Package 'GRIN2'

May 7, 2026

Title Genomic Random Interval (GRIN)

Version 2.0.0

Description Improved version of 'GRIN' software that streamlines its use in practice to analyze genomic lesion data, accelerate its computing, and expand its analysis capabilities to answer additional scientific questions including a rigorous evaluation of the association of genomic lesions with RNA expression. Pounds, Stan, et al. (2013) <DOI:10.1093/bioinformatics/btt372>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Imports circlize, data.table, dplyr, forcats, ggplot2, graphics, grDevices, grid, magrittr, stats, stringr, survival, tibble, tidyselect, utils, writexl

Suggests knitr, rmarkdown, biomaRt, gridGraphics, Gviz, ensembldb, GenomeInfoDb, ComplexHeatmap,

NeedsCompilation no

Maintainer Abdelrahman Elsayed <aelsayed@stjude.org>

Depends R (>= 4.2.0)

LazyData true

VignetteBuilder knitr

URL https://github.com/abdel-elsayed87/GRIN2

BugReports https://github.com/abdel-elsayed87/GRIN2/issues

Author Abdelrahman Elsayed [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-8150-6825>), Xueyuan Cao [aut], Lakshmi Anuhya patibandla [aut], Stanley Pounds [aut, cph]

Repository CRAN

Date/Publication 2025-06-17 05:30:02 UTC

Contents

`alex.boxplots` *Generate Box Plots of Gene Expression by Lesion Groups*

Description

Generates box plots of gene expression levels stratified by lesion groups for a subset of genes selected based on a user-specified q-value threshold from Kruskal Wallis test results.

Usage

```
alex.boxplots(out.dir, alex.data, alex.kw.results, q, gene.annotation)
```

Arguments

`out.dir`	Path to the directory where the resulting PDF file containing the box plots will be saved.
`alex.data`	A list returned by the `alex.prep.lsn.expr` function, containing three components: `"row.mtch"`, `"alex.expr"` (gene expression matrix), and `"alex.lsn"` (lesion matrix). Rows of `alex.expr` and `alex.lsn` is ordered by Ensembl gene IDs; columns represent patient IDs.
`alex.kw.results`	Output of the `KW.hit.express` function containing Kruskal Wallis test results.
`q`	Q-value threshold. Only genes with q-values below this threshold will be included in the output plots.
`gene.annotation`	A `data.frame` containing gene annotation data, either provided by the user or retrieved using `get.ensembl.annotation`. Must contain four columns: `"gene"` (Ensembl gene ID), `"chrom"` (chromosome), `"loc.start"` (start position), and `"loc.end"` (end position).

Value

A PDF file saved in `out.dir`, containing one box plot per page for each selected gene, showing gene expression distribution across lesion groups.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org>, Stanley Pounds <stanley.pounds@stjude.org>

References

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

`alex.prep.lsn.expr`, `KW.hit.express`

Examples

```
data(expr_data)
data(lesion_data)
data(hg38_gene_annotation)

# Prepare expression and lesion data
alex.data <- alex.prep.lsn.expr(expr_data, lesion_data,
                                hg38_gene_annotation, min.expr = 5, min.pts.lsn = 5)

# Run Kruskal Wallis test
alex.kw.results <- KW.hit.express(alex.data, hg38_gene_annotation, min.grp.size = 5)

# Generate box plots for significant genes
dir.create(resultsFolder <- file.path(tempdir(), "temp.out"))
alex.boxplots(out.dir = resultsFolder,
              alex.data = alex.data,
              alex.kw.results = alex.kw.results,
              q = 1e-15,
              gene.annotation = hg38_gene_annotation)
unlink(resultsFolder, recursive = TRUE)
```

`alex.pathway` *Visualize Lesion and Expression Data by Pathway*

Description

Computes pairwise distances between subjects based on lesion profiles in genes associated with a specified pathway, and returns a figure with two panels: one showing lesion data and another showing expression data, both ordered based on the computed distances (useful for hierarchical clustering). It also returns the corresponding ordered data.

Usage

```
alex.pathway(alex.data, lsn.clrs = NULL, lsn.data, pathways, selected.pathway)
```

Arguments

`alex.data`	Output of the `alex.prep.lsn.expr` function. A list of three data tables: `"row.mtch"`, `"alex.expr"` (expression matrix), and `"alex.lsn"` (lesion matrix). Rows in both matrices are ordered by Ensembl gene IDs; columns represent patient IDs.
`lsn.clrs`	Optional. A named vector of colors for lesion types. If not specified, default colors will be assigned using `default.grin.colors()`.
`lsn.data`	A data frame of lesion data in GRIN-compatible format with the following columns: `"ID"` (patient ID), `"chrom"` (chromosome), `"loc.start"` (lesion start), `"loc.end"` (lesion end), and `"lsn.type"` (e.g., gain, loss, mutation, fusion).
`pathways`	A data frame with three columns: `"gene.name"` (gene symbol), `"ensembl.id"` (Ensembl gene ID), and `"pathway"` (pathway name).
`selected.pathway`	A character string indicating the pathway of interest.

Details

This function identifies all genes associated with the specified pathway, extracts lesion and expression data for those genes, and computes pairwise distances between subjects based on their lesion profiles. It uses hierarchical clustering to order the subjects and visualizes lesion and expression matrices in two aligned panels. It also returns a data frame containing the ordered expression and lesion data for all genes in the pathway.

Value

A list with the following element:

`ordered.path.data`
: A data frame with lesion and expression data for pathway genes, ordered according to hierarchical clustering (matching the order used in the plot).

A figure with two panels will also be generated showing:

1. Lesion data of pathway genes across subjects
2. Expression data of the same genes across subjects

Both panels are ordered by subject similarity based on lesion profiles.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org>, Stanley Pounds <stanley.pounds@stjude.org>

See Also

alex.prep.lsn.expr, hclust

Examples

```
data(expr_data)
data(lesion_data)
data(hg38_gene_annotation)
data(pathways)

# Prepare matched expression and lesion data
alex.data <- alex.prep.lsn.expr(expr_data, lesion_data,
                                hg38_gene_annotation, min.expr = 5, min.pts.lsn = 5)

# Visualize pathway-level association using JAK pathway as an example
alex.path=alex.pathway(alex.data,
                       lsn.data = lesion_data,
                       pathways = pathways,
                       selected.pathway = "Jak_Pathway")

# Access the ordered data matrix used in the plot
head(alex.path$ordered.path.data)
```

`alex.prep.lsn.expr` *Prepare Lesion and Expression Data for Kruskal Wallis Test*

Description

Prepares matched lesion and expression data matrices for use with the `KW.hit.express` function, which performs Kruskal Wallis tests to assess associations between lesion groups and gene expression levels.

Usage

```
alex.prep.lsn.expr(
  expr.mtx,
  lsn.data,
  gene.annotation,
  min.expr = NULL,
  min.pts.lsn = NULL
)
```

Arguments

`expr.mtx`	A data frame or matrix of normalized, log2-transformed gene expression values with genes in rows and subjects in columns. The first column must be named `"ensembl.ID"` and contain Ensembl gene IDs.
`lsn.data`	A data frame of lesion data in GRIN-compatible format. Must contain five columns: `"ID"` (patient ID), `"chrom"` (chromosome), `"loc.start"` (lesion start position), `"loc.end"` (lesion end position), and `"lsn.type"` (lesion type; e.g., gain, loss, mutation, fusion, etc..).
`gene.annotation`	A gene annotation data frame, either user-provided or retrieved via `get.ensembl.annotation()` from the GRIN2.0 package. Must contain four columns: `"gene"` (Ensembl gene ID), `"chrom"` (chromosome), `"loc.start"` (gene start), and `"loc.end"` (gene end).
`min.expr`	Minimum total expression level required for a gene to be retained (i.e., sum of expression values across all subjects). Useful to filter out genes with very low expression.
`min.pts.lsn`	Minimum number of subjects required to have a lesion in a gene for that gene to be retained for the KW test.

Details

The function uses `prep.lsn.type.matrix()` internally to create a lesion matrix where each gene is represented by one row and all lesion types are included. It filters genes to retain only those with both sufficient expression and lesion data. The final expression and lesion matrices are matched by gene and patient IDs, with rows ordered by Ensembl gene ID and columns by patient ID.

Value

A list with the following components:

`alex.expr`	A matrix of gene expression data with Ensembl gene IDs as row names and patient IDs as column names.
`alex.lsn`	A matrix of lesion data for the same genes and patients as in `alex.expr`, similarly ordered.
`alex.row.mtch`	A data frame with two columns showing the matched Ensembl gene IDs from the expression and lesion matrices.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org>, Stanley Pounds <stanley.pounds@stjude.org>

References

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

`KW.hit.express`

Examples

```
data(expr_data)
data(lesion_data)
data(hg38_gene_annotation)

# Prepare matched lesion and expression data
alex.data <- alex.prep.lsn.expr(expr_data, lesion_data,
                                hg38_gene_annotation, min.expr = 1,
                                min.pts.lsn = 5)
```

---

alex.waterfall.plot *Generate Waterfall Plot of Lesion and Expression Data*

Description

Creates a waterfall plot displaying gene expression levels grouped by lesion status for a selected gene.

Usage

```
alex.waterfall.plot(waterfall.prep, lsn.data, lsn.clrs = NULL, delta = 0.5)
```

Arguments

waterfall.prep	Output from alex.waterfall.prep. A list containing three data tables: "gene.lsn.exp" with patient IDs, lesion types, and expression levels for the gene of interest; "lsns" with all lesions affecting the gene (GRIN-compatible format); and "stats" with the Kruskal Wallis test result (from KW.hit.express).
lsn.data	Lesion data in GRIN-compatible format.
lsn.clrs	Named vector of colors for lesion types. If not provided, default colors will be automatically assigned using default.grin.colors().
delta	Spacing argument for the waterfall plot (default is 0.5).

Details

This function generates a waterfall-style plot that visualizes gene expression across patients, grouped by lesion category. Patients are grouped by lesion type (sorted alphabetically), and within each group, expression levels are ordered from lowest to highest. The median expression level appears at the center of each group, allowing intuitive comparison between lesion categories.

Value

A side-by-side graphical representation of lesion status and gene expression for each patient, grouped by lesion type.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org>, Stanley Pounds <stanley.pounds@stjude.org>

References

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

alex.prep.lsn.expr, KW.hit.express, alex.waterfall.prep

Examples

```
data(expr_data)
data(lesion_data)
data(hg38_gene_annotation)

# Prepare expression and lesion data
alex.data <- alex.prep.lsn.expr(expr_data, lesion_data,
                                hg38_gene_annotation, min.expr = 1, min.pts.lsn = 5)

# Run Kruskal Wallis test
alex.kw.results <- KW.hit.express(alex.data, hg38_gene_annotation, min.grp.size = 5)

# Prepare data for the WT1 gene
```

```
WT1.waterfall.prep <- alex.waterfall.prep(alex.data, alex.kw.results, "WT1", lesion_data)

# Generate waterfall plot for WT1
alex.waterfall.plot(WT1.waterfall.prep, lesion_data)
```

`alex.waterfall.prep` *Prepare Lesion and Expression Data for Waterfall Plots*

Description

Prepares matched lesion and expression data for a selected gene to be used with the `alex.waterfall.plot` function.

Usage

```
alex.waterfall.prep(alex.data, alex.kw.results, gene, lsn.data)
```

Arguments

`alex.data`	Output from `alex.prep.lsn.expr`. A list of three data tables: `"row.mtch"` (gene matching info), `"alex.expr"` (expression matrix), and `"alex.lsn"` (lesion matrix). Rows are ordered by Ensembl gene IDs, and columns are ordered by patient IDs.
`alex.kw.results`	Kruskal Wallis test results for gene expression by lesion group, as returned by `KW.hit.express`.
`gene`	Gene of interest, specified by either its gene symbol or Ensembl ID.
`lsn.data`	Lesion data in GRIN-compatible format. A data frame with five required columns: `"ID"` (patient ID), `"chrom"` (chromosome), `"loc.start"` (lesion start), `"loc.end"` (lesion end), and `"lsn.type"` (lesion type; e.g., gain, loss, mutation, fusion, etc...).

Details

This function extracts and combines lesion and expression data for a specified gene across patients. It returns a data table showing each patient's lesion status and expression level for the gene. It also extracts the corresponding Kruskal Wallis test result and all lesions that affect the gene from the lesion data.

Value

A list with the following components:

`gene.lsn.exp`	A data table with three columns: `"ID"` (patient ID), `"<gene_name>_lsn"` (lesion status: e.g., none, gain, mutation, multiple), and `"<gene_name>_expr"` (expression level of the gene in that patient).

lsns	A data table of all lesions affecting the gene of interest, extracted from the input lesion data (GRIN-compatible format).
stats	A one-row data frame containing the Kruskal Wallis test result for the gene, from KW.hit.express.
gene.ID	The gene name (symbol or Ensembl ID) provided as input.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org>, Stanley Pounds <stanley.pounds@stjude.org>

References

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

alex.prep.lsn.expr, KW.hit.express

Examples

```
data(expr_data)
data(lesion_data)
data(hg38_gene_annotation)

# Prepare matched expression and lesion data
alex.data <- alex.prep.lsn.expr(expr_data, lesion_data,
                                hg38_gene_annotation, min.expr = 1, min.pts.lsn = 5)

# Run Kruskal Wallis test
alex.kw.results <- KW.hit.express(alex.data, hg38_gene_annotation, min.grp.size = 5)

# Prepare lesion and expression data for waterfall plot of WT1
WT1.waterfall.prep <- alex.waterfall.prep(alex.data, alex.kw.results, "WT1", lesion_data)
```

---

clin_data *Example Clinical Dataset for T-cell Acute Lymphoblastic Leukemia (T-ALL)*

Description

This dataset contains clinical and demographic information for 265 newly diagnosed T-ALL patients. The data originates from Liu, Yu, et al. (2017) and includes variables relevant to patient characteristics and clinical outcomes.

Usage

```
clin_data
```

Format

A data frame with 265 rows and 11 columns:

ID Unique patient identifier

Sex Patient gender

Race Patient race

Age_Days Age at diagnosis, in days

WBC White Blood Cell count at diagnosis

MRD29 Minimal Residual Disease percentage at day 29 post-treatment

MRD.binary Categorical MRD status (0 = MRD <= 0.1, 1 = MRD > 0.1)

os.time Overall survival time in years (from diagnosis to last follow-up or death)

os.censor Overall survival status (0 = alive at last follow-up, 1 = deceased)

efs.time Event-free survival time in years

efs.censor Event status for event-free survival (0 = censored, 1 = event occurred)

Source

Liu, Yu, et al. (2017), *Nature Genetics*. Supplementary tables: https://www.nature.com/articles/ng.3909#Sec27. Additional clinical variables were integrated from the TARGET database (https://ocg.cancer.gov/programs/t

---

`compute.gw.coordinates`

Compute Genome-wide Plotting Coordinates

Description

Computes and assigns genome-wide plotting coordinates to lesion, gene, and chromosome data for use in genome-wide lesion plots.

Usage

```
compute.gw.coordinates(grin.res, scl = 1e+06)
```

Arguments

`grin.res`	GRIN results, typically the output of the `grin.stats` function.
`scl`	Chromosome unit length in base pairs. Default is 1,000,000, meaning each chromosome is divided into segments of 1 million base pairs for plotting.

Details

This function processes the GRIN results to add genome-wide x-axis coordinates necessary for plotting lesions and genes across all chromosomes. It divides each chromosome into segments based on the specified `scl` value and computes cumulative start and end positions across chromosomes to ensure a continuous x-axis. Specifically:

- Chromosome sizes are updated to include `x.start` and `x.end` columns, where each chromosome starts where the previous one ends.
- Gene and lesion data are similarly updated with `x.start` and `x.end` coordinates, scaled by `scl`, and adjusted for cumulative chromosome positions.

Value

A list identical in structure to the original `grin.res` object, with the following additions:

gene.hits Unchanged. GRIN gene-level summary statistics, including hit counts and p/q-values.

gene.lsn.data Unchanged. Gene-lesion overlaps showing which lesion affects which gene for each patient.

lsn.data Input lesion data with added `x.start` and `x.end` columns for genome-wide coordinates.

gene.data Input gene annotation data with added `x.start` and `x.end` columns for genome-wide coordinates.

chr.size Chromosome size table (22 autosomes + X and Y) with added `x.start` and `x.end` columns for plotting.

gene.index Mapping of `gene.lsn.data` rows to their corresponding chromosomes.

lsn.index Mapping of `gene.lsn.data` rows to their corresponding lesions.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org>, Stanley Pounds <stanley.pounds@stjude.org>

References

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

`grin.stats`

Examples

```
data(lesion_data)
data(hg38_gene_annotation)
data(hg38_chrom_size)

# Run GRIN model
grin.results <- grin.stats(lesion_data,
                           hg38_gene_annotation,
```

```
                          hg38_chrom_size)

# Assign genome-wide coordinates for plotting
genome.coord <- compute.gw.coordinates(grin.results)
```

`count.hits` *Count Gene Lesion Hits*

Description

Computes the number of genomic lesions ("hits") affecting each gene by lesion category. It also calculates the number of unique subjects whose lesions overlap each gene by lesion type.

Usage

```
count.hits(ov.data)
```

Arguments

`ov.data`	A list of six `data.frame` objects generated by the `find.gene.lsn.overlaps()` function, containing gene-lesion overlap data and supporting indices.

Details

This function summarizes the output of `find.gene.lsn.overlaps()` by generating two key matrices:

- **nsubj.mtx**: For each gene, the number of *unique subjects* with at least one overlapping lesion of each type.
- **nhit.mtx**: For each gene, the *total number of overlapping lesions* (hits), regardless of subject redundancy, categorized by lesion type.

For example, if the gene **NOTCH1** is affected by three separate mutations in the same subject, that subject will be counted once in `nsubj.mtx`, but all three hits will be counted in `nhit.mtx`.

Value

A list containing the following components:

`lsn.data`	Original input lesion data.
`lsn.index`	A `data.frame` indexing the rows in `gene.lsn.data` that correspond to each lesion.
`gene.data`	Original input gene annotation data.
`gene.index`	A `data.frame` indexing the rows in `gene.lsn.data` that correspond to each chromosome.
`nhit.mtx`	A `data.frame` where rows correspond to genes and columns to lesion types. Each value is the number of hits (lesions) of a certain type affecting the gene.

nsubj.mtx	A data.frame with the same structure as nhit.mtx, but showing the number of unique subjects with at least one hit of each lesion type per gene.
gene.lsn.data	A data.frame where each row represents a gene overlapped by a lesion. Includes gene name (gene) and subject ID (ID).
glp.data	A data.frame ordered by gene and lesion start positions. The cty column encodes event boundaries: 1 = gene start, 2 = lesion start, 3 = lesion end, 4 = gene end.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org>, Stanley Pounds <stanley.pounds@stjude.org>

References

Pounds, S. et al. (2013). A genomic random interval model for statistical analysis of genomic lesion data.

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

prep.gene.lsn.data, find.gene.lsn.overlaps

Examples

```
data(lesion_data)
data(hg38_gene_annotation)

# Prepare gene and lesion data for GRIN analysis:
prep.gene.lsn <- prep.gene.lsn.data(lesion_data, hg38_gene_annotation)

# Identify overlapping gene-lesion events:
gene.lsn.overlap <- find.gene.lsn.overlaps(prep.gene.lsn)

# Count the number of subjects and lesions (hits) affecting each gene:
count.nsubj.nhits <- count.hits(gene.lsn.overlap)
```

---

default.grin.colors *Assign Default GRIN Colors*

Description

Assigns a default set of colors to lesion types for use in GRIN plots.

Usage

```
default.grin.colors(lsn.types)
```

Arguments

lsn.types	A character vector of unique lesion types, typically derived from the lesion data.

Details

This function provides a predefined palette of up to 10 distinct colors for lesion types used in GRIN visualizations. If more than 10 lesion types are provided, the function will prompt the user to manually define custom colors to ensure visual distinction.

Value

A named character vector of colors corresponding to each lesion type.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org>, Stanley Pounds <stanley.pounds@stjude.org>

References

Pounds, S. B., et al. (2013). A genomic random interval model for statistical analysis of genomic lesion data.

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

Examples

```
data(lesion_data)

# Extract unique lesion types
lsn.types <- unique(lesion_data$lsn.type)

# Assign default colors to lesion types
default.grin.colors(lsn.types)
```

---

expr_data *Example T-ALL Gene Expression Dataset*

Description

Log2-normalized gene expression data for 417 genes across 265 newly diagnosed T-cell Acute Lymphoblastic Leukemia (T-ALL) patients, as reported by Liu, Yu, et al. (2017).

Usage

```
expr_data
```

Format

`expr_data`:

A data frame with 417 rows and 265 columns:

gene Ensembl gene IDs of the 417 selected genes included in the dataset.

... Each remaining column represents a T-ALL patient with log2-normalized expression values.

Source

Data extracted from the supplementary materials of Liu, Yu, et al. (2017), *Nature Genetics*. https://www.nature.com/articles/ng.3909#Sec27

`find.gene.lsn.overlaps` *Find Gene Lesion Overlaps*

Description

Identifies overlaps between genes and genomic lesions using the output from the `prep.gene.lsn.data()` function. This function detects all instances where a lesion spans or intersects the genomic coordinates of a gene.

Usage

```
find.gene.lsn.overlaps(gl.data)
```

Arguments

`gl.data`	A list of five `data.frame` objects returned by the `prep.gene.lsn.data()` function. These include processed and indexed gene and lesion data ready for overlap analysis.

Value

A list containing the following components:

`lsn.data`	Original input lesion data.
`gene.data`	Original input gene annotation data.
`gene.lsn.data`	A `data.frame` ordered by chromosome and start position, containing both gene and lesion entries. Each row includes a `cty` code indicating the type and boundary of the interval: 1 = gene start, 2 = lesion start, 3 = lesion end, 4 = gene end.
`gene.lsn.hits`	A `data.frame` where each row corresponds to a gene overlapped by a lesion. Includes 11 columns: `"gene"` (Ensembl gene ID), `"gene.chrom"`, `"gene.loc.start"`, `"gene.loc.end"` (chromosome and coordinates of the gene), `"ID"` (patient/sample ID), `"lsn.chrom"`, `"lsn.loc.start"`, `"lsn.loc.end"` (chromosome and coordinates of the lesion), and `"lsn.type"` (type of lesion).

gene.index	A data.frame indexing the rows in gene.lsn.data that correspond to each chromosome's genes (row start and row end per chromosome).
lsn.index	A data.frame indexing the rows in gene.lsn.data that correspond to each lesion (row start and row end per lesion).

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org>, Stanley Pounds <stanley.pounds@stjude.org>

References

Pounds, S. et al. (2013). A genomic random interval model for statistical analysis of genomic lesion data.

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

prep.gene.lsn.data

Examples

```
data(lesion_data)
data(hg38_gene_annotation)

# Prepare gene and lesion data for GRIN-based computations:
prep.gene.lsn <- prep.gene.lsn.data(lesion_data, hg38_gene_annotation)

# Identify genes that are overlapped by lesions:
gene.lsn.overlap <- find.gene.lsn.overlaps(prep.gene.lsn)
```

---

genomewide.log10q.plot

Genome-wide -log10(q-value) Plot

Description

Generates a genome-wide plot of -log10(q-values) for each annotated gene or lesion boundary evaluated by GRIN. The plot is lesion-type specific (e.g., gain, loss, mutation).

Usage

```
genomewide.log10q.plot(
  grin.res,
  lsn.grps,
  lsn.colors = NULL,
  max.log10q = NULL
)
```

Arguments

grin.res	GRIN results object (output from grin.stats) using either gene annotation or lesion boundaries as marker input.
lsn.grps	A character vector specifying which lesion group(s) to include in the plot.
lsn.colors	A named vector of colors corresponding to each lesion group. If NULL, default colors will be assigned using the default.grin.colors() function.
max.log10q	Numeric; maximum value for -log10(q-value) displayed on the plot. Any value above this threshold will be capped at max.log10q.

Details

This function visualizes the significance of lesions affecting genomic loci across chromosomes. It plots -log10(q-values) for either gene-based or lesion-boundary markers based on the GRIN analysis. The plot is faceted or colored by lesion group (e.g., gain, loss).

Value

A genome-wide plot showing -log10(q-values) for genes or lesion boundaries associated with specific lesion types.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org> and Stanley Pounds <stanley.pounds@stjude.org>

See Also

grin.lsn.boundaries

Examples

```
data(lesion_data)
data(hg38_gene_annotation)
data(hg38_chrom_size)

# Example 1: Use lesion boundaries for gains
gain <- lesion_data[lesion_data$lsn.type == "gain", ]
lsn.bound.gain <- grin.lsn.boundaries(gain, hg38_chrom_size)
GRIN.results.gain.bound <- grin.stats(gain, lsn.bound.gain, hg38_chrom_size)

genomewide.log10q.plot(GRIN.results.gain.bound,
                       lsn.grps = c("gain"),
                       lsn.colors = c("gain" = "red"),
                       max.log10q = 10)

# hg38_gene_annotation can be used instead of the boundaries as the marker data.

# This function can be used similarly for other lesion types (e.g., loss, mutation).
```

genomewide.lsn.plot *Genome-wide Lesion Plot*

Description

Generates a genome-wide lesion plot displaying all lesion types affecting different chromosomes.

Usage

```
genomewide.lsn.plot(
  grin.res,
  ordered = FALSE,
  pt.order = NULL,
  lsn.colors = NULL,
  max.log10q = NULL
)
```

Arguments

grin.res	GRIN results (output from the grin.stats function).
ordered	Logical; if TRUE, patient IDs will be reordered according to the pt.order data frame. If FALSE (default), patient IDs are ordered alphabetically.
pt.order	A data frame with two columns: "ID" (patient identifiers matching those in the lesion data) and "pts.order" (numeric vector specifying the new patient order from 1 to n patients). Only required if ordered = TRUE.
lsn.colors	A named vector of colors assigned to lesion types. If not provided, colors will be automatically assigned using the default.grin.colors function.
max.log10q	Numeric; maximum value for -log10(q-value) used in the plot. Any value greater than max.log10q will be capped at this value in the left panel of the plot.

Details

This function uses genome-wide coordinates (from compute.gw.coordinates) to generate a three-panel plot. The middle panel shows lesions by chromosome across patients. The left panel displays the -log10(q-values) from the GRIN results for each gene, and the right panel shows the number of patients affected at each locus, color-coded by lesion type.

Value

A genome-wide lesion plot consisting of three aligned panels:

- Middle panel: genome-wide lesion map across all chromosomes and patients.
- Left panel: -log10(q-values) of each locus from GRIN results showing Statistical Significance of Lesion Frequencies.
- Right panel: number of affected patients at each locus, colored by lesion category.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org> and Stanley Pounds <stanley.pounds@stjude.org>

References

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

compute.gw.coordinates

Examples

```
data(lesion_data)
data(hg38_gene_annotation)
data(hg38_chrom_size)

# Run GRIN analysis
grin.results <- grin.stats(lesion_data,
                           hg38_gene_annotation,
                           hg38_chrom_size)

# Generate genome-wide lesion plot with alphabetical patient ordering
genomewide.plot <- genomewide.lsn.plot(grin.results, max.log10q = 50)
```

---

get.chrom.length *Get Chromosome Length*

Description

Retrieves chromosome size data for the human GRCh38 (hg38) genome assembly using UCSC chromInfo data, accessed via the circlize package.

Usage

```
get.chrom.length(genome.assembly)
```

Arguments

genome.assembly	Character string specifying the genome assembly. Currently, only "Human_GRCh38" is supported.

Details

The function fetches chromosome size information from the UCSC genome browser via the circlize::read.chromInfo() function. It returns data for all 22 autosomes in addition to X and Y chromosomes in the human GRCh38 (hg38) assembly. Chromosome names are formatted without the "chr" prefix.

Value

A data frame with the following columns:

chrom Chromosome identifier (e.g., 1, 2, ..., X, Y).

size Chromosome size in base pairs.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org> and Stanley Pounds <stanley.pounds@stjude.org>

References

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

read.chromInfo

Examples

```
# Retrieve chromosome size data for the GRCh38 genome assembly
hg38.chrom.size <- get.chrom.length("Human_GRCh38")
```

---

get.ensembl.annotation

Get Ensembl Gene and Regulatory Features Annotation Data

Description

Retrieves gene and regulatory feature annotation data from the Ensembl BioMart database for the Human GRCh38 (hg38) genome assembly using the biomaRt package.

Usage

```
get.ensembl.annotation(genome.assembly)
```

Arguments

`genome.assembly`
: Character string. Currently, only `"Human_GRCh38"` is supported.

Details

This function retrieves:

- Gene annotation: Ensembl ID, chromosome, start/end positions, gene name, description, biotype, strand, and cytogenetic band.
- Predicted regulatory features: Promoters, enhancers, CTCF binding sites, etc., from the Ensembl regulatory build.
- Validated regulatory features: Experimentally confirmed enhancers and TSSs from the FANTOM5 project.

Value

A list with three components:

gene.annotation Data frame of gene-level annotation.

reg.annotation.predicted Data frame of predicted regulatory features.

reg.annotation.validated Data frame of validated regulatory features (FANTOM5).

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org> and Stanley Pounds <stanley.pounds@stjude.org>

References

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

Zerbino, Daniel R., et al. (2015). The ensembl regulatory build.

Kinsella, Rhoda J., et al. (2011). Ensembl BioMarts: a hub for data retrieval across taxonomic space.

See Also

useEnsembl, getBM

Examples

```
hg38.ann <- get.ensembl.annotation("Human_GRCh38")
# gene level annotations:
hg38.genes <- hg38.ann$gene.annotation
# regulatory sequences from the ensembl genome build:
hg38.reg.pred <- hg38.ann$reg.annotation.predicted
# regulatory sequences from the FANTOM5 project:
hg38.reg.val <- hg38.ann$reg.annotation.validated
```

grin.assoc.lsn.outcome

Associate Lesions with Clinical Outcomes

Description

Performs statistical association analysis between binary gene-lesion events and clinical outcomes of interest, including binary outcomes (e.g., Minimal Residual Disease (MRD)) and time-to-event outcomes (e.g., Event-Free Survival (EFS), and Overall Survival (OS)). The function supports both univariate and covariate-adjusted logistic regression and Cox proportional hazards models.

Usage

```
grin.assoc.lsn.outcome(
  lsn.mtx,
  clin.data,
  annotation.data,
  clinvars,
  covariate = NULL
)
```

Arguments

`lsn.mtx`	A binary lesion matrix where each row represents a unique gene-lesion pair (e.g., `ENSG00000148400_gain`). Each column represents a patient. Values are denoted as `1` if the patient harbors the specified lesion, and `0` otherwise. This matrix is typically produced using the `prep.binary.lsn.mtx` function.
`clin.data`	A clinical data `data.frame`, where the first column `"ID"` represent patient identifiers matching those in `lsn.mtx`.
`annotation.data`	A gene annotation `data.frame`, either provided by the user or retrieved using the `get.ensembl.annotation` function. Must include the columns: `"gene"` (Ensembl ID), `"chrom"` (chromosome), `"loc.start"` (gene start position), and `"loc.end"` (gene end position).
`clinvars`	A character vector of clinical outcome variables to analyze. Binary variables (e.g., MRD) should be coded as `0` (negative) and `1` (positive). Survival outcomes (e.g., EFS, OS) must be precomputed using the `Surv` function and added as new columns to `clin.data`.
`covariate`	Optional. A character vector specifying covariates to include as model adjustments (e.g., risk group, age, gender, etc...).

Details

For each gene-lesion pair in the binary lesion matrix, the function can performs:

- **Logistic regression** for binary outcomes (e.g., MRD), producing odds ratios (OR), 95_confidence intervals (CI), p-values, and FDR-adjusted q-values.

- **Cox proportional hazards models** for survival outcomes (e.g., EFS, OS), producing hazard ratios (HR), 95\

Models can optionally be adjusted for covariates such as clinical or demographic factors. Summary counts of patients with and without lesions, stratified by outcome status, are also included in the output.

Value

A results `data.frame` containing gene annotation and association statistics for each gene-lesion pair across the specified clinical outcomes. The output includes:

- Odds ratio (OR), lower and upper 95CI, p-value, and q-value (FDR) for logistic regression models.
- Hazard ratio (HR), lower and upper 95CI, p-value, and q-value for Cox proportional hazards models.
- Patient counts for those with and without lesions, and corresponding outcome event statuses.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org> and Stanley Pounds <stanley.pounds@stjude.org>

References

Andersen, P. K., & Gill, R. D. (1982). Cox's regression model for counting processes: A large sample study.

Therneau, T. M., & Grambsch, P. M. (2000). *Modeling Survival Data: Extending the Cox Model*.

Dobson, A. J. (1990). *An Introduction to Generalized Linear Models*.

See Also

`prep.binary.lsn.mtx`, `coxph`, `glm`

Examples

```
data(lesion_data)
data(hg38_gene_annotation)
data(clin_data)

# Step 1: Prepare gene-lesion overlap
gene.lsn <- prep.gene.lsn.data(lesion_data, hg38_gene_annotation)
gene.lsn.overlap <- find.gene.lsn.overlaps(gene.lsn)

# Step 2: Create a binary lesion matrix (minimum 5 patients per lesion)
lsn.binary.mtx <- prep.binary.lsn.mtx(gene.lsn.overlap, min.ngrp = 5)

# Step 3: Create survival objects and add to clinical data
library(survival)
clin_data$EFS <- Surv(clin_data$efs.time, clin_data$efs.censor)
clin_data$OS <- Surv(clin_data$os.time, clin_data$os.censor)
```

```
# Step 4: Specify outcomes of interest
clinvars <- c("MRD.binary", "EFS", "OS")

# Step 5: Run association analysis
assc.outcomes <- grin.assoc.lsn.outcome(lsn.binary.mtx,
                                        clin_data,
                                        hg38_gene_annotation,
                                        clinvars)

# Optional: Adjust for covariates using the 'covariate' argument
```

`grin.barplt` *GRIN Lesion Stacked Bar Plot*

Description

Generates a stacked bar plot showing the number of patients affected by different types of genomic lesions in a user-specified list of genes of interest, based on GRIN analysis results.

Usage

```
grin.barplt(grin.res, count.genes, lsn.colors = NULL)
```

Arguments

`grin.res`	A data frame of GRIN results, typically the output from the `grin.stats` function.
`count.genes`	A character vector of gene names to include in the bar plot. Only genes present in the GRIN results table will be used.
`lsn.colors`	(Optional) A named vector specifying colors for each lesion type. If not provided, default lesion colors will be automatically assigned using the `default.grin.colors` function.

Details

The function subsets the GRIN results to the genes specified in `count.genes`, extracts the number of patients affected by each lesion type for each gene, and visualizes the data as a horizontal stacked bar plot. Each bar represents a gene and is segmented by lesion type (e.g., gain, loss, mutation), with segment size proportional to the number of affected patients.

This visualization is useful for highlighting the burden and distribution of different lesion types across key driver genes or other genes of interest.

Value

A ggplot2-based stacked bar plot showing lesion type distribution across the selected genes.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org> and Stanley Pounds <stanley.pounds@stjude.org>

See Also

grin.stats, default.grin.colors

Examples

```
data(lesion_data)
data(hg38_gene_annotation)
data(hg38_chrom_size)

# Run GRIN analysis
grin.results <- grin.stats(lesion_data,
                           hg38_gene_annotation,
                           hg38_chrom_size)

# Define a list of genes to be included in the bar plot (e.g., candidate driver genes)
count.genes <- c("TAL1", "FBXW7", "PTEN", "IRF8", "NRAS",
                 "BCL11B", "MYB", "LEF1", "RB1", "MLLT3",
                 "EZH2", "ETV6", "CTCF", "JAK1", "KRAS",
                 "RUNX1", "IKZF1", "KMT2A", "RPL11", "TCF7",
                 "WT1", "JAK2", "JAK3", "FLT3")

# Generate the bar plot
grin.barplt(grin.results, count.genes)
```

---

grin.lsn.boundaries *GRIN Evaluation of Lesion Boundaries*

Description

This function evaluates copy number variations (CNVs), specifically gains and deletions, using unique lesion start and end positions to define genomic boundaries. The analysis is lesion-type specific and spans the entire genome.

Usage

```
grin.lsn.boundaries(lsn.data, chrom.size)
```

Arguments

lsn.data	A lesion data table containing only gain or deletion events. If gains are subdivided (e.g., into gains and amplifications based on log2Ratio values), both subtypes can be included. The same applies for deletions (e.g., homozygous and heterozygous).
chrom.size	A chromosome size table with two required columns: "chrom" (chromosome identifier) and "size" (chromosome size in base pairs).

Details

The function identifies unique CNV boundaries by evaluating the start and end positions of lesions on each chromosome. Large lesions may be split into smaller boundaries based on overlapping smaller lesions in other samples. This boundary-based approach ensures comprehensive genome-wide coverage, including intergenic regions and areas without annotated features.

The first boundary on each chromosome spans from the start of the chromosome to the start of the first lesion affecting any of the included patient samples. Similarly, the last boundary extends from the end of the last lesion to the end of the chromosome.

This method is particularly useful when analyzing CNV data at a finer resolution than gene-level annotation allows.

Value

A `data.frame` with five columns:

`gene`	Boundary identifier, based on unique start and end positions.
`chrom`	Chromosome on which the boundary resides.
`loc.start`	Start position of the boundary.
`loc.end`	End position of the boundary.
`diff`	Length of the boundary in base pairs.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org> and Stanley Pounds <stanley.pounds@stjude.org>

See Also

`grin.stats`

Examples

```
data(lesion_data)
data(hg38_chrom_size)

# This analysis is lesion-type specific. For example, extract gains:
gain <- lesion_data[lesion_data$lsn.type == "gain", ]

# Generate lesion boundaries for gains:
lsn.bound.gain <- grin.lsn.boundaries(gain, hg38_chrom_size)

# Run GRIN using lesion boundaries as markers instead of gene annotations:
GRIN.results.gain.bound <- grin.stats(gain, lsn.bound.gain, hg38_chrom_size)

# The same analysis can be applied to deletions, mutations, or structural rearrangements.
```

grin.oncoprint.mtx *GRIN OncoPrint-Compatible Lesion Matrix*

Description

Prepares a binary lesion matrix based on GRIN analysis results that can be directly used as input for the oncoPrint function from the **ComplexHeatmap** package. This matrix summarizes the presence or absence of lesion types across patients for a user-defined list of genes.

Usage

```
grin.oncoprint.mtx(grin.res, oncoprint.genes)
```

Arguments

grin.res	A data frame of GRIN results, typically generated by the grin.stats function.
oncoprint.genes	A character vector of Ensembl gene IDs specifying the genes to include in the OncoPrint.

Details

This function filters the GRIN results for a specified set of genes (using their Ensembl IDs), and constructs a gene-by-patient binary matrix indicating the presence of one or more lesion types per gene. Each row represents a gene, each column a patient, and the matrix values reflect whether that gene is affected by any lesion in the given patient.

The output matrix is fully compatible with the oncoPrint() function from the **ComplexHeatmap** package and allows visualization of lesion patterns across a defined gene set.

This is particularly useful for visualizing mutation, copy number alterations and other structural rearrangements in driver genes or genes selected by statistical criteria (e.g., significance threshold from GRIN results).

Value

A binary data frame (matrix) of dimensions length(oncoprint.genes) and number of patients, suitable for use with the oncoPrint() function.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org> and Stanley Pounds <stanley.pounds@stjude.org>

References

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

grin.stats, oncoPrint

Examples

```
data(lesion_data)
data(hg38_gene_annotation)
data(hg38_chrom_size)

# Run GRIN analysis
grin.results <- grin.stats(lesion_data,
                           hg38_gene_annotation,
                           hg38_chrom_size)

# Define a list of genes (using Ensembl IDs) to include in the OncoPrint
oncoprint.genes <- c("ENSG00000148400", "ENSG00000171862", "ENSG00000171843",
                     "ENSG00000156531", "ENSG00000162367", "ENSG00000096968",
                     "ENSG00000105639", "ENSG00000118513", "ENSG00000102974",
                     "ENSG00000133703")

# Alternatively, select genes with multiple lesion types and significant q-values
genes.const <- grin.results$gene.hits[grin.results$gene.hits$q2.nsubj < 0.01, ]
selected.genes <- as.vector(genes.const$gene)

# Generate OncoPrint-compatible lesion matrix
oncoprint.mtx <- grin.oncoprint.mtx(grin.results, oncoprint.genes)
```

---

grin.stats *Execute GRIN Statistical Framework*

Description

Executes the Genomic Random Interval (GRIN) statistical framework to determine whether a specific genomic locus (gene or regulatory region) is significantly affected by either individual or a constellation of multiple lesion types.

Usage

```
grin.stats(lsn.data, gene.data = NULL, chr.size = NULL, genome.version = NULL)
```

Arguments

lsn.data	A data.frame containing lesion data formatted for GRIN analysis. It must include the following five columns: • ID: Sample or patient identifier. • chrom: Chromosome on which the lesion is located. • loc.start: Genomic start coordinate of the lesion. • loc.end: Genomic end coordinate of the lesion.

- `lsn.type`: Lesion type (e.g., gain, loss, mutation, fusion, etc...).

For Single Nucleotide Variants (SNVs), loc.start and loc.end should be the same. For Copy Number Alterations (CNAs) such as gains and deletions, these fields represent the lesion start and end positions (lesion boundary). Structural rearrangements (e.g., translocations, inversions) should be represented by two entries (two separate rows), one for each breakpoint. An example dataset is available in the GRIN2 package (`lesion_data.rda`).

`gene.data` A `data.frame` containing gene annotation data. Must include the following columns:

- `gene`: Ensembl gene ID.
- `chrom`: Chromosome where the gene is located.
- `loc.start`: Gene start position.
- `loc.end`: Gene end position.

This data can be user-provided or retrieved automatically via `get.ensembl.annotation()` if `genome.version` is specified.

`chr.size` A `data.frame` specifying chromosome sizes. Must contain:

- `chrom`: Chromosome number.
- `size`: Chromosome length in base pairs.

The data can be user-provided or directly retrieved using `get.chrom.length()` if `genome.version` is specified.

`genome.version` Optional. If gene annotation and chromosome size files are not provided, users can specify a supported genome assembly to retrieve these files automatically. Currently, the package only support "Human_GRCh38" genome assembly.

Details

The GRIN algorithm evaluates each locus to determine whether the observed frequency and distribution of lesions is greater than expected by chance. This is modeled using a convolution of independent, non-identical Bernoulli distributions, accounting for lesion type, locus size, and chromosome context.

For each gene, the function calculates:

- A p-value for the enrichment of lesion events
- An FDR-adjusted q-value using the Pounds & Cheng (2006) method
- Significance of multi-lesion constellation patterns (e.g., p-value for a locus being affected by 1, 2, etc., lesion types)

Value

A list containing:

`gene.hits` A `data.frame` of GRIN results for each gene, including annotation, subject/hit counts by lesion type, and p/q-values for individual and multi-lesion constellation significance.

`lsn.data` The original lesion input data.

gene.data	The original gene annotation input data.
gene.lsn.data	A data.frame where each row represents a gene-lesion overlap. Includes columns "gene" (Ensembl ID) and "ID" (sample ID).
chr.size	The chromosome size reference table used in computations.
gene.index	Indexes linking genes to rows in gene.lsn.data by chromosome.
lsn.index	Indexes linking lesions to rows in gene.lsn.data.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org>, Stanley Pounds <stanley.pounds@stjude.org>

References

Pounds, S. et al. (2013). A genomic random interval model for statistical analysis of genomic lesion data.

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

prep.gene.lsn.data, find.gene.lsn.overlaps, count.hits, prob.hits

Examples

```
data(lesion_data)
data(hg38_gene_annotation)
data(hg38_chrom_size)

# Example1: Run GRIN with user-supplied annotation and chromosome size:
grin.results <- grin.stats(lesion_data,
                           hg38_gene_annotation,
                           hg38_chrom_size)

# Example 2: User can specify genome version to automatically retrieve annotation
# and chromosome size data:
# grin.results <- grin.stats(lesion_data,
#                            genome.version = "Human_GRCh38")
```

---

grin.stats.lsn.plot *GRIN Statistics Lesions Plot*

Description

Generates a plot showing all lesion types that span a specified gene or regulatory feature with GRIN stats added.

Usage

```
grin.stats.lsn.plot(grin.res, feature = NULL, lsn.clrs = NULL, expand = 5e-04)
```

Arguments

grin.res	GRIN results for genes or regulatory elements, as returned by the grin.stats function.
feature	Ensembl ID of a feature of interest. This can be either a gene (e.g., Ensembl gene ID) or a regulatory region from Ensembl Regulatory Build or FANTOM5 project.
lsn.clrs	Named vector of colors for each lesion type. If not provided, colors are automatically assigned using default.grin.colors.
expand	Numeric value that controls the proportion of the flanking region (upstream/downstream) around the gene to be included. Default is 0.0005. Set to 0 to restrict the plot strictly to the locus boundaries.

Details

The plot consists of two panels:

- **Top panel**: Displays all lesion types overlapping the selected gene or regulatory feature. Lesions are color-coded by type, as indicated in the legend.
- **Bottom panel**: Summarizes GRIN statistics for the feature, including the number of subjects affected per lesion type, and the corresponding $-\log_{10}(p)$ and $-\log_{10}(q)$ values for significance.

This plot is particularly useful for regulatory features, which typically lack transcript structure. Therefore, no transcript or exon structure is shown.

Value

A two-panel plot showing lesion distribution and GRIN statistics for a given gene or regulatory feature, without a transcript annotation panel.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org> and Stanley Pounds <stanley.pounds@stjude.org>

References

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

grin.stats

Examples

```
data(lesion_data)
data(hg38_gene_annotation)
data(hg38_chrom_size)

# Run GRIN analysis
grin.results <- grin.stats(lesion_data,
                           hg38_gene_annotation,
                           hg38_chrom_size)

# Plot lesion and GRIN stats for a gene of interest (e.g., WT1)
grin.stats.lsn.plot(grin.results, feature = "ENSG00000184937")

# Can also be also used for regulatory features without transcript panels
```

---

hg38_chrom_size *Chromosome Length Data (hg38)*

Description

This dataset contains the lengths (in base pairs) of the 22 autosomes in addition to the X and Y chromosomes based on the GRCh38 human genome assembly. The data was retrieved from the UCSC Genome Browser using the `get.chrom.length` function with "Human-GRCh38" as the selected genome.

Usage

```
hg38_chrom_size
```

Format

`hg38_chrom_size`**:**

A data frame with 24 rows and 2 columns:

chrom Chromosome name (1,2, 3, ..., X, Y).

size Chromosome length in base pairs.

Source

Retrieved from UCSC Genome Browser `chr.info` text files using the `get.chrom.length` function with the GRCh38 genome assembly.

hg38_cytoband *GRCh38 Chromosome Cytobands*

Description

This dataset contains the start and end positions (in base pairs) of cytogenetic bands (cytobands) for all 22 autosomes, as well as the X and Y chromosomes, based on the Human GRCh38 (hg38) genome assembly.

Usage

```
hg38_cytoband
```

Format

`hg38_cytoband`**:**

A data frame with 1,549 rows and 5 columns:

chrom Chromosome name (1:22, X, Y).

chromStart Start position of the cytoband in base pairs.

chromEnd End position of the cytoband in base pairs.

name Name of the cytoband (e.g., p11.1, q22.3).

gieStain Staining pattern (e.g., gpos, gneg, acen) used for cytogenetic visualization.

Source

Retrieved from the UCSC Genome Browser (GRCh38 assembly): https://hgdownload.soe.ucsc.edu/goldenPath/hg38/database/

hg38_gene_annotation *Example Gene Annotation Data File*

Description

This dataset contains example annotation data for 417 selected genes (matching the gene set in the example gene expression dataset). The data was retrieved from the Ensembl BioMart database using the `get.ensembl.annotation` function with "Human-GRCh38" as the genome assembly (hg38).

Usage

```
hg38_gene_annotation
```

Format

hg38_gene_annotation:

A data frame with 417 rows and 9 columns:

gene Ensembl gene ID.

chrom Chromosome on which the gene is located.

loc.start Gene start position (in base pairs).

loc.end Gene end position (in base pairs).

description Description of the gene.

gene.name Gene symbol.

biotype Gene biotype, including categories such as protein-coding, long noncoding RNA (lncRNA), microRNA (miRNA), small nuclear RNA (snRNA), small nucleolar RNA (snoRNA), immunoglobulin (IG), T-cell receptor (TCR), and pseudogene.

chrom.strand Strand on which the gene is located: forward (1) or reverse (-1).

chrom.band Chromosomal cytoband where the gene is located.

Source

Retrieved from the Ensembl BioMart database using the `get.ensembl.annotation` function with the "Human-GRCh38" genome assembly (hg38). Then a subset of 417 out of around 62,000 genes was selected for the example dataset.

KW.hit.express *Associate Lesion Groups with Gene Expression*

Description

Performs the Kruskal Wallis (KW) test to evaluate the association between lesion groups and the expression level of the corresponding gene.

Usage

```
KW.hit.express(alex.data, gene.annotation, min.grp.size = NULL)
```

Arguments

`alex.data` Output from the `alex.prep.lsn.expr` function. A list of three data frames:

- `row.mtch`: Table of matched lesion expression entries, including gene IDs.
- `alex.expr`: Gene expression matrix (rows = genes by Ensembl ID, columns = patient IDs).
- `alex.lsn`: Lesion status matrix with the same dimensions/order as `alex.expr`.

`gene.annotation` Gene annotation table. Must be a `data.frame` with the following columns: `gene` (Ensembl gene ID), `chrom` (chromosome), `loc.start` (start position), and `loc.end` (end position). Can be obtained manually or via the `get.ensembl.annotation` function.

`min.grp.size` Minimum number of patients required in a lesion group to be included in the test. For a gene to be tested, there must be at least two groups with more than `min.grp.size` patients each.

Details

For each row in the `row.mtch` table, the function performs a Kruskal Wallis test comparing expression values of the gene across lesion groups. The lesion groups are defined in the `alex.lsn` matrix. Patients with multiple types of lesions in a gene are assigned the label `"multiple"`, and those with no lesion are labeled `"none"`. The expression values are obtained from the corresponding row in `alex.expr`.

The function tests whether expression levels significantly differ across lesion groups for the same gene.

Value

A data frame where each row corresponds to a gene tested. Columns include:

`gene`	Ensembl gene ID.
`gene.name`	HGNC gene symbol.
`p.KW`	Kruskal Wallis test p-value.
`q.KW`	FDR-adjusted q-value for multiple testing correction.
`_n.subjects`	Number of subjects in each lesion group, including "none" and "multiple".
`_mean`	Mean expression per lesion group.
`_median`	Median expression per lesion group.
`_sd`	Standard deviation of expression per lesion group.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org> and Stanley Pounds <stanley.pounds@stjude.org>

References

Hollander, M., & Wolfe, D. A. (1973). Nonparametric Statistical Methods. New York: Wiley. pp. 115-120.

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

`alex.prep.lsn.expr`

Examples

```
data(expr_data)
data(lesion_data)
data(hg38_gene_annotation)

# Prepare matched lesion-expression data
alex.data <- alex.prep.lsn.expr(expr_data, lesion_data,
                                hg38_gene_annotation,
                                min.expr = 1, min.pts.lsn = 5)

# Perform Kruskal Wallis test between lesion groups and expression levels:
alex.kw.results <- KW.hit.express(alex.data,
                                  hg38_gene_annotation,
                                  min.grp.size = 5)
```

`lesion_data` *Example T-ALL Lesion Dataset*

Description

Genomic lesion dataset including copy number variations, single nucleotide variants, and structural rearrangements affecting 265 newly diagnosed T-cell Acute Lymphoblastic Leukemia (T-ALL) patients, as reported by Liu, Yu, et al. (2017). The original lesion coordinates were based on the GRCh37 (hg19) human genome assembly. We converted these coordinates to GRCh38 (hg38) using the UCSC LiftOver tool (https://genome.ucsc.edu/cgi-bin/hgLiftOver) prior to running GRIN2 analyses.

Usage

```
lesion_data
```

Format

`lesion_data`**:**

A data frame with 6,861 rows and 5 columns:

ID Patient identifier for the individual affected by the lesion

chrom Chromosome on which the lesion is located

loc.start Lesion start position (in base pairs, hg38)

loc.end Lesion end position (in base pairs, hg38)

lsn.type Type of lesion (e.g., gain, loss, mutation, fusion, etc.)

Source

Adapted from the supplementary tables of Liu, Yu, et al. (2017), *Nature Genetics* (https://www.nature.com/articles/ng.3909#Sec27)

`lsn.transcripts.plot` *Plot of Gene Lesions and Transcripts*

Description

The function can generate four types of lesion plots. (1) If the `gene` argument is specified, the function returns a gene-level plot showing all lesions affecting the gene, along with the transcript track and GRIN statistics. (2) If `chrom`, `plot.start`, and `plot.end` are specified, the function generates a locus-level plot for that genomic region, including the transcript track. If `transTrack = FALSE`, the function can return similar plots without the transcript track (useful for large regions such as chromosome bands or entire chromosomes). (3) If `lesion.grp` is specified, only lesions from that specific group will be shown in the plot. (4) If `lesion.grp` is not specified, all lesion types will be displayed for the given locus.

Usage

```
lsn.transcripts.plot(
  grin.res,
  gene = NULL,
  transTrack = TRUE,
  lsn.clrs = NULL,
  chrom = NULL,
  plot.start = NULL,
  plot.end = NULL,
  lesion.grp = NULL,
  spec.lsn.clr = NULL,
  extend.left = NULL,
  extend.right = NULL,
  expand = 5e-04,
  hg38.transcripts = NULL,
  hg38.cytoband = NULL
)
```

Arguments

`grin.res`	GRIN results (Output of the `grin.stats()` function).
`gene`	Gene symbol of interest.
`transTrack`	Logical; if `FALSE`, the transcript track will be excluded (useful for plots of large genomic regions such as entire chromosome arms or bands).
`lsn.clrs`	Optional named vector of lesion colors. If not provided, default colors from `default.grin.colors()` will be used.
`chrom`	Chromosome number. Required when plotting a locus (used with `plot.start` and `plot.end`).
`plot.start`	Start coordinate (in base pairs) of the locus of interest.
`plot.end`	End coordinate (in base pairs) of the locus of interest.

lesion.grp	Lesion group to include in locus plots. Only lesions from this group will be shown. Required when chrom, plot.start, and plot.end are specified.
spec.lsn.clr	Optional color for highlighting the lesion group of interest in locus plots.
extend.left	Optional numeric value to extend the left side of the transcripts track (for alignment adjustments).
extend.right	Optional numeric value to extend the right side of the transcripts track (for alignment adjustments).
expand	Numeric; controls the proportion of upstream and downstream regions included in the plot relative to gene coordinates. Default is 0.0005. Set to 0 to plot the gene region only.
hg38.transcripts	Transcripts data from AnnotationHub (hg38, version 110). Required if transTrack = TRUE.
hg38.cytoband	Data frame of hg38 cytogenetic bands (start and end coordinates in base pairs).

Details

The function returns a plot that displays lesions affecting either a gene or a user-defined genomic region. When plotting a gene:

- The top panel shows all transcripts of certain gene or group of genes in a small region retrieved from Ensembl if transTrack=TRUE (default).
- The middle panel visualizes lesions affecting the gene or locus, color-coded by type.
- The bottom panel presents GRIN statistics, including the number of affected subjects, -log10(p), and -log10(q) values in case of gene plots.

When plotting a genomic locus (via chrom, plot.start, plot.end):

- Only the transcripts track (if transTrack = TRUE) and lesion panel are shown.
- GRIN statistics are omitted.

For large regions like cytobands or entire chromosomes, set transTrack = FALSE to avoid overcrowding from long transcript tracks.

Value

A multi-panel plot showing:

- Lesions and transcripts for a gene (with GRIN statistics), or
- Lesions and optional transcripts for a genomic locus, or
- Lesions alone for large genomic regions if transcripts and GRIN panels are excluded.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org> and Stanley Pounds <stanley.pounds@stjude.org>

References

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

grin.stats

Examples

```
data(lesion_data)
data(hg38_gene_annotation)
data(hg38_chrom_size)
data(hg38_cytoband)

# run GRIN analysis using grin.stats function
grin.results=grin.stats(lesion_data,
                        hg38_gene_annotation,
                        hg38_chrom_size)

# Plots Showing Different Types of Lesions Affecting a region of Interest without plotting the
# transcripts track (this will allow plotting a larger locus of the chromosome such as a
# chromosome band (should specify transTrack = FALSE):
cdkn2a.locus=lsn.transcripts.plot(grin.results, transTrack = FALSE,
                                  hg38.cytoband=hg38_cytoband, chrom=9,
                                  plot.start=19900000, plot.end=25600000,
                                   lesion.grp = "loss", spec.lsn.clr = "blue")

 # Plots Showing Different Types of Lesions Affecting the whole chromosome:
 chrom.plot=lsn.transcripts.plot(grin.results, transTrack = FALSE,
                                 hg38.cytoband=hg38_cytoband, chrom=9,
                                 plot.start=1, plot.end=141000000)
```

---

onco.print.props *Oncoprint Proportions by Lesion Type*

Description

Calculates and assigns the proportion of each oncoprint rectangle to be color-filled based on the average size of lesion types. Lesion types are ordered by their average genomic size, and proportions are either computed automatically or manually specified by the user.

Usage

```
onco.print.props(lsn.data, clr = NULL, hgt = NULL)
```

Arguments

`lsn.data`	A data frame with five columns: • `ID`: Subject or patient identifier • `chrom`: Chromosome on which the lesion is located • `loc.start`: Start genomic position of the lesion • `loc.end`: End genomic position of the lesion • `lsn.type`: Lesion category (e.g., gain, mutation, fusion)
`clr`	Optional. A named vector of colors for each lesion type. If not provided, default colors will be assigned using `default.grin.colors`.
`hgt`	Optional. A named numeric vector specifying the proportion (height) of the oncoprint rectangle to be filled for each lesion type. If not provided, proportions will be determined automatically based on average lesion sizes.

Details

In cases where a patient has multiple types of lesions (e.g., gain and mutation) in the same gene, this function ensures that all lesion types are visually represented within a single oncoprint rectangle.

If `hgt` is not specified, lesion types are ranked by their average genomic size (calculated as `loc.end - loc.start + 1`), and the oncoprint proportions are derived accordingly. Smaller lesions (like point mutations) will occupy a smaller portion of the rectangle, while larger lesions (like CNVs) will occupy a larger portion.

Alternatively, the user can manually define the fill proportions via the `hgt` parameter.

Value

A list with the following components:

- `col`: Named vector of colors assigned to each lesion type
- `hgt`: Named vector of normalized proportions to fill for each lesion type
- `legend`: Legend parameters for use in oncoprint plots

Author(s)

Lakshmi Patibandla <LakshmiAnuhya.Patibandla@stjude.org>, Abdelrahman Elsayed <abdelrahman.elsayed@stjude
Stanley Pounds <stanley.pounds@stjude.org>

References

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

Examples

```
data(lesion_data)

# Automatically assign oncoprint proportions based on average lesion size:
onco.props <- onco.print.props(lesion_data)
```

```
# Alternatively, manually specify the oncoprint fill proportions for each lesion type:
onco.props <- onco.print.props(lesion_data,
                        hgt = c("gain" = 4, "loss" = 3, "mutation" = 2, "fusion" = 1))
```

`order.index.gene.data` *Order and Index Gene Annotation Data*

Description

This function orders and indexes gene annotation data by chromosome, gene start, and gene end positions. It is typically used to prepare gene data for overlap analyses with lesion data.

Usage

```
order.index.gene.data(gene.data)
```

Arguments

`gene.data`	A `data.frame` containing gene annotation information, either provided by the user or retrieved using the `get.ensembl.annotation` function from the GRIN2.0 package. The `data.frame` must contain four columns: **"gene"** Ensembl gene ID. **"chrom"** Chromosome on which the gene is located. **"loc.start"** Start position of the gene. **"loc.end"** End position of the gene.

Value

A list with two components:

`gene.data`	The input gene annotation data, ordered by chromosome and genomic coordinates.
`gene.index`	A `data.frame` with two columns (`row.start` and `row.end`) indicating the row indices for genes on each chromosome.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org> and Stanley Pounds <stanley.pounds@stjude.org>

References

Pounds, S., et al. (2013). A genomic random interval model for statistical analysis of genomic lesion data.

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

Examples

```
data(hg38_gene_annotation)

ordered.genes <- order.index.gene.data(hg38_gene_annotation)
```

`order.index.lsn.data` *Order and Index Lesion Data*

Description

This function orders and indexes lesion data by lesion type, chromosome, and subject ID. It prepares lesion data for downstream GRIN analysis by structuring it in a way that facilitates efficient access and overlap computations.

Usage

```
order.index.lsn.data(lsn.data)
```

Arguments

`lsn.data`	A `data.frame` containing lesion data formatted for GRIN. It must include the following five columns: **"ID"** Patient identifier. **"chrom"** Chromosome on which the lesion is located. **"loc.start"** Start position of the lesion. **"loc.end"** End position of the lesion. **"lsn.type"** Lesion type (e.g., gain, loss, mutation, fusion, etc...).

Value

A list with two elements:

`lsn.data`	The input lesion data, ordered by lesion type, chromosome, and subject.
`lsn.index`	A `data.frame` with two columns, `row.start` and `row.end`, indicating the index range of lesions for each subject-lesion type-chromosome combination. For example, if a patient has a single deletion on chromosome 5, `row.start` will equal `row.end`. If there are four deletions, the range will span four rows.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org> and Stanley Pounds <stanley.pounds@stjude.org>

References

Pounds, S., et al. (2013). A genomic random interval model for statistical analysis of genomic lesion data.

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

Examples

```
data(lesion_data)

ordered.lsn <- order.index.lsn.data(lesion_data)
```

`pathways` *Genes Annotated to Biological Pathways*

Description

A dataset listing genes annotated to various biological pathways, as reported in Liu, Yu, et al. (2017).

Usage

```
pathways
```

Format

`pathways`**:**

A data frame with 121 rows and 3 columns:

gene.name Gene symbol.

ensembl.id Ensembl gene ID.

pathway Biological pathway to which the gene is annotated.

Source

Extracted from the supplementary materials of Liu, Yu, et al. (2017) https://www.nature.com/articles/ng.3909#Sec27

`prep.binary.lsn.mtx` *Prepare Binary Lesion Matrix*

Description

Constructs a binary matrix representing the presence or absence of specific lesion types affecting individual genes across patients. Each row corresponds to a gene-lesion type combination, and each column corresponds to a patient.

Usage

```
prep.binary.lsn.mtx(ov.data, min.ngrp = 0)
```

Arguments

ov.data	A list of six data.frame objects representing the output from the find.gene.lsn.overlaps function.
min.ngrp	Optional integer specifying the minimum number of patients that must be affected by a given gene-lesion combination to be retained in the output matrix. The default is 0, which includes all combinations affecting at least one patient.

Details

The function processes the overlap results from find.gene.lsn.overlaps and constructs a binary matrix with dimensions: (gene and lesion type) by patient.

Each row is labeled using the format <gene.ID>_<lesion.type> (e.g., ENSG00000118513_gain for a gain affecting the MYB gene). For each gene-lesion combination, a patient receives a value of 1 if affected by that specific lesion type in the corresponding gene, and 0 otherwise.

Rows representing rare lesions (i.e., affecting fewer patients than min.ngrp) are excluded from the final matrix if min.ngrp > 0.

Value

A binary matrix (as a data.frame) where:

- Rows correspond to gene-lesion combinations (gene.ID_lesion.type).
- Columns correspond to patient IDs.
- Entries are binary: 1 if the patient is affected by such a specific type of lesion in that gene, 0 otherwise.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org>, Stanley Pounds <stanley.pounds@stjude.org>

References

Pounds, S., et al. (2013). A genomic random interval model for statistical analysis of genomic lesion data.

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

prep.gene.lsn.data, find.gene.lsn.overlaps

Examples

```
data(lesion_data)
data(hg38_gene_annotation)

# 1) Prepare gene-lesion input data:
prep.gene.lsn <- prep.gene.lsn.data(lesion_data,
```

```
                                    hg38_gene_annotation)

# 2) Identify gene-lesion overlaps:
gene.lsn.overlap <- find.gene.lsn.overlaps(prep.gene.lsn)

# 3) Create binary lesion matrix including only lesion-gene pairs affecting >= 5 patients:
lsn.binary.mtx <- prep.binary.lsn.mtx(gene.lsn.overlap, min.ngrp = 5)
```

`prep.gene.lsn.data` *Prepare Gene and Lesion Data for GRIN Analysis*

Description

Prepares and indexes gene and lesion data for downstream GRIN (Genomic Random Interval) analysis. This function merges and orders gene and lesion coordinates to support efficient computation of overlaps between genes and all different types of genomic lesions (structural or sequence lesions).

Usage

```
prep.gene.lsn.data(lsn.data, gene.data, mess.freq = 10)
```

Arguments

`lsn.data`	A `data.frame` containing lesion data in GRIN-compatible format. Must include the following five columns: **ID** Unique patient identifier. **chrom** Chromosome on which the lesion is located. **loc.start** Start position of the lesion in base pairs. **loc.end** End position of the lesion in base pairs. **lsn.type** Type of lesion (e.g., gain, loss, mutation, fusion, etc...).
`gene.data`	A `data.frame` containing gene annotation data with the following four required columns: **gene** Ensembl gene ID. **chrom** Chromosome on which the gene is located. **loc.start** Start position of the gene in base pairs. **loc.end** End position of the gene in base pairs.
`mess.freq`	Integer specifying the frequency at which progress messages are displayed. Messages are printed every `mess.freq`-th lesion block processed (default is 10).

Details

This function performs pre-processing by ordering and indexing both gene and lesion data. It combines gene and lesion coordinates into a unified structure, marking each with a specific code (`cty`) that identifies whether the row represents a gene or lesion. This merged data is then used in the `find.gene.lsn.overlaps()` function to detect gene-lesion overlaps.

Value

A list with the following components:

lsn.data Original lesion data.

gene.data Original gene annotation data.

gene.lsn.data Combined and ordered data.frame of gene and lesion intervals. The `cty` column encodes position type: 1 = gene start, 2 = lesion start, 3 = lesion end, 4 = gene end.

gene.index Index data.frame indicating the start and end rows for each chromosome within `gene.lsn.data` for genes.

lsn.index Index data.frame indicating the start and end rows for each lesion (grouped by type, chromosome, and subject) within `gene.lsn.data`.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org> and Stanley Pounds <stanley.pounds@stjude.org>

References

Pounds, S., et al. (2013). A genomic random interval model for statistical analysis of genomic lesion data. Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

`order.index.gene.data`, `order.index.lsn.data`, `find.gene.lsn.overlaps`

Examples

```
data(lesion_data)
data(hg38_gene_annotation)

# Prepare gene and lesion data for GRIN analysis:
prep.gene.lsn <- prep.gene.lsn.data(lesion_data, hg38_gene_annotation)
```

`prep.lsn.type.matrix` *Prepare Lesion Type Matrix*

Description

Constructs a matrix that summarizes the type(s) of lesions affecting each gene across patients. Each row represents a gene, and each column represents a patient.

Usage

```
prep.lsn.type.matrix(ov.data, min.ngrp = 0)
```

Arguments

ov.data	A list of six data.frame objects returned by the find.gene.lsn.overlaps function, containing gene-lesion overlap results.
min.ngrp	Optional integer specifying the minimum number of patients that must be affected by any lesion in a given gene for that gene to be retained in the final matrix. The default is 0, which includes all genes affected by any lesion in at least one patient.

Details

This function produces a matrix with genes as rows and patients as columns. For each gene-patient pair:

- If the patient has no lesion in the gene, the entry is "none".
- If the gene is affected by exactly one type of lesion in the patient (e.g., gain OR mutation), the entry is labeled with the corresponding lesion type.
- If the gene is affected by more than one lesion type in the patient (e.g., gain AND mutation), the entry is labeled as "multiple".

Genes affected in fewer than min.ngrp patients across all lesion types will be excluded if min.ngrp > 0.

Value

A character matrix where:

- Rows correspond to genes (identified by Ensembl gene IDs).
- Columns correspond to patient IDs.
- Entries are "none", a specific lesion type (e.g., "gain", "mutation"), or "multiple".

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org>, Stanley Pounds <stanley.pounds@stjude.org>

References

Pounds, S., et al. (2013). A genomic random interval model for statistical analysis of genomic lesion data.

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

prep.gene.lsn.data, find.gene.lsn.overlaps

Examples

```
data(lesion_data)
data(hg38_gene_annotation)

# 1) Prepare gene and lesion data:
prep.gene.lsn <- prep.gene.lsn.data(lesion_data, hg38_gene_annotation)

# 2) Identify gene-lesion overlaps:
gene.lsn.overlap <- find.gene.lsn.overlaps(prep.gene.lsn)

# 3) Create lesion type matrix for genes affected in >= 5 patients:
lsn.type.mtx <- prep.lsn.type.matrix(gene.lsn.overlap, min.ngrp = 5)
```

---

prob.hits *Find Probability of Locus Hit*

Description

Computes the probability that each genomic locus (e.g., gene or regulatory region) is affected by one or more types of genomic lesions. This function estimates statistical significance for lesion enrichment using a convolution of independent but non-identical Bernoulli distributions.

Usage

```
prob.hits(hit.cnt, chr.size = NULL)
```

Arguments

`hit.cnt`	A list returned by the `count.hits()` function, containing the number of subjects and hits affecting each locus by lesion type.
`chr.size`	A `data.frame` containing chromosome sizes for all 22 autosomes and the X and Y chromosomes. It must include two columns: `"chrom"` for chromosome number, and `"size"` for chromosome lengths in base pairs.

Details

This function estimates a p-value for each locus based on the probability of observing the observed number of lesions (or more) by chance, under a model where lesion events are treated as independent Bernoulli trials.

For each lesion type, the model considers heterogeneity in lesion probability across loci based on their genomic context (e.g., locus size, chromosome size). These probabilities are then combined using a convolution of Bernoulli distributions to estimate the likelihood of observing the actual hit counts.

In addition, the function calculates:

- **FDR-adjusted q-values** using the method of Pounds and Cheng (2006), which estimates the proportion of true null hypotheses.
- **p- and q-values for multi-lesion constellation hits**, i.e., the probability that a locus is affected by one (`p1`), two (`p2`), or more types of lesions simultaneously.

Value

A list with the following components:

gene.hits	A data.frame containing GRIN statistical results. Includes gene annotations, the number of subjects and hits by lesion type, and the computed p-values and FDR-adjusted q-values for lesion enrichment across one or more lesion types.
lsn.data	Original input lesion data.
gene.data	Original input gene annotation data.
gene.lsn.data	A data.frame in which each row corresponds to a gene overlapped by a specific lesion. Includes columns for Ensembl gene ID (gene) and patient/sample ID (ID).
chr.size	Chromosome size information used in the computation.
gene.index	A data.frame indexing rows in gene.lsn.data corresponding to each chromosome.
lsn.index	A data.frame indexing rows in gene.lsn.data corresponding to each lesion.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org> and Stanley Pounds <stanley.pounds@stjude.org>

References

Pounds, S. et al. (2013). A genomic random interval model for statistical analysis of genomic lesion data.

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

prep.gene.lsn.data, find.gene.lsn.overlaps, count.hits

Examples

```
data(lesion_data)
data(hg38_gene_annotation)
data(hg38_chrom_size)

# 1) Prepare gene and lesion data:
prep.gene.lsn <- prep.gene.lsn.data(lesion_data, hg38_gene_annotation)

# 2) Identify overlapping gene-lesion events:
gene.lsn.overlap <- find.gene.lsn.overlaps(prep.gene.lsn)

# 3) Count number of subjects and lesions affecting each gene:
count.subj.hits <- count.hits(gene.lsn.overlap)

# 4) Compute p- and q-values for lesion enrichment per gene:
hits.prob <- prob.hits(count.subj.hits, hg38_chrom_size)
```

`top.alex.waterfall.plots` *Generate Waterfall Plots for Top Significant Genes*

Description

Generates waterfall plots for genes with significant associations between lesion status and expression level, based on the Kruskal Wallis (KW) test results. Only genes with q-values below a user-specified threshold will be plotted.

Usage

```
top.alex.waterfall.plots(out.dir, alex.data, alex.kw.results, q, lsn.data)
```

Arguments

`out.dir`	A character string specifying the output directory where the waterfall plots for selected genes will be saved. Directory must exist or be created by the user prior to running the function.
`alex.data`	A list of three data tables returned by `alex.prep.lsn.expr`: • `"row.mtch"` matching expression and lesion rows by Ensembl gene ID. • `"alex.expr"` matrix of gene expression. • `"alex.lsn"` matrix of lesion status. All matrices must have rows ordered by Ensembl gene ID and columns ordered by patient ID.
`alex.kw.results`	A data table of Kruskal Wallis test results, returned by the `KW.hit.express` function.
`q`	A numeric threshold indicating the maximum allowed KW q-value for a gene to be included in the waterfall plots.
`lsn.data`	Lesion data provided in GRIN-compatible format (as used in `alex.prep.lsn.expr`).

Details

For each gene in the `alex.kw.results` table with a q-value less than or equal to the user-specified `q` threshold, the function generates a waterfall plot displaying the relationship between lesion status and gene expression level. Each plot is saved as a separate PDF file in the `out.dir` folder.

Internally, this function relies on helper functions such as `alex.waterfall.prep` and `alex.waterfall.plot` to prepare and render the plots.

Value

No object is returned to the R environment. The function creates a set of PDF files (one per gene) in the specified `out.dir` directory. Each file contains a labeled waterfall plot illustrating gene expression across different lesion groups.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org>, Stanley Pounds <stanley.pounds@stjude.org>

References

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

alex.prep.lsn.expr, KW.hit.express, alex.waterfall.prep, alex.waterfall.plot

Examples

```
data(expr_data)
data(lesion_data)
data(hg38_gene_annotation)

# 1) Prepare matched expression and lesion matrices:
alex.data <- alex.prep.lsn.expr(expr_data, lesion_data,
                                hg38_gene_annotation,
                                min.expr = 5, min.pts.lsn = 5)
# 2) Run Kruskal Wallis test:
alex.kw.results <- KW.hit.express(alex.data,
                                  hg38_gene_annotation,
                                  min.grp.size = 5)

# 3) Create temporary output folder and generate waterfall plots:
dir.create(resultsFolder <- file.path(tempdir(), "temp.out"))
waterfall.plts <- top.alex.waterfall.plots(out.dir = resultsFolder,
                                           alex.data = alex.data,
                                           alex.kw.results = alex.kw.results,
                                           q = 1e-15,
                                           lsn.data = lesion_data)

# Clean up:
unlink(resultsFolder, recursive = TRUE)
```

---

write.grin.xlsx *Write GRIN Results to Excel File*

Description

Writes GRIN results to an Excel file containing multiple sheets. The output includes the GRIN summary statistics, input data (lesion and gene annotation), chromosome sizes, gene-lesion overlaps, and explanatory metadata to help with the results interpretation.

Usage

```
write.grin.xlsx(grin.result, output.file)
```

Arguments

grin.result	A list returned by the grin.stats function, containing GRIN analysis output.
output.file	A character string specifying the name of the output Excel file. Must end with ".xlsx".

Value

The function creates a multi-sheet Excel file at the specified location. The file contains the following sheets:

- gene.hits: The GRIN results table. Includes gene annotation, number of subjects affected by each lesion type (e.g., gain, loss, mutation), total lesion hits per gene, and associated p-values and FDR-adjusted q-values for the probability of lesion enrichment.
- gene.lsn.data: A table where each row corresponds to a lesion overlapping a specific gene. Columns include "gene" (Ensembl gene ID) and "ID" (patient identifier).
- lsn.data: The input lesion dataset used in the GRIN analysis.
- gene.data: The input gene annotation dataset, typically retrieved from Ensembl.
- chr.size: A table listing the size (in base pairs) of chromosomes 1:22, X, and Y.
- interpretation: A guide to understanding the structure and content of each sheet, with detailed descriptions of columns in the gene.hits results table.
- method.paragraph: A summary of the GRIN methodology, including relevant references for citation.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@stjude.org> and Stanley Pounds <stanley.pounds@stjude.org>

References

Pounds, S., et al. (2013). A genomic random interval model for statistical analysis of genomic lesion data.

Cao, X., Elsayed, A. H., & Pounds, S. B. (2023). Statistical Methods Inspired by Challenges in Pediatric Cancer Multi-omics.

See Also

grin.stats

Examples

```
data(lesion_data)
data(hg38_gene_annotation)
data(hg38_chrom_size)

# Run GRIN analysis using lesion, gene, and chromosome size data:
grin.results <- grin.stats(lesion_data,
                           hg38_gene_annotation,
                           hg38_chrom_size)
```

```
# Write results to an Excel file:
tmp_file <- file.path(tempdir(), "GRIN_Results.xlsx")
write.grin.xlsx(grin.results, output.file = tmp_file)
if (file.exists(tmp_file)) file.remove(tmp_file)
```

Index