list(),
  ...
)
```

Arguments

GC_chart	Gene chart object.
group	The name of the column in the data to create value pairs from.
data	data.frame containing linking data.
value1	Optional vector of group values to generate links for.
value2	Optional vector of group values to generate links for.
cluster	Numeric or character vector or NULL; specifies which clusters.
curve	Logical; if 'TRUE', links are curved, otherwise straight.
measure	Character; specifies which measure to use for link color intensity. Should be "identity", "similarity", or "none".
show_links	Logical; if 'TRUE', links are shown, otherwise hidden.
label	Logical; if 'TRUE', shows measure labels on the links, otherwise hidden.
normal_color	Color for the links in their normal state.
inverted_color	Color for inverted links.
use_group_colors	Logical; if 'TRUE', color links by group.
color_bar	Logical; if 'TRUE', the color bar is displayed.
colorBarOptions	List of options to customize the color bar appearance.
linkWidth	Numeric; specifies the width of the links. A value of '1' represents full width, and '0' represents no width.
linkStyle	A list of CSS styles to apply to the links.
labelStyle	A list of CSS styles specifically for the labels.
...	Additional arguments passed to the links.

Value

Modified 'GC_chart' object with added links.

See Also

* [get_links()]

Examples

```
# Add links between all groups in each cluster
genes_data <- data.frame(
  start = c(10, 90, 130, 170, 240, 250, 300, 340, 380, 420),
  end = c(40, 120, 160, 200, 210, 270, 330, 370, 410, 450),
  name = c('Gene 1', 'Gene 2', 'Gene 3', 'Gene 4', 'Gene 5',
           'Gene 6', 'Gene 7', 'Gene 8', 'Gene 9', 'Gene 10'),
  group = c('A', 'B', 'C', 'A', 'B', 'C', 'A', 'B', 'C', 'D'),
  identity = c(NA, NA, NA, 50, 40, 100, 60, 65, 20, NA),
  similarity = c(NA, NA, NA, 40, 30, 90, 50, 55, 10, NA),
  cluster = c(1, 1, 1, 2, 2, 2, 3, 3, 3, 3)
)
GC_chart(genes_data,
         cluster = "cluster",
         height = "200px") %>%
  GC_links(
    group = "group",
    value1 = "A",
    value2 = "B",
    measure = "identity",
    label = FALSE) %>%
  GC_labels(label = "group")

# Add links between group A of cluster 1 and A and B of cluster 2
GC_chart(genes_data,
         cluster = "cluster",
         height = "200px") %>%
         GC_labels(label = "group") %>%
         GC_links(group = "group",
                  value1 = c("A", "A"),
                  value2 = c("B", "A"),
                  label = FALSE,
                  cluster = c(1,2))

# Style links and color bar
GC_chart(genes_data,
         cluster = "cluster",
         height = "200px"
) %>%
  GC_links(
    group = "group",
    data = NULL,
    curve = TRUE,
    measure = "identity",
    show_links = TRUE,
    label = TRUE,
    normal_color = "#1f77b4",
    inverted_color = "#d62728",
```

```
  use_group_colors = FALSE,
  color_bar = TRUE,
  colorBarOptions = list(
    x = 0,
    y = 24,
    width = 10,
    height = 60,
    labelOptions = list(
      fontSize = 8,
      xOffset = 2,
      yOffset = 0
      # Any other CSS style
    ),
    titleOptions = list(
      fontSize = 10,
      xOffset = 2,
      yOffset = 0
      # Any other CSS style
    ),
    barOptions = list(
      stroke = "#000",
      strokeWidth = 0.5,
      opacity = 1
      # Any other CSS style
    )
  ),
  linkWidth = 1,
  linkStyle = list(
    stroke = "black",
    strokeWidth = 0.5,
    fillOpacity = 0.4
    # Any other CSS style
  ),
  labelStyle = list(
    fontSize = "8px"
    # Any other CSS style
  )
) %>%
GC_labels(label = "group", cluster = 1) %>%
GC_clusterLabel()
```

GC_normalize *Normalize Gene Clusters in a Genomic Chart*

Description

This function normalizes the genomic coordinates of a set of genes within a cluster, ensuring that the genes are evenly spaced along the entire range of the cluster. The function allows for the option to preserve the original gene lengths and to introduce customized gaps between the genes.

Usage

```
GC_normalize(
  GC_chart,
  group = NULL,
  cluster = NULL,
  normalize = TRUE,
  preserve_gene_length = TRUE,
  gap = NULL,
  custom_gaps = list()
)
```

Arguments

GC_chart	A chart object containing genomic data and clustering information. The chart object must include a 'series' component with data for each cluster.
group	Column name containing gene names to apply custom gaps to.
cluster	A vector of cluster names to be normalized. If 'NULL', all clusters in the 'GC_chart' will be normalized.
normalize	A logical value indicating whether to normalize the gene positions within each cluster. Default is 'TRUE'.
preserve_gene_length	A logical vector indicating whether to preserve the original gene lengths for each cluster. If a single value is provided, it is recycled for all clusters.
gap	A numeric vector specifying the proportion of the total length to be used as the gap between genes in each cluster. The value must be between 0 and 1. If a single value is provided, it is recycled for all clusters. If 'NULL', the gaps are calculated based on the actual spacing between the genes in the original data.
custom_gaps	A named list where each name corresponds to a gene name and each value is a numeric factor by which to adjust the gap after that gene.

Value

The modified 'GC_chart' object with normalized gene coordinates for the specified clusters.

Examples

```
genes_data <- data.frame(
  start = c(10, 90, 130, 10, 90, 130),
  end = c(40, 120, 160, 40, 120, 160),
  name = c('Gene 1', 'Gene 2', 'Gene 3','Gene 1', 'Gene 2', 'Gene 3'),
  group = c('A', 'B', 'C', 'A', 'B', 'C'),
  cluster =  c(1, 1, 1, 2, 2, 2)
)

GC_chart(genes_data, group = "group", cluster = "cluster", height = "150px") %>%
  GC_normalize(
    cluster = 2,
    preserve_gene_length = TRUE,
```

```
    gap = NULL
)
```

GC_scale *Update Scale of a GC Chart Cluster*

Description

Modify the scale settings for specified clusters within a GC chart.

Usage

```
GC_scale(
  GC_chart,
  cluster = NULL,
  start = NULL,
  end = NULL,
  padding = 2,
  hidden = FALSE,
  breaks = list(),
  tickValues = NULL,
  reverse = FALSE,
  axis_position = "bottom",
  axis_type = "position",
  y = NULL,
  scale_breaks = FALSE,
  scale_break_threshold = 20,
  scale_break_padding = 1,
  ticksCount = 10,
  ticksFormat = NULL,
  tickStyle = list(),
  textStyle = list(),
  lineStyle = list(),
  ...
)
```

Arguments

GC_chart	A GC chart object.
cluster	Numeric or character vector specifying clusters in the GC chart to update.
start	Numeric vector indicating the starting point for the scale. Default is NULL.
end	Numeric vector indicating the end point for the scale. Default is NULL.
padding	Numeric value or percentage string indicating the padding on either side of the scale. The value can be a number or a string in the format of '2%'. Default is 2.
hidden	Logical flag indicating whether the axis is hidden. Default is FALSE.

breaks	List specifying settings for the scale breaks. Default is an empty list ().
tickValues	Numeric vector or NULL, custom tick values to be used at the top of the cluster. If NULL, the default tick values are used.
reverse	Logical flag indicating whether to reverse the scale for the corresponding cluster. Default is FALSE.
axis_position	Character string indicating the type of the axis ('top' or bottom'). Default is 'bottom'.
axis_type	Character string indicating the type of the axis ('position' or 'range'). Default is 'position'.
y	Numeric value from 1 to 100 indicating the y-position of the x-axis. Default is NULL.
scale_breaks	Logical flag indicating if scale breaks should be employed. Default is FALSE.
scale_break_threshold	Numeric value indicating the threshold percentage for determining scale breaks. Default is 20.
scale_break_padding	Numeric value indicating the padding on either side of a scale break. Default is 1.
ticksCount	Numeric value indicating the number of ticks on the scale. Default is 10.
ticksFormat	Format for tick labels; depends on axis_type, defaulting to ",.0f" for 'position' or ".2s" for 'range' when NULL.
tickStyle	List specifying the style for the ticks.
textStyle	List specifying the style for the tick text.
lineStyle	List specifying the style for the axis line.
...	Additional arguments for scale settings.

Value

Updated GC chart with new scale settings.

Examples

```
genes_data <- data.frame(
  start = c(100, 1000, 2000),
  end = c(150, 1500, 2500),
  name = c('Gene 4', 'Gene 5', 'Gene 6'),
  group = c('B', 'A', 'C'),
  cluster = c(2, 2, 2)
)
#Example usage with all custom options
GC_chart(genes_data, cluster = "cluster", group = "group", height = "200px") %>%
  GC_scale(
    start = 1,
    end = 2600,
    padding = 2,
    hidden = FALSE,
```

```
  reverse = FALSE,
  axis_position = "bottom",
  # breaks = list(
  #  list(start = 160, end = 900),
  #  list(start = 1600, end = 1900)
  # ),
  # tickValues = c(1, 2600),
  scale_breaks = TRUE,
  scale_break_threshold = 20,
  scale_break_padding = 1,
  ticksCount = 10,
  ticksFormat = ",.0f",
  y = NULL,
  tickStyle =
    list(
      stroke = "grey",
      strokeWidth = 1,
      lineLength = 6
      # Any other CSS styles
    ),
  textStyle =
    list(
      fill = "black",
      fontSize = "10px",
      fontFamily = "Arial",
      cursor = "default"
      # Any other CSS styles
    ),
  lineStyle = list(
    stroke = "grey",
    strokeWidth = 1
    # Any other CSS styles
  )
) %>%
GC_legend(FALSE)
```

GC_scaleBar *Update Scale Bar of a GC Chart Cluster*

Description

Modify the scale bar settings for specified clusters within a GC chart.

Usage

```
GC_scaleBar(
  GC_chart,
  show = TRUE,
  cluster = NULL,
  scaleBarLineStyle = list(),
```

```
  scaleBarTickStyle = list(),
  labelStyle = list(),
  ...
)
```

Arguments

GC_chart	A GC chart object.
show	Logical. Whether to show the scale bar.
cluster	Numeric or character vector. Clusters in the GC chart to update.
scaleBarLineStyle	List of style options for the scale bar line.
scaleBarTickStyle	List of style options for the scale bar ticks.
labelStyle	List of style options for the scale bar label.
...	Additional arguments for scale bar settings.

Value

Updated GC chart with new scale bar settings.

Examples

```
genes_data <- data.frame(
 start = c(1000, 9000, 13000, 17000, 21000),
 end = c(4000, 12000, 16000, 20000, 24000),
 name = c('Gene 1', 'Gene 3', 'Gene 4', 'Gene 5', 'Gene 6'),
 group = c('A', 'B', 'B', 'A', 'C'),
 cluster = c(1, 1, 2, 2, 2)
)

# Set scale bar for individual clusters
GC_chart(genes_data, cluster ="cluster", group = "group") %>%
GC_scaleBar(cluster = 1, title = "1 kb", scaleBarUnit = 1000) %>%
GC_scaleBar(cluster = 2, title = "2 kb", scaleBarUnit = 2000)

# Style scale bar
GC_chart(genes_data, cluster ="cluster", group = "group", height = "400px") %>%
  GC_scaleBar(
    title = "1kb",
    scaleBarUnit = 1000,
    cluster = NULL,
    x = 0,
    y = 50,
    labelStyle = list(
      labelPosition =  "left",
      fontSize = "10px",
      fontFamily = "sans-serif",
      fill = "red", # Text color
      cursor = "default"
```

```
    # Any other CSS style for the label
  ),
  textPadding = -2,
  scaleBarLineStyle = list(
    stroke = "black",
    strokeWidth = 1
    # Any other CSS style for the line
  ),
  scaleBarTickStyle = list(
    stroke = "black",
    strokeWidth = 1
    # Any other CSS style for the tick
  )
)
```

GC_sequence *Update Sequence Display of a GC Chart Cluster*

Description

Modify the sequence display and break markers of specified clusters within a GC chart.

Usage

```
GC_sequence(
  GC_chart,
  show = TRUE,
  cluster = NULL,
  y = 50,
  sequenceStyle = list(),
  markerStyle = list(),
  ...
)
```

Arguments

GC_chart	A GC chart object.
show	Logical, whether to display the sequence (default is TRUE).
cluster	Numeric or character vector specifying clusters to update.
y	Vertical position of the sequence line (default is 50).
sequenceStyle	A list of styling options for the sequence line.
markerStyle	A list of styling options for the sequence break markers.
...	Additional customization arguments for sequence display.

Details

This function allows customization of the sequence line and break markers in a GC chart. It offers options to adjust the sequence line (‘sequenceStyle‘) and break markers (‘markerStyle‘). The ‘y‘ parameter can be used to set the vertical position of the sequence.

Value

An updated GC chart with modified sequence display settings.

Examples

```
genes_data <- data.frame(
  start = c(10, 90, 130, 170, 210),
  end = c(40, 120, 160, 200, 240),
  name = c('Gene 1', 'Gene 3', 'Gene 4', 'Gene 5', 'Gene 6'),
  group = c('A', 'B', 'B', 'A', 'C'),
  cluster = c(1, 1, 2, 2, 2)
)

# Basic usage
GC_chart(genes_data, cluster ="cluster", group = "group", height = "200px") %>%
GC_labels("name") %>%
GC_sequence(show = TRUE, y = 50, cluster = NULL)

# Customize sequence and marker styles
GC_chart(genes_data, cluster="cluster", group = "group", height = "200px") %>%
  GC_scale(hidden = TRUE, scale_breaks = TRUE) %>%
  GC_sequence(
    start = NULL,
    end = NULL,
    sequenceStyle = list(
      stroke = "blue",
      strokeWidth = 1
      # Any other CSS style
    ),
    markerStyle = list(
      stroke = "blue",
      strokeWidth = 1,
      gap = 3,
      tiltAmount = 5
      # Any other CSS style
    )
  ) %>%
  GC_legend(FALSE)
```

GC_title *Add Title to a GC_chart*

Description

Modify the cluster title and subtitle of specified clusters within a GC chart and adjust the display settings.

Usage

```
GC_title(
  GC_chart,
  title = NULL,
  subtitle = NULL,
  style = list(),
  subtitleFont = list(),
  titleFont = list(),
  show = TRUE,
  height = NULL,
  ...
)
```

Arguments

GC_chart	A GC chart object.
title	Character vector. Titles to set for the clusters.
subtitle	Character vector. Subtitles to set for the clusters.
style	A list of CSS styles to be applied to the chart container. Each element of the list should be a valid CSS property-value pair. For example, list(backgroundColor = "white", border = "2px solid black"). Default is an empty list.
subtitleFont	List. Settings for the subtitle font.
titleFont	List. Settings for the title font.
show	Logical. Whether to display the title. Default is TRUE.
height	Character. Height for the title (e.g., "50px").
...	Additional customization arguments for title and subtitle.

Value

Updated GC chart with new title settings.

Examples

```
genes_data <- data.frame(
  start = c(10, 50, 90, 130, 170, 210),
  end = c(40, 80, 120, 160, 200, 240),
  name = c('Gene 1', 'Gene 2', 'Gene 3', 'Gene 4', 'Gene 5', 'Gene 6'),
  group = c('A', 'A', 'B', 'B', 'A', 'C'),
  cluster = c(1, 1, 1, 2, 2, 2)
)

# Basic usage
```

```
GC_chart(genes_data, cluster ="cluster", group = "group", height = "400px") %>%
GC_labels("name") %>%
GC_title(
  title = "Cluster 1 Data",
  subtitle = "Detailed View",
  show = TRUE
)

# Customizing title style
GC_chart(genes_data, cluster = "cluster", group = "group", height = "500px") %>%
  GC_labels("name") %>%
  GC_title(
    title = "Cluster 1 Data",
    subtitle = "Detailed View",
    show = TRUE,
    height = "50px",
    cluster = 1,
    x = 0,
    y = 25, # height / 2
    align = "center",
    spacing = 20,
    style = list(
      backgroundColor = "#0000"
      # Any other CSS styles
    ),
    titleFont = list(
      fontSize = "16px",
      fontStyle = "normal",
      fontWeight = "bold",
      textDecoration = "normal",
      fontFamily = "sans-serif",
      cursor = "default",
      fill = "black"
      # Any other CSS styles
    ),
    subtitleFont = list(
      fontSize = "14px",
      fontStyle = "normal",
      fontWeight = "bold",
      textDecoration = "normal",
      fontFamily = "sans-serif",
      cursor = "default",
      fill = "black"
      # Any other CSS styles
    )
  )
```

GC_tooltip *Set Tooltip for a Gene Chart*

Description

This function configures the tooltip for a gene chart.

Usage

```
GC_tooltip(
  GC_chart,
  formatter = "<b>Start:</b> {start}<br><b>End:</b> {end}",
  show = TRUE,
  cluster = NULL,
  ...
)
```

Arguments

GC_chart	The gene chart object to be modified.
formatter	A character string defining the HTML content of the tooltip. It can include placeholders like {start} and {end} which will be replaced by actual data values. The default value shows start and end data.
show	Logical, whether to display the tooltip or not.
cluster	Optional; used to specify which clusters in the chart should have tooltips.
...	Additional arguments that can be used to further customize the tooltip.

Value

Returns the gene chart object with the tooltip configured.

Examples

```
# Set tooltip
genes_data <- data.frame(
  start = c(10, 90, 130, 170, 210),
  end = c(40, 120, 160, 200, 240),
  name = c('Gene 1', 'Gene 2', 'Gene 3', 'Gene 4', 'Gene 5'),
  group = c('A', 'B', 'B', 'A', 'C')
)

# Add tooltips to the gene chart
GC_chart(genes_data, group = "group", height = "200px") %>%
GC_tooltip(formatter = " <b>Start:</b> {start}<br><b>end:</b> {end}")
```

GC_trackMouse *Track Mouse Movement in a GC_chart*

Description

This function enables or disables mouse tracking on specified clusters within a GC chart. When enabled, the x and y coordinates of the mouse are displayed which can be used to place annotations.

Usage

```
GC_trackMouse(GC_chart, show = TRUE, cluster = NULL)
```

Arguments

GC_chart	A GC chart object to which the annotations will be added.
show	Logical, specifies whether to track the mouse or not.
cluster	Numeric or character vector specifying the clusters to which annotations should be added.

Value

Updated GC chart object with mouse tracking settings.

See Also

GC_annotation

Examples

```
genes_data <- data.frame(
  start = c(10, 50, 90, 130, 170, 210),
  end = c(40, 80, 120, 160, 200, 240),
  name = c('Gene 1', 'Gene 2', 'Gene 3', 'Gene 4', 'Gene 5', 'Gene 6'),
  group = c('A', 'A', 'B', 'B', 'A', 'C'),
  cluster = c(1, 1, 1, 2, 2, 2)
)

# Enable mouse tracking
GC_chart(genes_data, cluster = "cluster", group = "group", height = "220px") %>%
GC_trackMouse()
```

GC_transcript *Modify Transcript Characteristics within a Chart*

Description

This function updates the chart with specific characteristics for transcripts based on the given parameters. It can show/hide transcripts, apply a color scheme, assign custom colors, filter by cluster, and accept additional options.

Usage

```
GC_transcript(
  GC_chart,
  transcript = NULL,
  type = NULL,
  strand = NULL,
  group = NULL,
  selection = NULL,
  show = TRUE,
  colorScheme = NULL,
  customColors = NULL,
  styleExons = list(),
  styleIntrons = list(),
  styleUTRs = list(),
  itemStyleExons = list(),
  itemStyleIntrons = list(),
  itemStyleUTRs = list(),
  labelOptions = list(),
  ...
)
```

Arguments

GC_chart	The chart object to be modified.
transcript	Optional column name used for clustering transcript data. Default is NULL.
type	Column name identifying the feature type ("UTR", "exon"). Default is NULL.
strand	Optional column name indicating strand orientation. Acceptable values include 1, 'forward', 'sense', or '+' to represent the forward strand, and -1, 0, 'reverse', 'antisense', "complement" or '-' to represent the reverse strand. Default is NULL, meaning strand information is not used.
group	Optional column name used for transcript grouping to influence color aesthetics.
selection	Numeric or character, the specific transcript to filter transcripts by.
show	Logical, whether to show the transcripts or not.
colorScheme	Character or NULL, the name of the color scheme to use.
customColors	List or NULL, custom colors to apply to the genes.

styleExons	List, styles to apply to exons in the chart.
styleIntrons	List, styles to apply to introns in the chart.
styleUTRs	List, styles to apply to UTRs in the chart.
itemStyleExons	List, a list of styles to apply to individual exons in the chart.
itemStyleIntrons	List, a list of styles to apply to individual introns in the chart.
itemStyleUTRs	List, a list of styles to apply to individual UTRs in the chart.
labelOptions	List, options for styling labels such as font size, color, and position.
...	Additional arguments to be passed to the gene options.

Value

Returns the modified gene chart object.

Examples

```
transcript_data <- data.frame(
  transcript = c("transcript1", "transcript1", "transcript1", "transcript1",
                 "transcript2", "transcript2", "transcript2"),
  type = c("5_utr", "exon", "exon", "3_utr",
           "5_utr", "exon", "3_utr"),
  start = c(1, 101, 201, 301,
            1, 101, 301),
  end = c(50, 150, 250, 350,
          50, 150, 350),
  strand = rep("forward", 7)
)

# All default transcript settings
GC_chart(
  transcript_data,
  start = "start",
  end = "end",
  height = "200px"
) %>%
  GC_transcript(
    transcript = "transcript",
    strand = "strand",
    type = "type",
    group = NULL,
    show = TRUE,
    selection = NULL,
    colorScheme = NULL,
    customColors = NULL,
    styleExons = list(
      show = TRUE,
      strokeWidth = 0,
      cursor = "default",
      marker = "box",
      markerSize = "medium",
```

```
      arrowheadWidth = NULL,
      arrowheadHeight = NULL,
      markerHeight = NULL,
      cornerRadius = NULL
      # Any other CSS style
    ),
    styleIntrons = list(
      show = TRUE,
      strokeWidth = 1,
      fill = "none",
      cursor = "default",
      marker = "intron",
      markerSize = "medium",
      arrowheadWidth = NULL,
      arrowheadHeight = NULL,
      markerHeight = NULL,
      cornerRadius = NULL
      # Any other CSS style
    ),
    styleUTRs = list(
      show = TRUE,
      fontSize = "10px",
      fontStyle = "normal",
      fontFamily = "sans-serif",
      cursor = "default",
      color = "black",
      fill = "#FFF",
      strokeWidth = 1,
      marker = "box",
      markerSize = "medium",
      arrowheadWidth = NULL,
      arrowheadHeight = NULL,
      markerHeight = NULL,
      cornerRadius = NULL
      # Any other CSS style
    ),
    labelOptions = list(
      show = TRUE,
      xOffset = 2,
      yOffset = 0,
      fontSize = "12px",
      fontStyle = "normal",
      fontWeight = "normal",
      fontFamily = "sans-serif",
      cursor = "default",
      color = "black"
    ),
    itemStyleExons = list(),
    itemStyleIntrons = list(),
    itemStyleUTRs = list()
  )
# Change the appearance of a specific intron
GC_chart(transcript_data,
```

```
        start = "start",
        end = "end",
        height = "200px"
) %>%
  GC_transcript(
    transcript = "transcript",
    type = "type",
    selection = 2,
    itemStyleExons = list(list(index = 0, fill = "red")
    )
  )
```

---

genbank_to_fasta *Convert GenBank to FASTA Format*

Description

This function reads a GenBank file, extracts the sequence and writes it to a new file in FASTA format. It parses the DEFINITION and VERSION for the header and sequences from the ORIGIN section.

Usage

```
genbank_to_fasta(path, output_dir = NULL)
```

Arguments

path	Path to the GenBank file.
output_dir	Optional path for the output FASTA file. If NULL, the output is saved in the same directory as the input file with the same base name but with a .fasta extension.

Value

None explicitly, but writes the FASTA formatted data to a file.

Examples

```
# Path to the example GenBank file in the package
gbk_file <- system.file("extdata", "BGC0000001.gbk", package = "geneviewer")

# Convert the GenBank file to FASTA format
genbank_to_fasta(gbk_file)
```

`geneviewer-shiny` *Shiny bindings for geneviewer*

Description

Output and render functions for using geneviewer within Shiny applications and interactive Rmd documents.

Usage

```
GC_chartOutput(outputId, width = "100%", height = "400px")

renderGC_chart(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

`outputId`	Output variable to read from.
`width`	height Must be a valid CSS unit (like '100 or a number, which will be coerced to a string and have 'px' appended.
`height`	Height of the output widget, must be a valid CSS unit (like '100 have 'px' appended.
`expr`	An expression that generates a GC chart.
`env`	The environment in which to evaluate expr.
`quoted`	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

Value

GC_chart widget that can be placed in the UI.

See Also

[GC_chart()]

Examples

```
if (interactive()) {
  library(shiny)
  library(geneviewer)

  ui <- fluidPage(
    titlePanel("Omphalotin Gene Cluster Visualization"),
    mainPanel(
      GC_chartOutput("gcChart", width = "100%", height = "500px")
    )
  )
```

```
  server <- function(input, output) {
    output$gcChart <- renderGC_chart({
      GC_chart(
        ophA_clusters, # Ensure 'ophA_clusters' data is defined or available
        cluster = "cluster",
        group = "class"
      ) %>%
      GC_title(title = c("<i>O. olearius</i>", "<i>D. bispora</i>")) %>%
      GC_labels("name") %>%
      GC_legend(position = "bottom") %>%
      GC_scaleBar() %>%
      GC_clusterLabel(title = "ophA")
    })
  }

  shinyApp(ui = ui, server = server)
}
```

`get_introns` *Calculate Intron Positions Based on Exons*

Description

This function takes a dataset containing transcript information and calculates the start and end positions of introns based on the positions of exons. It generates a combined dataframe including both the original exons and the calculated introns.

Usage

```
get_introns(data)
```

Arguments

`data`	A dataframe containing 'start' and 'end' positions of transcripts. Optional columns are 'type' (exon or UTR), 'transcript' (containing unique transcript IDs), and 'strand' (indicating the direction, forward or reverse, of each transcript).

Value

A dataframe with the original exons and the calculated introns, sorted by transcript and start position. Each row includes the 'start', 'end', 'type' (exon, UTR or intron), 'transcript', and 'strand' for each segment.

hs_dystrophin_transcripts *Human Dystrophin Transcripts Data*

Description

This dataset contains the Exon positions on human Dystrophin transcripts 'Dp427p2', 'Dp260-2', 'Dp140', 'Dp116', and 'Dp71'.

Usage

```
hs_dystrophin_transcripts
```

Format

A data frame with 202 rows and 5 columns:

transcript Transcript names. A character vector representing the names of the transcripts.

type Transcript type.

start Start positions. An integer vector showing the starting positions of each Exon.

end End positions. An integer vector showing the ending positions of each Exon.

length Transcript lengths. An integer vector indicating the length of each transcript.

human_hox_genes *Human HOX Gene Cluster Data*

Description

This dataset represents the positions and clusters of various human HOX genes. HOX genes are a group of related genes that control the body plan of an embryo along the head-tail axis. The dataset includes genes from HOXA, HOXB, HOXC, and HOXD clusters.

Usage

```
human_hox_genes
```

Format

A data frame with 39 rows and 6 columns:

symbol HGNC symbols for the HOX genes. A character vector.

chromosome_name Chromosome number where each gene is located. A numeric vector.

start Start positions of the genes on the chromosome. A numeric vector.

end End positions of the genes on the chromosome. A numeric vector.

strand -1 for reverse and 1 for forward.

cluster Cluster identification as a character vector, specifying the HOX gene cluster (HOXA, HOXB, HOXC, HOXD).

name Simplified names for the HOX genes. A character vector.

`mummer_alignment` *Perform Sequence Alignment Using MUMmer*

Description

This function orchestrates the alignment of sequences in a specified directory using MUMmer, a tool for aligning large DNA or protein sequences. It can handle GenBank and FASTA file formats. and performs checks to ensure necessary files are present.

Usage

```
mummer_alignment(
  path,
  cluster = NULL,
  maptype = "many-to-many",
  seqtype = "protein",
  mummer_options = "",
  filter_options = "",
  remove_files = TRUE,
  output_dir = tempdir()
)
```

Arguments

`path`	The directory containing the sequence files.
`cluster`	Optional vector of cluster names to consider for alignment. If NULL, clusters are inferred from file names. The order of names determines the alignment sequence.
`maptype`	The type of mapping to perform; "many-to-many" or "one-to-one". "many-to-many" allows for multiple matches between clusters, "one-to-one" restricts alignments to unique matches between a pair.
`seqtype`	The type of sequences, either "protein" or "nucleotide".
`mummer_options`	Additional command line options for MUMmer. To see all available options, you can run ‘nucmer –help‘ or ‘promer –help‘ in the terminal depending on whether you are aligning nucleotide or protein sequences.
`filter_options`	Additional options for filtering MUMmer results. To view all filtering options, run ‘delta-filter –help‘ in the terminal.
`remove_files`	Logical indicating whether to remove intermediate files generated during the process, defaults to TRUE.
`output_dir`	Optional directory for output files; defaults to `tempdir()`

Value

A data frame combining all alignment results, or NULL if errors occur during processing.

References

Kurtz S, Phillippy A, Delcher AL, Smoot M, Shumway M, Antonescu C, Salzberg SL (2004). Versatile and open software for comparing large genomes. Genome Biology, 5(R12).

Examples

```
## Not run:
# Basic alignment with default options
mummer_alignment(
  path = "/path/to/sequences",
  maptype = "many-to-many",
  seqtype = "protein"
)

# Alignment with specific MUMmer options
mummer_alignment(
  path = "/path/to/sequences",
  maptype = "one-to-one",
  seqtype = "protein",
  mummer_options = "--maxgap=500 --mincluster=100",
  filter_options = "-i 90"
)

## End(Not run)
```

---

ophA_clusters *ophA Gene Cluster from Omphalotus olearius*

Description

This dataset represents the ophA gene cluster from Omphalotus olearius and Dendrothele bispora, involved in the production of omphalotin, a cyclic N-methylated peptide.

Usage

```
ophA_clusters
```

Format

A data frame with 17 rows and 5 columns:

name Gene names. A character vector.

start Start positions of the genes. An integer vector.

end End positions of the genes. An integer vector.

class Classifications of the genes. A character vector indicating the type of protein or function associated with each gene.

cluster Cluster identification as a character vector, specifying whether the gene belongs to the ophA or dbophA gene cluster.

References

van der Velden NS et al. Autocatalytic backbone N-methylation in a family of ribosomal peptide natural products. Nat Chem Biol. 2017 Aug;13(8):833-835. doi: 10.1038/nchembio.2393. Epub 2017 Jun 5. PMID: 28581484.

`protein_blast` *Perform Protein BLAST Analysis Within Specified Clusters*

Description

This function conducts a BLAST analysis for protein sequences within specified clusters. It generates all possible protein combinations between a query cluster and other clusters, performs pairwise alignments, calculates sequence identity and similarity, and filters results based on a minimum identity threshold.

Usage

```
protein_blast(
  data,
  query,
  id = "protein_id",
  start = "start",
  end = "end",
  cluster = "cluster",
  genes = NULL,
  identity = 30,
  parallel = TRUE
)
```

Arguments

`data`	A dataframe or a character vector specifying the path to .gbk files. When a character vector is provided, it is interpreted as file paths to .gbk files which are then read and processed. The dataframe must contain columns for unique protein identifiers, cluster identifiers, protein sequences, and the start and end positions of each gene.
`query`	The name of the query cluster to be used for BLAST comparisons.
`id`	The name of the column that contains the gene identifiers. Defaults to "protein_id".

start	The name of the column specifying the start positions of genes. Defaults to "start".
end	The name of the column specifying the end positions of genes. Defaults to "end".
cluster	The name of the column specifying the cluster names. Defaults to "cluster".
genes	An optional vector of gene identifiers to include in the analysis. Defaults to NULL.
identity	Minimum identity threshold for BLAST hits to be considered significant. Defaults to 30.
parallel	Logical indicating whether to use parallel processing for alignments. Defaults to TRUE.

Value

A modified version of the input ‘data‘ dataframe, including additional columns for BLAST results (identity, similarity).

Note

This function relies on the Biostrings and pwalign package for sequence alignment and the dplyr package for data manipulation. Ensure these packages are installed and loaded into your R session.

Examples

```
## Not run:
path_to_folder <- "path/to/gbk/folder/"
data_updated <- protein_blast(
                        path_to_folder,
                        id = "protein_id",
                        query = "cluster A",
                        identity = 30
                        )

## End(Not run)
```

---

read_bed *Read BED Files*

Description

This function reads BED files from a specified directory or file path and combines them into a single data frame. BED files use 0-based coordinate starts, while this function transforms the coordinates to 1-based during import.

Usage

```
read_bed(path)
```

Arguments

path	A character string specifying the directory containing BED files or the file path to a single BED file.

Details

This function can read multiple BED files from a directory or a single BED file from a specified path. It adds a 'filename' column with the name of the file, and combines the data frames from all files into one.

Value

A data frame combining data from the BED files.

Examples

```
## Not run:
# Read BED files from a directory
bed_data <- read_bed("path/to/directory")

# Read a single BED file
bed_data <- read_bed("path/to/file.bed")

## End(Not run)
```

---

read_fasta *Read Protein Sequences from FASTA Files*

Description

This function reads protein sequences from the specified FASTA file or all FASTA files within a directory. It specifically looks for metadata in the FASTA headers with key-value pairs separated by an equals sign '='. For example, from the header '>protein1 [gene=scnD] [protein=ScnD]', it extracts 'gene' as the key and 'scnD' as its value, and similarly for other key-value pairs.

Usage

```
read_fasta(fasta_path, sequence = TRUE, keys = NULL, file_extension = "fasta")
```

Arguments

fasta_path	Path to the FASTA file or directory containing FASTA files.
sequence	Logical; if 'TRUE', the protein sequences are included in the returned data frame.
keys	An optional vector of strings representing specific keys within the fasta header to retain in the final data frame. If 'NULL' (the default), all keys within the specified feature are included.
file_extension	Extension of the FASTA files to be read from the directory (default is 'fasta').

Details

The Biostrings package is required to run this function.

Value

A data frame with columns for each piece of information extracted from the FASTA headers.

Examples

```
## Not run:
# Read sequences from a single FASTA file
sequences_df <- read_fasta("path/to/single_file.fasta")

# Read all sequences from a directory of FASTA files
sequences_df <- read_fasta("path/to/directory/", file_extension = "fa")

# Read sequences and include the protein sequences in the output
sequences_df <- read_fasta("path/to/directory/", sequence = TRUE)

## End(Not run)
```

---

read_gbk *Read Data from GenBank Files*

Description

This function reads data from a single GenBank file or directory with GenBank files. It allows selective extraction of information by specifying sections and features.

Usage

```
read_gbk(path, sections = NULL, features = NULL, origin = TRUE)
```

Arguments

path	A string representing the file path to the target GenBank (.gbk) file or directory.
sections	An optional vector of strings representing the names of specific sections within the GenBank file to extract (e.g., "LOCUS", "DEFINITION", "ACCESSION", "VERSION"). If ‘NULL‘ (the default), the function extracts all available sections.
features	An optional vector of strings indicating specific feature types to extract from the FEATURES section of the GenBank file (e.g., "CDS", "gene", "mRNA"). If ‘NULL‘ (the default), the function extracts all feature types present in the FEATURES section.
origin	A boolean flag; when set to ‘TRUE‘ (the default), the origin sequence data is included in the output.

Value

A list containing the contents of the specified sections and features of the GenBank file. Each section and feature is returned as a separate list element.

Examples

```
# Path to example GenBank file in the package
genbank_file <- system.file(
  "extdata",
  "BGC0000001.gbk",
  package = "geneviewer"
  )

# Read all data from the example GenBank file
gbk_data <- read_gbk(genbank_file)

# Read only specific sections from the example GenBank file
gbk_data <- read_gbk(genbank_file, sections = c("LOCUS", "DEFINITION"))

# Read specific features from the FEATURES section of the example GenBank file
gbk_data <- read_gbk(genbank_file, features = c("gene", "CDS"))

# Read data without the origin sequence
gbk_data <- read_gbk(genbank_file, origin = FALSE)
```

---

read_gff *Read GFF Files*

Description

This function reads GFF files from a specified directory or file path and combines them into a single data frame.

Usage

```
read_gff(path, fields = NULL)
```

Arguments

path	A character string specifying the directory containing GFF files or the file path to a single GFF file.
fields	An optional vector of character strings specifying the fields to extract from the GFF files.

Details

This function can read multiple GFF files from a directory or a single GFF file. It processes each file, extracts the specified fields (if provided), adds a 'name' column with the filename, and combines the data frames from all files into one.

Value

A data frame containing the combined data from the GFF files.

Examples

```
## Not run:
# Read GFF files from a directory
gff_data <- read_gff("path/to/directory")

# Read a single GFF file
gff_data <- read_gff("path/to/file.gff")

# Read specific fields from GFF files
gff_data <- read_gff(
"path/to/directory",
fields = c("seqid", "start", "end", "attributes")
)

## End(Not run)
```

Index