

Package ‘GRIN2’

August 22, 2026

Title Genomic Random Interval (GRIN)

Version 2.1.0

Description Implements the Genomic Random Interval (GRIN) framework for identifying genomic loci affected by genomic lesions more frequently than expected by chance. Supports multiple lesion classes, lesion constellation analysis, exon-level target-size modeling, genomic lesion visualization, and gene-level association analyses linking genomic lesions or gene expression with binary and time-to-event clinical outcomes. Includes tools for retrieving versioned GRCh38 Ensembl gene, exon, and regulatory-element annotations. The statistical framework is described in Pounds et al. (2013) <[doi:10.1093/bioinformatics/btt372](https://doi.org/10.1093/bioinformatics/btt372)>.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.2.0)

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

Suggests AnnotationHub, biomaRt, ComplexHeatmap, ensemblDb,
GenomeInfoDb, GenomicRanges, gridExtra, gridGraphics, Gviz,
IRanges, knitr, rmarkdown, S4Vectors

VignetteBuilder knitr, rmarkdown

LazyData true

NeedsCompilation no

Maintainer Abdelrahman Elsayed <aelsayed@stjude.org>

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

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

Config/roxygen2/markdown TRUE

Config/roxygen2/version 8.0.0

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 2026-08-22 15:20:20 UTC

Contents

alex.boxplots	3
alex.pathway	5
alex.prep.lsn.expr	7
alex.waterfall.plot	9
alex.waterfall.prep	11
clin_data	13
compute.gw.coordinates	14
count.hits	15
default.grin.colors	18
example_exon_annotation	19
expr_data	20
find.gene.lsn.overlaps	20
genomewide.log10q.plot	22
genomewide.lsn.plot	24
get.chrom.length	26
get.ensembl.annotation	27
grin.assoc.expr.outcome	30
grin.assoc.lsn.outcome	33
grin.barplt	36
grin.logRank	38
grin.lsn.boundaries	40
grin.oncoprint.mtx	42
grin.results	44
grin.stats	45
grin.stats.lsn.plot	49
hg38_chrom_size	51
hg38_cytoband	51
hg38_exon_chrom_size	52
hg38_gene_annotation	53
KW.hit.express	54
lesion_data	56
lsn.transcripts.plot	57
onco.print.props	61
order.index.gene.data	63
order.index.lsn.data	65
pathways	66
prep.binary.lsn.mtx	67
prep.gene.lsn.data	68
prep.lsn.type.matrix	72

<i>alex.boxplots</i>	3
prob.hits	73
top.alex.waterfall.plots	76
write.grin.xlsx	78
Index	80

alex.boxplots	<i>Generate Box Plots of Gene Expression by Lesion Groups</i>
---------------	---

Description

Generates box plots of gene expression levels stratified by lesion groups for genes selected according to a user-specified q-value threshold from the Kruskal-Wallis lesion-expression association results.

Usage

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

Arguments

out.dir	Path to an existing directory where the resulting PDF files containing the box plots will be saved. One PDF file is generated for each selected gene.
alex.data	Output from the alex.prep.lsn.expr function. A list containing "alex.expr" (expression data), "alex.lsn" (lesion-group data), and "alex.row.mtch" (matched Ensembl gene IDs). The expression and lesion data contain genes in rows and subjects in columns, with matching row and column order.
alex.kw.results	Kruskal-Wallis lesion-expression association results returned by KW.hit.express . The data frame must contain "gene" and "q.KW".
q	Numeric q-value threshold. Genes with $q.KW < q$ are included in the output box plots.
gene.annotation	A gene annotation data frame containing at least "gene" (Ensembl gene ID) and "gene.name" (gene symbol). When a gene symbol is unavailable, the Ensembl gene ID is used to label the corresponding plot.

Details

For each gene with a Kruskal-Wallis q value below the specified threshold, expression values are grouped according to lesion status. Subjects are displayed as individual points together with a box plot showing the expression distribution within each lesion group.

Lesion groups are ordered according to their expression values using [fct_reorder](#). One PDF file is generated for each selected gene and saved in `out.dir`.

Value

Generates one PDF file for each selected gene in `out.dir`. Each file contains a box plot showing gene expression 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"),
           showWarnings = FALSE)

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)
```

<i>alex.pathway</i>	<i>Visualize Lesion and Expression Data by Pathway</i>
---------------------	--

Description

Visualizes lesion and expression data for genes associated with a selected biological pathway. Subjects are ordered by hierarchical clustering based on their lesion profiles across pathway genes, and pathway genes are ordered based on similarities in their lesion profiles.

The function generates two aligned panels showing lesion and expression data and returns the corresponding ordered data.

Usage

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

Arguments

<i>alex.data</i>	Output from the alex.prep.lsn.expr function. A list containing "alex.expr" (expression data), "alex.lsn" (lesion-group data), and "alex.row.mtch" (matched Ensembl gene IDs). The expression and lesion data contain genes in rows and subjects in columns, with matching row and column order.
<i>lsn.clrs</i>	Optional named vector specifying colors for lesion groups. Names must correspond to lesion types represented in the data. If NULL, colors for individual lesion types are assigned using default.grin.colors , with additional colors assigned to "none" and "multiple" groups.
<i>lsn.data</i>	A data frame containing genomic lesion data in GRIN-compatible format. It must contain the columns "ID" (subject ID), "chrom" (chromosome), "loc.start" (lesion start position), "loc.end" (lesion end position), and "lsn.type" (lesion type).
<i>pathways</i>	A data frame containing pathway annotations with the columns "gene.name" (gene symbol), "ensembl.id" (Ensembl gene ID), and "pathway" (pathway name).
<i>selected.pathway</i>	A character string specifying the pathway to visualize. The value must match a pathway in the "pathway" column of pathways.

Details

Genes annotated to `selected.pathway` are identified from pathways, and their lesion and expression data are extracted from `alex.data`.

Pairwise lesion-profile distances are calculated using `dist.lsn`. Hierarchical clustering is then used to order both pathway genes and subjects. Subjects are clustered using lesion profiles across the selected pathway genes. The same subject and gene ordering is used in both visualization panels.

The upper panel displays lesion groups using lesion-specific colors. The lower panel displays standardized gene expression values, with lower expression represented in blue, expression near the gene mean represented in white, and higher expression represented in red.

Gene symbols are used as row labels when available. If a gene symbol is unavailable, the Ensembl gene ID is used instead.

Value

A list containing:

`ordered.path.data`

A data frame containing lesion and expression data for the selected pathway genes. Columns represent subjects in the order determined by hierarchical clustering. Rows contain lesion data followed by expression data and are labeled with `"_lsn"` and `"_expr"`, respectively.

The function also generates a figure with two aligned panels showing lesion and expression data for the selected pathway genes. Both panels use the same hierarchical clustering-based ordering.

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](#), [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 lesion and expression data using the JAK pathway
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
alex.path$ordered.path.data[1:6,1:6]

```

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

Description

Prepares matched gene-level lesion and expression data matrices for use with [KW.hit.express](#), which performs Kruskal-Wallis tests to assess associations between genomic 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 containing normalized gene expression data with genes in rows and subjects in columns. The first column must be named "gene" and contain unique, unversioned Ensembl gene IDs (e.g., "ENSG00000148400"). Ensembl version suffixes such as ".5" in "ENSG00000148400.5" are not supported and should be removed before analysis. Gene symbols should be converted to Ensembl gene IDs before using this function. All remaining columns must correspond to subjects and contain numeric expression measurements; missing values (NA) are allowed.
lsn.data	A data frame containing genomic lesion data in GRIN-compatible format. It must contain the columns "ID" (subject ID), "chrom" (chromosome), "loc.start" (lesion start position), "loc.end" (lesion end position), and "lsn.type" (lesion type; e.g., mutation, gain, deletion, or structural rearrangement).
gene.annotation	A gene annotation data frame, either supplied by the user or retrieved using get.ensembl.annotation . It must contain the columns "gene" (Ensembl gene ID), "chrom" (chromosome), "loc.start" (gene start position), and "loc.end" (gene end position).

min.expr	Optional numeric value specifying the minimum total expression required for a gene to be retained. Total expression is calculated as the sum of expression values across the matched subjects. If NULL, genes are not filtered according to total expression.
min.pts.lsn	Optional numeric value specifying the minimum number of subjects with at least one genomic lesion affecting a gene for that gene to be retained. If NULL, genes are not filtered according to lesion frequency.

Details

Genomic lesions are first mapped to genes using [prep.gene.lsn.data](#) and [find.gene.lsn.overlaps](#). The resulting gene-lesion overlaps are then converted to a gene-by-subject lesion-type matrix using [prep.lsn.type.matrix](#).

Only subjects represented in both the lesion and expression datasets are retained. Genes may additionally be filtered according to `min.expr` and `min.pts.lsn`. The lesion and expression matrices are subsequently restricted to their shared set of genes and ordered identically by Ensembl gene ID and subject ID.

The resulting matched matrices can be supplied directly to [KW.hit.express](#) for lesion-expression association analysis.

Value

A list with the following components:

alex.expr	A numeric matrix of gene expression data with Ensembl gene IDs as row names and subject IDs as column names.
alex.lsn	A data frame containing lesion groups for the same genes and subjects and in the same order as <code>alex.expr</code> .
alex.row.mtch	A data frame containing 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](#), [prep.gene.lsn.data](#), [find.gene.lsn.overlaps](#), [prep.lsn.type.matrix](#), [get.ensembl.annotation](#)

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	<i>Generate Waterfall Plot of Lesion and Expression Data</i>
---------------------	--

Description

Generates a waterfall plot displaying genomic lesions and gene expression levels across subjects for a selected gene. Subjects are grouped according to lesion status and ordered by expression level within each lesion group.

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 "gene.lsn.exp" with subject IDs, lesion groups, and expression values for the selected gene; "lsns" with genomic lesions overlapping the gene; "stats" with the corresponding Kruskal-Wallis lesion-expression association results; and "gene.ID" with the gene symbol or Ensembl gene ID used to label the plot.
lsn.data	A data frame containing genomic lesion data in GRIN-compatible format. It must contain the columns "ID", "chrom", "loc.start", "loc.end", and "lsn.type".
lsn.clrs	Optional named vector specifying colors for lesion groups. Names must correspond to lesion types represented in the data. If NULL, colors for individual lesion types are assigned using default.grin.colors , with additional colors assigned to "none" and "multiple" groups.
delta	Numeric value controlling the genomic spacing around the gene locus displayed in the DNA lesion panel. The default is 0.5.

Details

The left portion of the waterfall plot displays genomic lesions overlapping the selected gene, with lesion types distinguished by color. The genomic coordinates of the gene are indicated by vertical reference lines.

The right portion displays gene expression for the same subjects. Subjects are first grouped alphabetically according to lesion group and then ordered by expression level within each group. For each subject, expression is shown relative to the median expression of the corresponding lesion group.

Colors may be supplied through `lsn.clr.s`. When colors are not supplied, lesion-specific colors are assigned automatically using `default.grin.colors`.

Value

Generates a waterfall plot showing genomic lesion status and gene expression for the selected gene.

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 for use 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 containing "alex.expr" (expression data), "alex.lsn" (lesion-group data), and "alex.row.mtch" (matched Ensembl gene IDs). The expression and lesion data contain genes in rows and subjects in columns, with matching row and column order.
alex.kw.results	Kruskal-Wallis lesion-expression association results returned by KW.hit.express . The data frame must contain "gene" with Ensembl gene IDs and "gene.name" with gene symbols, together with gene coordinates and the Kruskal-Wallis results.
gene	Gene of interest specified by either its gene symbol or Ensembl gene ID. The value must match exactly one gene in alex.kw.results.
lsn.data	A data frame containing genomic lesion data in GRIN-compatible format. It must contain the columns "ID" (subject ID), "chrom" (chromosome), "loc.start" (lesion start position), "loc.end" (lesion end position), and "lsn.type" (lesion type).

Details

The function identifies the selected gene in the Kruskal-Wallis results and extracts its matched lesion-group and expression values across subjects from alex.data. It also identifies all lesions in lsn.data whose genomic coordinates overlap the selected gene.

The gene symbol is used to label the lesion and expression columns when available. If the gene symbol is missing, the Ensembl gene ID is used instead.

clin_data	<i>Example Clinical Dataset for T-cell Acute Lymphoblastic Leukemia (T-ALL)</i>
-----------	---

Description

This example dataset contains clinical and demographic information for 265 newly diagnosed T-cell Acute Lymphoblastic Leukemia (T-ALL) patients and is provided for demonstrating clinical and outcome analyses in GRIN2. The data originate from Liu, Yu, et al. (2017) and include variables describing patient characteristics, treatment response, and clinical outcomes.

Usage

```
clin_data
```

Format

A data frame with 265 rows and 11 columns:

ID Unique patient identifier.

Sex Patient sex.

Race Patient race.

Age_Days Age at diagnosis, in days.

WBC White blood cell count at diagnosis.

MRD29 Measurable residual disease percentage at day 29 post-treatment.

MRD.binary Binary MRD status (0 = MRD $\leq$ 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 event indicator (0 = censored/alive at last follow-up, 1 = death).

efs.time Event-free survival time in years.

efs.censor Event-free survival event indicator (0 = censored, 1 = event).

Source

Data derived from supplementary materials accompanying Liu, Yu, et al. (2017), *Nature Genetics*.
[doi:10.1038/ng.3909](https://doi.org/10.1038/ng.3909)

Additional clinical variables were integrated from the Therapeutically Applicable Research to Generate Effective Treatments (TARGET) program: <https://www.cancer.gov/ccg/research/genome-sequencing/target>.

compute.gw.coordinates

Compute Genome-wide Plotting Coordinates

Description

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

Usage

```
compute.gw.coordinates(grin.res, scl = 1000000)
```

Arguments

<code>grin.res</code>	GRIN results, typically the output of the <code>grin.stats</code> function.
<code>scl</code>	Numeric; number of base pairs represented by one plotting unit. Default is 1,000,000, so genome-wide coordinates are expressed in megabase units.

Details

This function adds `x.start` and `x.end` coordinates that place genomic positions from all chromosomes on a continuous genome-wide plotting axis. The original chromosome-specific genomic coordinates in `loc.start` and `loc.end` are retained unchanged.

Chromosome lengths and genomic positions are scaled according to `scl`. Cumulative chromosome lengths are then used to offset the coordinates of each chromosome so that chromosomes are positioned consecutively along a continuous genome-wide axis.

Specifically:

- The chromosome size table is updated with `x.start` and `x.end` columns defining the genome-wide plotting boundaries of each chromosome.
- Gene coordinates in `gene.data` and `gene.hits` are updated with corresponding `x.start` and `x.end` plotting coordinates.
- Lesion coordinates in `lsn.data` are updated with corresponding `x.start` and `x.end` plotting coordinates.

Value

A GRIN results list containing the original components with genome-wide plotting coordinates added to the relevant chromosome, gene, and lesion data:

gene.hits GRIN gene-level summary statistics with added `x.start` and `x.end` plotting coordinates.

gene.lsn.data Unchanged gene-lesion overlap data.

lsn.data Lesion data with added `x.start` and `x.end` plotting coordinates.

gene.data Gene annotation data with added `x.start` and `x.end` plotting coordinates.

- chr.size** Chromosome size table with added x.start and x.end coordinates defining chromosome positions on the genome-wide plotting axis.
- gene.index** Unchanged chromosome index used to identify chromosome-specific rows in the gene data.
- lsn.index** Unchanged chromosome index used to identify chromosome-specific rows in the lesion 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

[grin.stats](#), [genomewide.lsn.plot](#)

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)

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

count.hits	<i>Count Gene-Lesion Hits</i>
------------	-------------------------------

Description

Computes the number of genomic lesions affecting each gene by lesion type and the number of unique subjects whose lesions overlap each gene for each lesion type.

Usage

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

Arguments

ov.data A list returned by `find.gene.lsn.overlaps()` containing gene-lesion overlap data, processed gene and lesion annotations, and supporting index objects. When exon-level analysis was requested, the list also contains gene- and chromosome-level exon target sizes and the lesion types designated for exon-level analysis.

Details

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

nhit.mtx The total number of overlapping lesions affecting each gene, categorized by lesion type. Multiple lesions of the same type in the same subject are counted separately.

nsubj.mtx The number of unique subjects with at least one overlapping lesion affecting each gene, categorized by lesion type. Multiple lesions of the same type affecting the same gene in one subject are counted once.

For example, if three separate mutations from the same subject overlap *NOTCH1*, all three lesions are counted in **nhit.mtx**, whereas that subject is counted once in **nsubj.mtx**.

Exon-level target sizes do not alter the hit or affected-subject counts. Gene-lesion overlaps are counted using the standard genomic coordinates of each gene, including for lesion types specified in `exon_level`. The gene-level exon target sizes, chromosome-level exon target sizes, and exon-level lesion-type specification are retained in the returned list for downstream GRIN probability calculations.

All genes represented in `gene.data` remain included in **nhit.mtx** and **nsubj.mtx**, regardless of whether they have a valid matching exon annotation. Genes without valid exon target sizes remain available for standard GRIN analyses but do not receive exon-level probability estimates for lesion types specified in `exon_level`.

Value

A list containing the following components:

lsn.data Processed lesion data.

lsn.index A `data.frame` indexing lesion groups defined by lesion type, chromosome, and subject.

gene.data Processed gene annotation data.

gene.index A `data.frame` indexing genes by chromosome.

nhit.mtx A numeric matrix in which rows correspond to genes and columns correspond to lesion types. Each value is the number of lesions of the specified type affecting the gene.

nsubj.mtx A numeric matrix with the same dimensions as **nhit.mtx**. Each value is the number of unique subjects with at least one lesion of the specified type affecting the gene.

gene.lsn.data A `data.frame` in which each row represents a gene overlapped by a genomic lesion.

glp.data The combined gene and lesion position table. The `cty` column identifies the boundary type: 1 = gene start, 2 = lesion start, 3 = lesion end, and 4 = gene end.

- gene.exon.size** Numeric vector containing the total annotated exon target size for each gene, aligned by `gene.row`. Genes without a valid matching exon annotation have a value of NA. These genes remain included in the hit and affected-subject counts and in standard GRIN analyses, but exon-level probability calculations are not performed for lesion types specified in `exon_level`. Returns NULL when exon-level analysis was not requested.
- exon.chrom.size** A `data.frame` containing the genome-wide annotated exon target size for each chromosome. This object is retained for use in downstream exon-level probability calculations. Returns NULL when exon-level analysis was not requested.
- exon_level** Character vector specifying the lesion types designated for exon-level analysis. Returns NULL when exon-level analysis was not requested.

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](#), [prob.hits](#)

Examples

```
data(lesion_data)
data(hg38_gene_annotation)
data(example_exon_annotation)
data(hg38_exon_chrom_size)

# Prepare gene and lesion data using the optional arguments
# for exon-level analysis
prep.gene.lsn <- prep.gene.lsn.data(
  lsn.data = lesion_data,
  gene.data = hg38_gene_annotation,
  exons.annotation = example_exon_annotation,
  exon.chrom.size = hg38_exon_chrom_size,
  exon_level = "mutation"
)

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

# Count lesions and affected subjects for each gene and lesion type
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

<code>lsn.types</code>	A character vector of lesion types, typically obtained from the "lsn.type" column of lesion data. Duplicate lesion types are automatically removed and the remaining lesion types are sorted before colors are assigned.
------------------------	--

Details

This function assigns colors from a predefined palette of up to 10 colors. Colors are assigned to unique lesion types in alphabetical order. If more than 10 unique lesion types are provided, custom colors must be specified manually in the plotting function.

Value

A named character vector in which names correspond to the unique lesion types and values correspond to their assigned colors.

Author(s)

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

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)
```

`example_exon_annotation`*Example Human Exon Annotation Data*

Description

Example human GRCh38 exon annotation data corresponding to all genes included in the `hg38_gene_annotation` example dataset. The dataset provides exon-level genomic coordinates that can be used for GRIN2 analyses of genomic lesion types restricted to exonic regions, such as coding single nucleotide variants (SNVs) and small insertions/deletions (indels). Exons were extracted from a single representative transcript for each gene, defined as the MANE Select transcript for protein-coding genes and the Ensembl canonical transcript for non-protein-coding genes.

Usage

```
data(example_exon_annotation)
```

Format

A data frame with 5,691 rows and 6 columns:

gene Ensembl gene identifier of the gene to which each annotated exon belongs.

chrom Chromosome on which the exon is located.

loc.start Exon start position in base pairs.

loc.end Exon end position in base pairs.

gene_name Official gene symbol.

transcript_exon Representative transcript identifier and exon number (e.g., `ENST00000257818_exon1`).

Source

Retrieved from the Ensembl BioMart database using `get.ensembl.annotation()` with the "Human-GRCh38" genome assembly (hg38). Exons were obtained from the MANE Select transcript for protein-coding genes and the Ensembl canonical transcript for non-protein-coding genes. The dataset contains exon annotations for all genes included in the `hg38_gene_annotation` example dataset.

 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). This dataset illustrates the gene expression data format used by GRIN2 expression-based analyses.

Usage

```
expr_data
```

Format

```
expr_data:
```

A data frame with 417 rows and 265 columns:

gene Unique, unversioned Ensembl gene IDs for the 417 genes included in the dataset.

... Each remaining column represents a T-ALL patient, with the column name corresponding to the subject ID and containing numeric log2-normalized gene 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 `prep.gene.lsn.data()`. The function detects instances in which a genomic lesion spans or intersects the genomic coordinates of a gene.

Usage

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

Arguments

gl.data A list returned by `prep.gene.lsn.data()` containing processed and indexed gene and lesion data. When exon-level analysis was requested, the list also contains gene- and chromosome-level exon target sizes and the lesion types designated for exon-level analysis.

Details

The function scans the combined gene and lesion position table generated by `prep.gene.lsn.data()` and identifies lesions that overlap the genomic coordinates of each gene. Each detected gene-lesion overlap is recorded for subsequent counting and statistical analysis.

Gene-lesion overlap detection is unchanged when `exon_level` is specified and continues to use the complete genomic coordinates of each gene. Exon-level gene and chromosome target sizes do not alter whether a lesion is considered to overlap a gene. Instead, these objects are carried forward for use in downstream GRIN probability calculations. Lesion types specified in `exon_level` should therefore contain only exonic lesions, as described in `prep.gene.lsn.data()`.

All genes represented in `gene.data` are included in overlap detection, regardless of whether they have a matching exon annotation. Genes without a valid exon target size remain available for standard GRIN analyses but will not receive probability estimates for lesion types specified in `exon_level`.

Value

A list containing the following components:

lsn.data Processed lesion data.

gene.data Processed gene annotation data.

gene.lsn.data A `data.frame` ordered by chromosome and genomic position that contains both gene and lesion boundaries. The `cty` column identifies the position type: 1 = gene start, 2 = lesion start, 3 = lesion end, and 4 = gene end.

gene.lsn.hits A `data.frame` in which each row represents a gene overlapped by a genomic lesion. It contains the gene and lesion row indices, gene identifier and coordinates, patient or sample identifier, lesion coordinates, and lesion type.

gene.index A `data.frame` indexing the rows corresponding to genes on each chromosome.

lsn.index A `data.frame` indexing lesion groups defined by lesion type, chromosome, and subject.

gene.exon.size Numeric vector containing the total annotated exon target size for each gene, aligned by `gene.row`. Genes without a valid matching exon annotation have a value of NA. These genes remain available for standard GRIN analyses but are excluded from probability calculations for lesion types specified in `exon_level`. Returns NULL when exon-level analysis was not requested.

exon.chrom.size A `data.frame` containing the genome-wide annotated exon target size for each chromosome. This object is carried forward for use in downstream exon-level probability calculations. Returns NULL when exon-level analysis was not requested.

exon_level Character vector specifying the lesion types designated for exon-level analysis. Returns NULL when exon-level analysis was not requested.

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](#), [count.hits](#), [prob.hits](#)

Examples

```
data(lesion_data)
data(hg38_gene_annotation)
data(example_exon_annotation)
data(hg38_exon_chrom_size)

# Prepare gene and lesion data using the optional arguments
# for exon-level analysis
prep.gene.lsn <- prep.gene.lsn.data(
  lsn.data = lesion_data,
  gene.data = hg38_gene_annotation,
  exons.annotation = example_exon_annotation,
  exon.chrom.size = hg38_exon_chrom_size,
  exon_level = "mutation"
)

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

genomewide.log10q.plot

Genome-wide $-\log_{10}(q\text{-value})$ Plot

Description

Generates a genome-wide plot of $-\log_{10}(q\text{-values})$ for each annotated gene or lesion boundary evaluated by GRIN. Statistical significance can be displayed for one or more selected lesion types.

Usage

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

Arguments

<code>grin.res</code>	GRIN results object (output from <code>grin.stats</code>) generated using either gene annotation or lesion boundaries as marker input.
<code>lsn.grps</code>	A character vector specifying the lesion type(s) to include in the plot.
<code>lsn.colors</code>	A named vector of colors corresponding to the selected lesion types. If <code>NULL</code> , colors are automatically assigned using <code>default.grin.colors</code> .
<code>max.log10q</code>	Numeric; optional maximum value of $-\log_{10}(\text{q-value})$ displayed on the plot. Values greater than <code>max.log10q</code> are capped at this value. If <code>NULL</code> , the plotting limit is determined automatically from the observed finite $-\log_{10}(\text{q-values})$.

Details

This function displays the genome-wide statistical significance of lesions affecting annotated genomic markers. Depending on the marker data supplied to `grin.stats`, these markers may represent genes or lesion boundaries.

The function first adds continuous genome-wide plotting coordinates using `compute.gw.coordinates` when these coordinates are not already present in `grin.res`.

Chromosomes are arranged consecutively along the vertical axis. For each selected lesion type, a horizontal line is drawn at the genomic position of each affected marker. Line length represents the corresponding $-\log_{10}(\text{q-value})$, with longer lines indicating greater statistical significance, and line color identifies the lesion type.

Value

Generates a genome-wide $-\log_{10}(\text{q-value})$ plot on the active graphics device and invisibly returns `NULL`. Chromosomes are displayed along the vertical axis, while the horizontal axis represents $-\log_{10}(\text{q-value})$. Each horizontal line corresponds to an affected gene or lesion boundary, with line length representing statistical significance and line color indicating lesion type.

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](#), [grin.lsn.boundaries](#), [genomewide.lsn.plot](#), [compute.gw.coordinates](#), [default.grin.colors](#)

Examples

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

```
# Use lesion boundaries as genomic markers for gain lesions
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)

# Plot genome-wide significance of gain lesion boundaries
genomewide.log10q.plot(GRIN.results.gain.bound,
                      lsn.grps = "gain",
                      lsn.colors = c("gain" = "red"),
                      max.log10q = 10)

# Gene annotation can also be used as the marker input to grin.stats instead
# of lesion boundaries.
# Multiple lesion types can be displayed together by including their names
# in lsn.grps.
```

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

Description

Generates a genome-wide lesion plot displaying genomic lesions across chromosomes and patients, together with lesion recurrence and statistical significance from the GRIN analysis.

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, patients are displayed according to the order specified in pt.order. If FALSE (default), patient IDs are ordered alphabetically.
pt.order	A data frame with two columns: "ID" containing patient identifiers matching those in the lesion data, and "pts.order" containing the numeric plotting order from 1 to the total number of patients. Required only when ordered = TRUE.
lsn.colors	A named vector of colors assigned to lesion types. Names must correspond to lesion types in the GRIN results. If NULL, colors are automatically assigned using default.grin.colors.

`max.log10q` Numeric; optional maximum value of $-\log_{10}(\text{q-value})$ displayed in the left panel. Values greater than `max.log10q` are capped at this value. If NULL, the plotting limit is determined automatically from the observed finite $-\log_{10}(\text{q-values})$.

Details

The function first uses `compute.gw.coordinates` to convert chromosome-specific genomic coordinates into continuous genome-wide coordinates used for plotting lesions across the genome.

The resulting plot contains three aligned panels. The center panel displays genomic lesions across patients and chromosomes. Each patient occupies one column, and lesions are colored according to lesion type. The left panel displays the $-\log_{10}(\text{q-value})$ for each gene-lesion type combination from the GRIN analysis, whereas the right panel displays the corresponding number of affected patients.

By default, patients are ordered alphabetically by patient ID. A user-defined patient order can instead be supplied through `pt.order` by setting `ordered = TRUE`.

Value

Generates a genome-wide lesion plot on the active graphics device and invisibly returns NULL. The plot consists of three aligned panels:

- Left panel: Displays the statistical significance of lesion recurrence for each gene and lesion type as $-\log_{10}(\text{q-value})$. Horizontal lines are colored by lesion type, with longer lines representing greater statistical significance.
- Middle panel: Displays the genome-wide distribution of lesions across chromosomes and patients. Each column represents a patient, chromosomes are arranged consecutively along the vertical axis, and lesions are colored according to lesion type.
- Right panel: Displays the number of patients affected by each lesion type at each gene. Horizontal lines are colored by lesion type, with longer lines representing a greater number of affected patients.

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](#), [compute.gw.coordinates](#), [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)

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

# Generate plot using a user-defined patient order, such as grouping patients
patient.ids <- sort(unique(lesion_data$ID))
pt.order <- data.frame(ID = patient.ids,
                      pts.order = seq_along(patient.ids))

genomewide.lsn.plot(grin.results,
                    ordered = TRUE,
                    pt.order = pt.order,
                    max.log10q = 50)

```

get.chrom.length

Get Chromosome Length

Description

Retrieves chromosome size data for the 22 autosomes and X and Y chromosomes of the human GRCh38 (hg38) genome assembly using UCSC chromosome information accessed through 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 retrieves chromosome size information for the GRCh38 (hg38) genome assembly using `circlize::read.chromInfo()`. The returned data are restricted to the 22 autosomes and X and Y chromosomes and are ordered from chromosome 1 through chromosome 22, followed by X and Y. Chromosome identifiers are returned without the "chr" prefix.

Value

A data.frame with 24 rows and the following columns:

chrom Chromosome identifier (1-22, X, or Y).

size Chromosome size in base pairs.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@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
```

Retrieve Ensembl Annotation Data

Description

Retrieves pre-generated and versioned Ensembl annotation resources for use with GRIN2. Annotation files are downloaded from the GRIN2 annotation repository, verified using MD5 checksums, and cached locally by default for subsequent analyses.

Usage

```
get.ensembl.annotation(
  genome.assembly = "Human_GRCh38",
  ensembl.version = 110L,
  annotation.type = c("gene", "exon", "regulatory", "all"),
  cache = TRUE,
  cache.dir = NULL,
  force.download = FALSE,
  quiet = FALSE
)
```

Arguments

genome.assembly	Character string specifying the genome assembly. Currently supported: "Human_GRCh38".
ensembl.version	Integer specifying the Ensembl release. Currently supported: 110.
annotation.type	Character string specifying the annotation resource to retrieve. One of "gene", "exon", "regulatory", or "all". Specifying "all" retrieves all three annotation resources.
cache	Logical indicating whether downloaded annotation files should be cached locally. Default is TRUE. Cached files are verified against their expected MD5 checksums before use.
cache.dir	Optional character string specifying the directory in which annotation files should be cached. If NULL, the standard GRIN2 user cache directory is used.
force.download	Logical indicating whether annotation files should be downloaded again even when valid cached copies are available. Default is FALSE.
quiet	Logical indicating whether download and status messages should be suppressed. Default is FALSE.

Details

GRIN2 uses pre-generated, versioned annotation resources to provide reproducible genomic analyses without requiring a live connection to Ensembl BioMart during annotation retrieval. This function downloads resources from the GRIN2 annotation repository when a valid cached copy is unavailable, when `force.download = TRUE`, or when a cached file fails checksum verification. Internet access is therefore required for the initial download. Once a valid resource has been cached, it can be reused without an internet connection.

The available annotation resources include:

Gene annotation

Gene-level annotation based on the specified Ensembl release. The resource includes protein-coding genes as well as non-coding gene biotypes, including long non-coding RNAs (lncRNAs), microRNAs (miRNAs), small nuclear RNAs (snRNAs), small nucleolar RNAs (snoRNAs), pseudogenes, and other annotated gene types. Each gene is represented by its Ensembl gene identifier and genomic interval, together with gene name, biotype, strand, chromosome band, and gene description

where available. The columns `gene`, `chrom`, `loc.start`, and `loc.end` provide the genomic interval required by GRIN2.

Exon annotation

The exon annotation represents one selected transcript per gene. For protein-coding genes, the MANE Select transcript is used when available. MANE (Matched Annotation from NCBI and EMBL-EBI) Select identifies a representative transcript with matching exon structure and sequence between Ensembl/GENCODE and RefSeq. For genes without a selected MANE transcript, including non-coding genes, the Ensembl Canonical transcript is used as the representative transcript. The Ensembl Canonical transcript is the representative transcript designated by Ensembl for a gene.

This resource is used primarily for exon-level GRIN2 analyses, in which selected lesion types, such as mutations and indels, can be evaluated using exon-based rather than whole-gene target sizes when the analysis is restricted to protein-altering alterations. It can be supplied to [grin.stats](#) through the `exons.annotation`, `exon.chrom.size`, and `exon_level` arguments.

This resource is used primarily for exon-level GRIN2 analyses, in which selected lesion types are evaluated using exon-based rather than whole-gene target sizes. It can be supplied together with chromosome-level exon target sizes through the `exons.annotation`, `exon.chrom.size`, and `exon_level` arguments used in the GRIN2 analysis workflow.

The resource contains the individual exons belonging to the selected transcript for each gene and includes exon genomic start and end coordinates, exon number, Ensembl exon identifier, transcript identifier, transcript biotype, transcript start and end coordinates, canonical-transcript status, MANE Select status, and gene-level annotation including gene name, gene boundaries, gene biotype, description, strand, and chromosome band. In this resource, `loc.start` and `loc.end` specifically represent the genomic start and end coordinates of each exon.

Regulatory annotation

Regulatory annotation based on the Ensembl Regulatory Build. The current GRCh38/Ensembl 110 resource contains predicted enhancer regions, predicted promoter regions, CTCF-binding sites, and open chromatin regions, together with gene intervals used to establish relationships between regulatory elements and nearby genes.

Regulatory features are additionally annotated with genes whose genomic intervals overlap the regulatory feature, the nearest upstream and downstream genes, their Ensembl gene identifiers, and the genomic distance between each regulatory feature and the corresponding nearest gene. The resource also reports genes located within 500 kb upstream and within 500 kb downstream of each regulatory feature. These fields facilitate interpretation of regulatory regions that may influence nearby genes even when the regulatory element does not directly overlap a gene.

Annotation resources are downloaded only when required. When `cache = TRUE`, a successfully downloaded and checksum-verified file is stored in the GRIN2 user cache and reused in subsequent calls. Annotation files are verified against their expected checksums before use. If a cached file fails checksum verification, a fresh copy is downloaded and verified. A newly downloaded file that does not match the expected checksum is rejected and the function returns an error.

Value

If `annotation.type` is "gene", "exon", or "regulatory", the corresponding annotation data frame is returned. If `annotation.type = "all"`, a named list with elements `gene`, `exon`, and `regulatory` is returned.

Examples

```
## Not run:
# Requires internet access if a valid cached copy is unavailable.
gene.annotation <- get.ensembl.annotation(
  genome.assembly = "Human_GRCh38",
  ensembl.version = 110,
  annotation.type = "gene"
)

head(gene.annotation)

## End(Not run)
```

```
grin.assoc.expr.outcome
```

Association Between Gene Expression and Clinical Outcomes

Description

Performs gene-level association analyses between gene expression and clinical outcomes. Time-to-event outcomes stored as `Surv` objects are analyzed using Cox proportional hazards regression, while binary outcomes coded as 0 and 1 are analyzed using logistic regression.

For each clinical outcome, a separate regression model is fitted for every gene to evaluate the association between that gene's expression and the outcome. Models can optionally be adjusted for one or more clinical covariates.

Usage

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

Arguments

<code>expr.mtx</code>	A data frame containing gene expression data with genes in rows and subjects in columns. The first column must be named "gene" and contain unique, unversioned Ensembl gene IDs (e.g., "ENSG00000148400"). Ensembl version suffixes such as ".5" in "ENSG00000148400.5" are not supported and should be removed before analysis. Gene symbols should be converted to Ensembl gene IDs before using this function. All remaining columns must correspond to subjects and contain numeric expression values.
<code>clin.data</code>	A data frame containing clinical information. The data frame must contain a column named ID with subject identifiers that correspond to the subject identifiers in <code>expr.mtx</code> .

annotation.data	A gene annotation data frame containing a column named <code>gene</code> with unversioned Ensembl gene IDs matching those in the <code>gene</code> column of <code>expr.mtx</code> . Annotation information is merged with the association results using these gene IDs.
clinvars	A character vector specifying the clinical outcome variables to analyze. Time-to-event outcomes must be stored in <code>clin.data</code> as <code>Surv</code> objects created using <code>survival::Surv()</code> . Binary outcomes must be numeric and coded as 0 and 1.
covariate	Optional character vector specifying one or more clinical covariates to include in the regression models. Covariates may be categorical or numeric. If <code>NULL</code> , models are fitted without covariate adjustment.

Details

Subject identifiers in the expression data and clinical data are matched and reordered before analysis.

For time-to-event outcomes stored as `Surv` objects, a separate Cox proportional hazards model is fitted for each gene using `coxph`. The reported hazard ratio represents the relative change in hazard associated with a one-unit increase in the expression value used as input to the model.

Cox models that generate convergence warnings are excluded from downstream inference. The hazard ratio, confidence interval, p value, and q value for these genes are returned as NA, and a warning reports the number of models that did not converge. When categorical covariates are included, sparse categories or categories with few or no outcome events may lead to model convergence problems. Users should review the distribution of covariates and outcome events when a large number of adjusted models fail to converge.

For binary outcomes coded numerically as 0 and 1, a separate logistic regression model is fitted for each gene using `glm` with `family = "binomial"`. The reported odds ratio represents the change in the odds of the outcome coded as 1 associated with a one-unit increase in expression.

Numeric outcomes containing values other than 0 and 1 are not analyzed and generate a warning. Continuous outcomes should be analyzed using an appropriate regression model outside this function.

When `covariate` is provided, the specified covariates are included together with gene expression in each Cox proportional hazards or logistic regression model.

P values are adjusted for multiple testing using the Benjamini-Hochberg false discovery rate procedure together with the Pounds and Cheng estimator of the proportion of tests having a true null hypothesis: $\pi.\hat{h}at = \min(1, 2 * \text{mean}(p))$.

Cox proportional hazards models assume proportional hazards. Users should evaluate this assumption when interpreting genes of particular interest.

Value

A data frame containing gene annotation information together with outcome-specific association results.

For time-to-event outcomes, results include:

- Hazard ratio (HR).
- Lower bound of the 95 percent confidence interval.

- Upper bound of the 95 percent confidence interval.
- Cox proportional hazards model p value.
- Multiple-testing-adjusted q value.

For binary outcomes, results include:

- Odds ratio (OR).
- Lower bound of the 95 percent confidence interval.
- Upper bound of the 95 percent confidence interval.
- Logistic regression p value.
- Multiple-testing-adjusted q value.

Cox models that do not converge are retained in the output with NA values for their model statistics.

Author(s)

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

References

- Cox, D. R. (1972). Regression Models and Life-Tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2), 187-202.
- Nelder, J. A., & Wedderburn, R. W. M. (1972). Generalized Linear Models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3), 370-384.
- Pounds, S., & Cheng, C. (2006). Robust estimation of the false discovery rate. *Bioinformatics*, 22(16), 1979-1987.

Examples

```
# Load the example datasets
data(expr_data)
data(clin_data)
data(hg38_gene_annotation)

# Create the event-free survival object
clin_data$EFS <- survival::Surv(clin_data$efs.time,
                                clin_data$efs.censor)

# Specify the survival endpoint
clinvars <- c("EFS")

# Run Cox proportional hazards models
coxph.efs <- grin.assoc.expr.outcome(
  expr.mtx = expr_data,
  clin.data = clin_data,
  annotation.data = hg38_gene_annotation,
  clinvars = clinvars
)
```

```
# Run Cox proportional hazards models with covariate adjustment
coxph.efs.adj <- grin.assoc.expr.outcome(
  expr.mtx = expr_data,
  clin.data = clin_data,
  annotation.data = hg38_gene_annotation,
  clinvars = clinvars,
  covariate = "WBC"
)

# A binary outcome coded as 0 and 1 can also be analyzed
clinvars <- c("MRD.binary")

logistic.mrd <- grin.assoc.expr.outcome(
  expr.mtx = expr_data,
  clin.data = clin_data,
  annotation.data = hg38_gene_annotation,
  clinvars = clinvars
)
```

grin.assoc.lsn.outcome

Associate Binary Genomic Lesions with Clinical Outcomes

Description

Performs association analyses between binary genomic lesion events and clinical outcomes. Time-to-event outcomes stored as `Surv` objects are analyzed using Cox proportional hazards regression, while binary outcomes coded as 0 and 1 are analyzed using logistic regression.

Each row of the input lesion matrix represents a specific gene-lesion event, such as a mutation, deletion, gain, or rearrangement affecting a particular gene. Separate models are fitted for each gene-lesion event and each clinical outcome.

Usage

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

Arguments

<code>lsn.mtx</code>	A binary lesion matrix, typically generated using <code>prep.binary.lsn.mtx</code> . Rows represent gene-lesion events (e.g., <code>ENSG00000148400_mutation</code>) and columns represent subjects. Values must be coded as 1 when a subject harbors the specified lesion and 0 otherwise. Missing values are permitted.
----------------------	--

<code>clin.data</code>	A clinical data frame containing a column named <code>ID</code> with subject identifiers corresponding to the column names of <code>lsn.mtx</code> .
<code>annotation.data</code>	A gene annotation data frame containing a column named <code>gene</code> . Annotation information is merged with the final results.
<code>clinvars</code>	A character vector specifying the clinical outcome variables to analyze. Time-to-event outcomes must be stored in <code>clin.data</code> as <code>Surv</code> objects created using <code>survival::Surv()</code> . Binary outcomes must be numeric and coded as 0 and 1.
<code>covariate</code>	Optional character vector specifying one or more clinical covariates to include in the regression models. Covariates may be categorical or numeric. If <code>NULL</code> , models are fitted without covariate adjustment.

Details

Subject identifiers in the lesion matrix and clinical data are matched and reordered before analysis. For time-to-event outcomes, a separate Cox proportional hazards model is fitted for each binary gene-lesion event using `coxph`. The reported hazard ratio compares subjects harboring the lesion with subjects who do not harbor the lesion.

Cox models in which no outcome events occur in one of the two lesion groups are not fitted because the lesion effect cannot be reliably estimated. These gene-lesion events are retained in the output with NA values for the hazard ratio, confidence interval, p value, and q value. A warning reports the number of models excluded for this reason.

Cox models that otherwise generate convergence warnings are also retained in the output with NA model statistics, and the number of additional non-converged models is reported separately.

For binary outcomes coded as 0 and 1, a separate logistic regression model is fitted for each gene-lesion event using `glm` with `family = "binomial"`. The reported odds ratio compares subjects harboring the lesion with subjects who do not harbor the lesion for the outcome coded as 1.

Logistic regression models that fail to converge or produce warnings consistent with complete or near-complete separation are returned with NA model statistics, and the number of affected models is reported to the user.

When `covariate` is provided, the specified covariates are included together with the binary lesion indicator in each Cox proportional hazards or logistic regression model.

P values are adjusted for multiple testing using the false discovery rate procedure together with the Pounds and Cheng estimator of the proportion of tests having a true null hypothesis: $\hat{p}_i = \min(1, 2 * \text{mean}(p))$. Models with missing p values are excluded from the multiple-testing adjustment.

The output also reports the numbers of subjects with and without each lesion according to clinical outcome status.

Cox proportional hazards models assume proportional hazards. Users should evaluate this assumption when interpreting gene-lesion associations of particular interest.

Value

A data frame containing gene annotation information, the corresponding gene-lesion identifier, and outcome-specific association statistics.

For time-to-event outcomes, results include:

- Hazard ratio (HR).
- Lower bound of the 95 percent confidence interval.
- Upper bound of the 95 percent confidence interval.
- Cox proportional hazards model p value.
- Multiple-testing-adjusted q value.
- Numbers of subjects with and without the lesion according to event status.

For binary outcomes, results include:

- Odds ratio (OR).
- Lower bound of the 95 percent confidence interval.
- Upper bound of the 95 percent confidence interval.
- Logistic regression p value.
- Multiple-testing-adjusted q value.
- Numbers of subjects with and without the lesion according to binary outcome status.

Models that cannot be reliably estimated or do not converge are retained in the output with NA values for their model statistics.

Author(s)

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

References

- Cox, D. R. (1972). Regression Models and Life-Tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2), 187-202.
- Nelder, J. A., & Wedderburn, R. W. M. (1972). Generalized Linear Models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3), 370-384.

See Also

[prep.binary.lsn.mtx](#), [coxph](#), [glm](#)

Examples

```
# Load the example datasets
data(lesion_data)
data(hg38_gene_annotation)
data(clin_data)

# Prepare gene-level lesion data
gene.lsn <- prep.gene.lsn.data(
  lesion_data,
  hg38_gene_annotation
)

# Identify overlaps between genomic lesions and genes
```

```

gene.lsn.overlap <- find.gene.lsn.overlaps(gene.lsn)

# Create a binary lesion matrix
lsn.binary.mtx <- prep.binary.lsn.mtx(
  gene.lsn.overlap,
  min.ngrp = 5
)

# Create the event-free survival object
clin_data$EFS <- survival::Surv(
  clin_data$efs.time,
  clin_data$efs.censor
)

# Analyze a binary and a survival outcome
clinvars <- c("MRD.binary", "EFS")

lsn.outcomes <- grin.assoc.lsn.outcome(
  lsn.mtx = lsn.binary.mtx,
  clin.data = clin_data,
  annotation.data = hg38_gene_annotation,
  clinvars = clinvars
)

# Models may also be adjusted for clinical covariates
lsn.outcomes.adj <- grin.assoc.lsn.outcome(
  lsn.mtx = lsn.binary.mtx,
  clin.data = clin_data,
  annotation.data = hg38_gene_annotation,
  clinvars = clinvars,
  covariate = "Sex"
)

```

grin.barplt

GRIN Lesion Stacked Bar Plot

Description

Generates a horizontal stacked bar plot showing the number of patients affected by different genomic lesion types across a user-specified set of genes, based on GRIN analysis results.

Usage

```

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

```

Arguments

<code>grin.res</code>	GRIN results object, typically the output from the <code>grin.stats</code> function.
<code>count.genes</code>	A character vector of gene names to include in the plot. Only genes present in the GRIN results are displayed.
<code>lsn.colors</code>	A named vector of colors assigned to lesion types. If NULL, colors are automatically assigned using <code>default.grin.colors</code> .

Details

The function extracts the number of patients affected by each lesion type from `grin.res$gene.hits` for the genes specified in `count.genes`.

Each horizontal bar represents a gene and is divided into segments corresponding to different lesion types. The length of each segment represents the number of patients affected by that lesion type, and the number of affected patients is displayed within each non-zero segment.

Genes are ordered according to the total number of lesion-type-specific affected-patient counts across the lesion categories displayed in the plot. This visualization can be used to compare the relative burden and distribution of different genomic lesion types across candidate driver genes or other genes of interest.

Value

A ggplot object containing a horizontal stacked bar plot. Each bar represents a selected gene, bar segments represent lesion types, and segment lengths represent the corresponding number of affected patients.

Author(s)

Abdelrahman Elsayed <abdelrahman.elsayed@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](#), [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 genes of interest to include in the stacked bar plot
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 stacked bar plot showing the distribution of lesion types
# across the selected genes
grin.barplt(grin.results,
  count.genes)
```

grin.logRank	<i>Log-Rank Test for Associations Between Genomic Lesions and Survival Outcomes</i>
--------------	---

Description

Performs gene-level log-rank tests to evaluate associations between genomic lesions and time-to-event outcomes. For each gene, subjects are grouped according to their lesion status and survival distributions are compared using the log-rank test.

Usage

```
grin.logRank(lsn.mtx,
  clin.data,
  annotation.data,
  clinvars,
  min.grp.size = NULL)
```

Arguments

lsn.mtx	A gene-by-subject lesion matrix, typically generated using prep.lsn.type.matrix . Rows represent genes and columns represent subjects. For each gene-subject combination, entries indicate lesion status: "none" if the gene is not affected, a specific lesion type (e.g., "mutation" or "fusion") if the gene is affected by one lesion type, or "multiple" if the gene is affected by two or more distinct lesion types in the same subject.
clin.data	A data frame containing clinical information. The data frame must contain a column named ID with subject identifiers that correspond to the subject identifiers in lsn.mtx.
annotation.data	A gene annotation data frame containing a column named gene with unversioned Ensembl gene IDs matching the gene IDs used as row names in lsn.mtx. Annotation information is merged with the final association results using these gene IDs.

<code>clinvars</code>	A character vector specifying the time-to-event clinical variables to analyze. Each variable must be stored in <code>clin.data</code> as a Surv object created using <code>survival::Surv()</code> .
<code>min.grp.size</code>	Optional numeric value specifying the minimum number of subjects required in each lesion-status group for a gene to be analyzed. Groups may include "none", individual lesion types such as "mutation" or "fusion", and "multiple". Groups containing fewer than <code>min.grp.size</code> subjects are excluded from the analysis for that gene. A gene is tested only if at least two groups remain after filtering.

Details

Subject identifiers in the lesion matrix and clinical data are matched and reordered before analysis.

For each time-to-event outcome specified in `clinvars`, the function performs a log-rank test for each gene using [survdif](#) with $\rho = 0$.

P values are adjusted for multiple testing using the Benjamini-Hochberg false discovery rate procedure together with the Pounds and Cheng estimator of the proportion of tests having a true null hypothesis: $\pi.\text{hat} = \min(1, 2 * \text{mean}(p))$.

The output also reports the number of subjects with and without an event within each lesion group.

Value

A data frame containing gene annotation information together with:

- `logRank_<endpoint>_pval`: Gene-level log-rank test p value for each survival endpoint specified in `clinvars`.
- `logRank_<endpoint>_qval`: Multiple-testing-adjusted q value for each survival endpoint.
- Numbers of subjects with an event in each lesion group.
- Numbers of subjects without an event in each lesion group.

Author(s)

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

References

Mantel, N. (1966). Evaluation of survival data and two new rank order statistics arising in its consideration. *Cancer Chemotherapy Reports*, 50(3), 163-170.

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B*, 57(1), 289-300.

Pounds, S., & Cheng, C. (2006). Robust estimation of the false discovery rate. *Bioinformatics*, 22(16), 1979-1987.

See Also

[prep.lsn.type.matrix](#), [Surv](#), [survdif](#)

Examples

```
# Load the example datasets
data(lesion_data)
data(clin_data)
data(hg38_gene_annotation)

# Prepare gene-level lesion data
gene.lsn <- prep.gene.lsn.data(lesion_data,
                              hg38_gene_annotation)

# Identify overlaps between genomic lesions and genes
gene.lsn.overlap <- find.gene.lsn.overlaps(gene.lsn)

# Create the lesion-type matrix
gene.lsn.type.mtx <- prep.lsn.type.matrix(gene.lsn.overlap,
                                          min.ngrp = 5)

# Create the event-free survival object
clin_data$EFS <- survival::Surv(clin_data$efs.time,
                                clin_data$efs.censor)

# Run gene-level log-rank tests
logRank.efs <- grin.logRank(lsn.mtx = gene.lsn.type.mtx,
                           clin.data = clin_data,
                           annotation.data = hg38_gene_annotation,
                           clinvars = "EFS",
                           min.grp.size = 3)
```

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

Description

Generates genome-wide genomic boundaries defined by the unique start and end positions of interval-based lesions, such as copy-number gains or deletions, for use as marker regions in GRIN analyses.

Usage

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

Arguments

lsn.data	A lesion data table containing interval-based lesions, typically copy-number gains or deletions. The data may include related subtypes analyzed together, such as gains and amplifications or heterozygous and homozygous deletions.
chrom.size	A chromosome size table with two required columns: "chrom" containing chromosome identifiers and "size" containing chromosome sizes in base pairs.

Details

The function partitions each chromosome into non-overlapping genomic boundaries using all unique lesion start and end positions observed in the supplied lesion data. A new boundary begins at each lesion start position and immediately after each lesion end position.

As a result, large lesions may be divided into multiple smaller boundaries when other lesions begin or end within the same genomic region. This allows GRIN to evaluate recurrent lesion patterns at a finer resolution than the original lesion intervals.

Boundaries span the entire chromosome, including regions outside observed lesions. The first boundary begins at position 1, and the final boundary extends to the end of the chromosome.

This approach is particularly useful for copy-number variation analyses in which recurrent genomic regions are evaluated independently of existing gene or feature annotations.

Value

A `data.frame` with five columns:

gene Unique boundary identifier constructed from chromosome, start position, and end position.

chrom Chromosome containing the boundary.

loc.start Start position of the boundary in base pairs.

loc.end End position of the boundary in base pairs.

diff Length of the boundary 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

[grin.stats](#), [genomewide.log10q.plot](#)

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 genome-wide 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 approach can be applied to deletions or related
# copy-number subtypes analyzed together.

```

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

Description

Creates a gene-by-subject lesion matrix from GRIN analysis results for use with the `oncoPrint()` function from the **ComplexHeatmap** package. Each matrix cell contains the lesion type or lesion types affecting a selected gene in a given subject.

Usage

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

Arguments

<code>grin.res</code>	A GRIN results object, typically generated by <code>grin.stats</code> . The object must contain <code>gene.lsn.data</code> with the columns <code>gene</code> , <code>ID</code> , and <code>lsn.type</code> , and <code>gene.data</code> with the columns <code>gene</code> and <code>gene.name</code> .
<code>oncoprint.genes</code>	A character vector of Ensembl gene IDs specifying the genes to include in the OncoPrint. Genes are retained in the order supplied by the user when lesion records are available.

Details

The function filters `grin.res$gene.lsn.data` to the requested genes and constructs a gene-by-subject character matrix. Rows represent genes, columns represent subjects, and each matrix cell contains the lesion type or lesion types observed for that gene-subject combination. Multiple lesion types are separated by semicolons.

Gene symbols from `grin.res$gene.data` are used as row labels when available. If a gene symbol is missing or empty, the corresponding Ensembl gene ID is used instead. If multiple Ensembl gene IDs share the same gene symbol, row labels are made unique automatically.

Genes without lesion records in `grin.res$gene.lsn.data` are not included in the returned matrix.

The resulting matrix can be supplied directly to `oncoPrint`.

Value

A character matrix with genes as rows and subjects as columns. Matrix entries contain semicolon-separated lesion types for affected gene-subject combinations and empty strings for unaffected combinations. Row names contain gene symbols when available and otherwise Ensembl gene IDs.

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 genes to include in the OncoPrint
oncoprint.genes <- c(
  "ENSG00000148400",
  "ENSG00000171862",
  "ENSG00000171843",
  "ENSG00000156531"
)

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

# Genes can also be selected directly from GRIN results
genes.const <- grin.results$gene.hits[
  grin.results$gene.hits$q2.nsubj < 0.01,
]

selected.genes <- genes.const$gene

oncoprint.mtx <- grin.oncoprint.mtx(
  grin.results,
  selected.genes
)
```

grin.results

*Example GRIN2 Analysis Results***Description**

A precomputed GRIN2 results object generated from the example genomic lesion dataset included with the package. The object represents a standard gene-level GRIN2 analysis and is provided for use in examples of downstream analysis, visualization, and export functions without repeatedly running `grin.stats()`.

Format

A named list containing the following 10 components:

`gene.hits` A data frame containing the primary GRIN2 analysis results for each gene, including lesion counts, affected-subject counts, lesion-specific probabilities, p-values, q-values, and constellation test statistics.

`lsn.data` The genomic lesion dataset used as input to `grin.stats`. Each row corresponds to a genomic lesion together with its genomic coordinates, lesion type, and subject identifier.

`gene.data` The gene annotation data used during the analysis, including genomic coordinates and gene identifiers.

`gene.lsn.data` A data frame describing overlaps between genomic lesions and genes.

`chr.size` Chromosome target sizes used for probability calculations.

`gene.index` Internal index used to efficiently identify chromosome-specific groups of genes during the analysis.

`lsn.index` Internal index used to efficiently identify chromosome-specific groups of genomic lesions during the analysis.

`gene.exon.size` Gene-specific exon target sizes used for exon-level analyses. This component is NULL for the standard gene-level example.

`exon.chrom.size` Chromosome exon target sizes used for exon-level analyses. This component is NULL for the standard gene-level example.

`exon_level` Indicates which lesion types were modeled using exon-level target sizes. This component is NULL for the standard gene-level example.

Details

The object was generated using:

```
grin.results <- grin.stats(
  lsn.data = lesion_data,
  gene.data = hg38_gene_annotation,
  chr.size = hg38_chrom_size
)
```

This dataset is intended primarily for package examples and demonstrations. It allows downstream GRIN2 functions to be illustrated without repeatedly running `grin.stats`. The object contains the complete output returned by `grin.stats` and can therefore be supplied directly to functions that require a GRIN2 results object.

Source

Generated from the example datasets `lesion_data`, `hg38_gene_annotation`, and `hg38_chrom_size` included with the GRIN2 package.

See Also

`grin.stats`, `lsn.transcripts.plot`, `grin.oncoprint.mtx`, `write.grin.xlsx`

Examples

```
data("grin.results")

# Display the available components.
names(grin.results)

# View the first few rows of the primary results table.
head(grin.results$gene.hits)
```

`grin.stats`*Execute the GRIN Statistical Framework*

Description

Executes the Genomic Random Interval (GRIN) statistical framework to determine whether genomic loci, such as genes or regulatory regions, are affected by genomic lesions more frequently than expected by chance. The function evaluates the enrichment of individual lesion types and, when multiple lesion types are present, constellations of complementary lesion types affecting the same locus.

By default, GRIN models each lesion type using the complete genomic interval of each locus and the complete size of the corresponding chromosome. Optionally, selected lesion types can be evaluated using exon-level gene and chromosome target sizes when the analysis is restricted to exonic lesions.

Usage

```
grin.stats(
  lsn.data,
  gene.data,
  chr.size,
  exons.annotation = NULL,
  exon.chrom.size = NULL,
  exon_level = NULL
)
```

Arguments

- lsn.data** A `data.frame` containing genomic lesion data in GRIN-compatible format. The following five columns are required:
- ID** Sample, subject, or patient identifier.
 - chrom** Chromosome on which the lesion is located.
 - loc.start** Genomic start coordinate of the lesion in base pairs.
 - loc.end** Genomic end coordinate of the lesion in base pairs.
 - lsn.type** Lesion category, such as mutation, breakpoint, gain, amplification, or deletion.
- For single nucleotide variants, `loc.start` and `loc.end` should generally be identical. Copy-number alterations should be represented by their genomic boundaries, whereas structural rearrangements should generally be represented by separate rows for their individual breakpoints.
- gene.data** A `data.frame` containing the genomic annotation for the loci to be analyzed. The following four columns are required:
- gene** Gene or locus identifier, typically an Ensembl gene identifier.
 - chrom** Chromosome on which the locus is located.
 - loc.start** Genomic start coordinate of the locus in base pairs.
 - loc.end** Genomic end coordinate of the locus in base pairs.
- The rows in `gene.data` define the locus universe analyzed by GRIN.
- chr.size** A `data.frame` containing complete chromosome lengths used for lesion types evaluated with the standard GRIN model. It must contain `chrom`, the chromosome identifier, and `size`, the chromosome length in base pairs. Chromosome identifiers must be represented consistently across the input datasets.
- exons.annotation** An optional `data.frame` containing exon coordinates used to calculate gene-level exon target sizes. This argument is required when `exon_level` is specified. Exons should be provided from one representative transcript per gene to avoid counting alternative exons from multiple transcripts more than once. We recommend using the MANE Select transcript for protein-coding genes and the Ensembl canonical transcript for non-protein-coding genes. Compatible exon annotation data for supported genome assemblies can be obtained using [get.ensembl.annotation](#). The following columns are required:
- gene** Gene identifier matching the identifier system used in `gene.data`. Multiple rows may represent different exons of the same gene.
 - chrom** Chromosome on which the exon is located.
 - loc.start** Exon start coordinate in base pairs.
 - loc.end** Exon end coordinate in base pairs.
- When a user-provided `exons.annotation` is used, a valid matching exon annotation must be provided for every gene in `gene.data`. Genes without a valid matching exon annotation remain available for standard GRIN analyses, but exon-level probabilities are not calculated for those genes.

<code>exon.chrom.size</code>	An optional <code>data.frame</code> containing the genome-wide exon target size for each chromosome. This argument is required when <code>exon_level</code> is specified and must contain <code>chrom</code> and <code>size</code> . A pre-generated dataset for GRCh38 is provided with the package as <code>hg38_exon_chrom_size</code> and can be loaded using <code>data(hg38_exon_chrom_size)</code> . These chromosome-level target sizes should be derived using the same genome assembly and representative-transcript selection procedure as <code>exons.annotation</code> .
<code>exon_level</code>	An optional character vector specifying lesion types that should be modeled using exon-level gene and chromosome target sizes. Each value must exactly match a category in <code>lsn.data\$lsn.type</code> . Lesions belonging to these types must be restricted to exonic regions before running GRIN. For example, when mutations are specified for exon-level analysis, exonic variants such as missense, nonsense, frameshift, and synonymous mutations may be included, whereas intronic and other non-coding mutations should be excluded before analysis. Lesion types not included in <code>exon_level</code> use the standard GRIN model. The default is <code>NULL</code> .

Details

GRIN uses a null probability model in which each lesion is assumed to occur uniformly along the chromosome on which it was observed. Under this model, the overlap of each lesion with a locus of interest is an independent Bernoulli trial with success probability proportional to the sum of the lengths of the lesion and the locus and inversely proportional to the chromosome length.

For each subject, the probabilities of all lesions belonging to that subject are combined to calculate the probability that at least one of the subject's lesions affects the locus. These subject-specific probabilities are convolved across subjects to obtain the null probability distribution for the number of subjects with at least one lesion affecting the locus. Statistical significance is determined by comparing the observed number of affected subjects with this null distribution. GRIN also evaluates the significance of the observed total number of lesion hits. P-values are adjusted for multiple testing using the Benjamini-Hochberg false discovery rate procedure together with the Pounds and Cheng estimator of the proportion of true null hypotheses, $\pi_{\text{hat}} = \min(1, 2 * \text{mean}(p))$.

The genomic target space depends on the lesion type. For lesion types not specified in `exon_level`, GRIN uses the complete locus interval from `gene.data` and complete chromosome length from `chr.size`. For lesion types specified in `exon_level`, GRIN instead uses the summed exon length of each gene from `exons.annotation` and the genome-wide exon target size of the chromosome from `exon.chrom.size`. This provides an exon-specific target space for analyses restricted to exonic variants.

When multiple lesion types are present, GRIN additionally calculates ordered constellation p-values and q-values to evaluate evidence that a locus is affected by one, two, or more complementary lesion types.

Value

A list containing the following components:

gene.hits A `data.frame` containing one row per gene or locus, including lesion-hit counts (`nhit.*`), affected-subject counts (`nsubj.*`), and their corresponding lesion-specific p-values and q-values (`p.nhit.*`, `q.nhit.*`, `p.nsubj.*`, and `q.nsubj.*`). When multiple lesion types are

present, ordered constellation p-values and q-values ($p1.*$, $p2.*$, ..., $q1.*$, $q2.*$, ...) are also returned.

lsn.data Input lesion data with the internal `lsn.row` index added.

gene.data Input gene or locus annotation data with the internal `gene.row` index added.

gene.lsn.data A `data.frame` containing detected gene-lesion overlaps and their corresponding gene, lesion, and subject information.

chr.size Chromosome-size reference used for standard GRIN calculations.

gene.index A `data.frame` indexing genes or loci by chromosome.

lsn.index A `data.frame` indexing lesion groups by lesion type, chromosome, and subject.

gene.exon.size Numeric vector containing gene-level exon target sizes. Returns `NULL` when exon-level analysis was not requested.

exon.chrom.size A `data.frame` containing chromosome-level exon target sizes. Returns `NULL` when exon-level analysis was not requested.

exon_level Character vector identifying lesion types modeled using exon-level target sizes. Returns `NULL` when exon-level analysis was not requested.

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.

Pounds, S., and Cheng, C. (2006). Robust estimation of the false discovery rate.

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](#), [get.ensembl.annotation](#)

Examples

```
data(lesion_data)
data(hg38_gene_annotation)
data(hg38_chrom_size)
data(example_exon_annotation)
data(hg38_exon_chrom_size)

# Example 1: Run GRIN using standard gene and chromosome target sizes
grin.results <- grin.stats(
  lsn.data = lesion_data,
  gene.data = hg38_gene_annotation,
  chr.size = hg38_chrom_size
)
```

```
# Example 2: Model mutations using exon-level target sizes
grin.exon.results <- grin.stats(
  lsn.data = lesion_data,
  gene.data = hg38_gene_annotation,
  chr.size = hg38_chrom_size,
  exons.annotation = example_exon_annotation,
  exon.chrom.size = hg38_exon_chrom_size,
  exon_level = "mutation"
)
```

grin.stats.lsn.plot *GRIN Lesion and Statistics Plot*

Description

Generates a regional plot showing genomic lesions overlapping a selected feature together with its GRIN statistics. This function is particularly designed for regulatory and other genomic features that do not have transcript or exon structure to display.

Usage

```
grin.stats.lsn.plot(grin.res,
  feature = NULL,
  lsn.clrs = NULL,
  expand = 0.0005)
```

Arguments

grin.res	GRIN results for genes, regulatory elements, or other genomic features, as returned by the grin.stats function.
feature	Character string specifying the feature of interest. This is typically an Ensembl regulatory feature ID or another regulatory-region identifier used as marker input to grin.stats. An Ensembl gene ID can also be provided.
lsn.clrs	A named vector of colors assigned to lesion types. If NULL, colors are automatically assigned using default.grin.colors.
expand	Numeric; proportion of the feature length added upstream and downstream to define the plotting region. Default is 0.0005. Setting expand = 0 restricts the plot to the exact feature boundaries.

Details

This function provides a regional visualization of lesions and GRIN statistics without displaying transcript or exon structure. It is therefore particularly useful for regulatory features, such as promoters, enhancers, CTCF binding sites, open chromatin regions, or other genomic elements for which a transcript panel is not applicable.

Genes can also be plotted with this function when only the lesion distribution and GRIN statistics are needed. However, for gene-centered visualization that includes transcript and exon structure, lsn.transcripts.plot is generally more informative.

The plot contains a regional lesion panel followed by statistical summaries:

- The upper panel displays individual genomic lesions overlapping the selected feature. Each row represents one overlapping lesion event, and lesions are colored according to lesion type. Dashed vertical lines mark the start and end coordinates of the selected feature.
- The first statistical summary reports, for each lesion type, the number of affected subjects together with the corresponding $-\log_{10}(p)$ and $-\log_{10}(q)$ values from the GRIN lesion-recurrence analysis.
- The second statistical summary reports the constellation $-\log_{10}(p)$ and $-\log_{10}(q)$ values across lesion orders evaluated by GRIN.

Value

Generates a regional lesion and GRIN statistics plot on the active graphics device and invisibly returns NULL. No transcript or exon annotation panel is included.

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](#), [lsn.transcripts.plot](#), [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)

# Plot lesions and GRIN statistics for a gene without displaying
# transcript or exon structure
grin.stats.lsn.plot(grin.results,
                    feature = "ENSG00000184937")

# This function is particularly useful when regulatory elements or other
# non-transcript genomic features are used as marker input to grin.stats.
# For gene-centered plots that include transcript and exon annotation,
# use lsn.transcripts.plot().
```

hg38_chrom_size	<i>Chromosome Length Data (GRCh38)</i>
-----------------	--

Description

This dataset contains chromosome lengths, in base pairs, for the 22 autosomes and the X and Y chromosomes of the human GRCh38 (hg38) genome assembly. The data were retrieved from UCSC chromosome information using the `get.chrom.length` function with "Human_GRCh38" as the selected genome assembly.

Usage

```
hg38_chrom_size
```

Format

```
hg38_chrom_size:
```

A data frame with 24 rows and 2 columns:

chrom Chromosome identifier (1-22, X, or Y).

size Chromosome length in base pairs.

Source

UCSC chromosome information for the GRCh38 (hg38) genome assembly, retrieved using `get.chrom.length("Human_GRCh38")` through `circize::read.chromInfo()`.

hg38_cytoband	<i>GRCh38 Chromosome Cytobands</i>
---------------	------------------------------------

Description

This dataset contains the genomic coordinates of cytogenetic bands (cytobands) for the 22 autosomes and the X and Y chromosomes of the human GRCh38 (hg38) genome assembly. The dataset is primarily used by `lcn.transcripts.plot` to provide chromosome cytoband information for regional lesion and transcript visualization.

Usage

```
hg38_cytoband
```

Format

hg38_cytoband:

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

chrom Chromosome identifier (1-22, X, or Y).

chromStart Start coordinate of the cytoband in base pairs, using the UCSC 0-based coordinate convention.

chromEnd End coordinate of the cytoband in base pairs, using the UCSC end-exclusive coordinate convention.

name Cytoband name (e.g., p11.1 or q22.3).

gieStain Giemsa staining classification used for cytogenetic visualization (e.g., gneg, gpos, acen).

Source

UCSC Genome Browser cytoband data for the GRCh38 (hg38) genome assembly: <https://hgdownload.soe.ucsc.edu/goldenPath/hg38/database/>

See Also

[lsn.transcripts.plot](#)

hg38_exon_chrom_size *Human GRCh38 Chromosome-Level Exon Target Sizes*

Description

A chromosome-level summary of the total annotated exon target size in the human GRCh38 genome assembly. For each chromosome, the exon lengths from the selected representative transcript of every annotated gene were combined to provide the chromosome-wide exonic search space used in GRIN2 exon-level probability calculations.

Representative transcripts were defined as the MANE Select transcript for protein-coding genes and the Ensembl canonical transcript for non-protein-coding genes.

Usage

```
data(hg38_exon_chrom_size)
```

Format

A data frame with one row per chromosome and 2 columns:

chrom Chromosome identifier.

size Total annotated exon target size on the chromosome in base pairs, calculated by aggregating the lengths of all exons from the representative transcript selected for each gene. Representative transcripts correspond to the MANE Select transcript for protein-coding genes and the Ensembl canonical transcript for non-protein-coding genes. This value represents the chromosome-wide exonic target space used in GRIN2 exon-level probability calculations.

Details

This dataset provides the chromosome-level denominator used for lesion types designated for exon-level analysis in GRIN2. It represents the total exonic target space on each chromosome and is therefore distinct from `example_exon_annotation`, which contains exon coordinates for only a subset of genes used in package examples.

The dataset should be used together with exon annotations generated from the same genome assembly and representative-transcript selection procedure.

Source

Derived from human GRCh38 exon annotations retrieved from the Ensembl BioMart database using `get.ensembl.annotation()`. Exons were obtained from the MANE Select transcript for protein-coding genes and the Ensembl canonical transcript for non-protein-coding genes. Exon lengths were aggregated for all genes on each chromosome.

hg38_gene_annotation	<i>Example Gene Annotation Data</i>
----------------------	-------------------------------------

Description

This dataset contains gene annotation data for 417 selected genes from the human GRCh38 genome assembly. The genes correspond to those included in the example gene expression dataset and are provided to support package examples and demonstrations without requiring retrieval of the complete genome-wide annotation dataset.

Usage

```
hg38_gene_annotation
```

Format

`hg38_gene_annotation`:

A data frame with 417 rows and 9 columns:

gene Unversioned Ensembl gene ID.

chrom Chromosome identifier on which the gene is located.

loc.start Gene start position in base pairs.

loc.end Gene end position in base pairs.

description Gene description.

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

chrom.band Chromosomal cytoband containing the gene.

Details

The annotation data were obtained using `get.ensembl.annotation` for the "Human_GRCh38" genome assembly and subsequently restricted to the 417 genes represented in the example expression dataset.

Source

Gene annotation data for the human GRCh38 genome assembly obtained using `get.ensembl.annotation("Human_GRCh38")` and restricted to the 417 genes represented in the example gene expression dataset.

See Also

[get.ensembl.annotation](#)

KW.hit.express	<i>Associate Lesion Groups with Gene Expression</i>
----------------	---

Description

Performs the Kruskal-Wallis test to evaluate associations between genomic lesion groups and expression levels of the corresponding genes.

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 containing: • alex.expr: Gene expression data with genes represented by Ensembl gene IDs in rows and subjects in columns. • alex.lsn: Lesion-group data for the same genes and subjects and in the same order as alex.expr. • alex.row.mtch: A data frame containing the matched Ensembl gene IDs from the expression and lesion data.
gene.annotation	A gene annotation data frame, either supplied by the user or retrieved using get.ensembl.annotation . It must contain the columns "gene" (Ensembl gene ID), "chrom" (chromosome), "loc.start" (gene start position), and "loc.end" (gene end position).
min.grp.size	Optional numeric value specifying the minimum number of subjects required in a lesion group for that group to be included in the Kruskal-Wallis test. For a gene to be tested, at least two groups must each contain at least min.grp.size subjects. Genes that do not satisfy this requirement are retained in the output with an NA Kruskal-Wallis p value.

Details

For each matched gene, the function compares expression values across the lesion groups defined in `alex.lsn` using the Kruskal-Wallis test. Expression values are obtained from the corresponding gene in `alex.expr`.

Subjects without a genomic lesion affecting the gene are represented by the lesion group "none". Subjects affected by more than one lesion type in the same gene may be represented by the group "multiple", as defined during preparation of the lesion matrix.

In addition to the Kruskal-Wallis p value, the function reports the number of subjects and the mean, median, and standard deviation of expression for each lesion group represented in the input data.

Kruskal-Wallis p values are adjusted for multiple testing using the Benjamini-Hochberg false discovery rate procedure implemented by [p.adjust](#).

Value

A data frame containing gene annotation information and lesion-expression association results. Each row corresponds to a matched gene. Results include:

- `p.KW`: Kruskal-Wallis test p value.
- `q.KW`: FDR-adjusted q value.
- Columns ending in `_n.subjects`: number of subjects in each lesion group.
- Columns ending in `_mean`: mean expression within each lesion group.
- Columns ending in `_median`: median expression within each lesion group.
- Columns ending in `_sd`: standard deviation of expression within each lesion group.

Genes that do not meet the minimum group-size requirement are retained with NA values for `p.KW` and `q.KW`.

Author(s)

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

References

Kruskal, W. H., & Wallis, W. A. (1952). Use of Ranks in One-Criterion Variance Analysis. *Journal of the American Statistical Association*, 47(260), 583-621.

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 (genes x subjects)
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

This dataset contains genomic lesions identified in 265 patients with newly diagnosed T-cell acute lymphoblastic leukemia (T-ALL), as reported by Liu, Yu, et al. (2017). The dataset includes copy-number alterations, single nucleotide variants, and structural rearrangements and is provided as an example lesion dataset for 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 identifier on which the lesion is located.

loc.start Lesion start position in base pairs (GRCh38).

loc.end Lesion end position in base pairs (GRCh38).

lsn.type Type of genomic lesion.

Details

The original genomic coordinates were based on the GRCh37 (hg19) human genome assembly. Coordinates were converted to the GRCh38 (hg38) genome assembly using the UCSC LiftOver tool before inclusion in GRIN2. All genomic coordinates in this dataset therefore correspond to GRCh38.

Source

Adapted from the supplementary data of Liu, Yu, et al. (2017), Nature Genetics. Original GRCh37 genomic coordinates were converted to GRCh38 using the UCSC LiftOver tool.

Liu, Y., et al. (2017). The genomic landscape of pediatric and young adult T-lineage acute lymphoblastic leukemia. Nature Genetics.

UCSC LiftOver: <https://genome.ucsc.edu/cgi-bin/hgLiftOver>

lsn.transcripts.plot *Plot Genomic Lesions with Transcript Annotations*

Description

Creates an integrated genomic-region plot combining patient-level genomic lesions with optional transcript annotations, chromosome context, genomic coordinates, lesion-type legends, and gene-level GRIN2 statistics.

The function supports both gene-centered plots and user-defined genomic regions. Single-position lesions, or lesion types explicitly selected by the user, are drawn at their exact genomic coordinates with user-controlled marker size.

Usage

```
lsn.transcripts.plot(  
  grin.res,  
  gene = NULL,  
  chrom = NULL,  
  plot.start = NULL,  
  plot.end = NULL,  
  transTrack = TRUE,  
  show.ideogram = TRUE,  
  show.axis = TRUE,  
  show.transcript.labels = FALSE,  
  transcript.mode = c("all", "selected"),  
  selected.transcript = NULL,  
  lesion.grp = NULL,  
  lsn.clrs = NULL,  
  spec.lsn.clr = NULL,  
  point.lesion.groups = NULL,  
  point.size.mm = 0.8,  
  expand = 0.0005,  
  hg38.transcripts = NULL,  
  hg38.cytoband = NULL,  
  genome = "hg38",  
  transcript.height = 0.31,  
  draw = TRUE  
)
```

Arguments

<code>grin.res</code>	A GRIN2 results object containing at least <code>lsn.data</code> . Gene-centered plots additionally require <code>gene.data</code> ; gene-level recurrence and constellation statistics are displayed when <code>gene.hits</code> is available.
<code>gene</code>	Optional character scalar specifying the gene to plot. When supplied, genomic coordinates are obtained from <code>grin.res[["gene.data"]]</code> .
<code>chrom</code>	Chromosome for a user-defined regional plot. Used only when <code>gene = NULL</code> . Values with or without the <code>chr</code> prefix are accepted.
<code>plot.start, plot.end</code>	Numeric start and end coordinates for a user-defined regional plot. Both are required when <code>gene = NULL</code> .
<code>transTrack</code>	Logical; whether to display transcript annotations.
<code>show.ideogram</code>	Logical; whether to display a chromosome ideogram.
<code>show.axis</code>	Logical; whether to display a genomic-coordinate axis.
<code>show.transcript.labels</code>	Logical; whether transcript or gene labels are displayed within the transcript track.
<code>transcript.mode</code>	Character string specifying which transcript annotations to display. Use "all" for all overlapping transcripts or "selected" for transcript IDs supplied through <code>selected.transcript</code> .
<code>selected.transcript</code>	Character vector of transcript IDs retained when <code>transcript.mode = "selected"</code> . Version suffixes are ignored during matching, allowing identifiers such as <code>ENST00000342505</code> and <code>ENST00000342505.8</code> to match.
<code>lesion.grp</code>	Optional character vector specifying lesion types to plot.
<code>lsn.clrs</code>	Optional named character vector assigning colors to lesion types. Vector names must match the lesion-type values in <code>grin.res[["lsn.data"]][lsn.type]</code> . If an unnamed vector is supplied, it must contain exactly one color for each plotted lesion type. When <code>lsn.clrs = NULL</code> , colors are assigned automatically.
<code>spec.lsn.clr</code>	Optional color used when a single lesion type is selected through <code>lesion.grp</code> .
<code>point.lesion.groups</code>	Optional character vector specifying lesion types to display as point markers. When <code>NULL</code> , each lesion is classified from its genomic coordinates: lesions with identical start and end coordinates are displayed as circles, whereas lesions spanning multiple positions are displayed as intervals.
<code>point.size.mm</code>	Positive numeric scalar specifying the diameter, in millimeters, of lesions displayed as circles. The default is 0.8.
<code>expand</code>	Non-negative numeric proportion used to expand a gene-centered plotting interval upstream and downstream. It is ignored for user-defined regional plots, which use the supplied coordinates exactly.
<code>hg38.transcripts</code>	An <code>EnsDb</code> transcript-annotation object compatible with <code>ensemldb::getGeneRegionTrackForGviz()</code> . Required when <code>transTrack = TRUE</code> . A compatible GRCh38 transcript annotation can be obtained through AnnotationHub using the workflow shown in the examples.

hg38.cytoband	Cytoband annotation supplied to <code>Gviz::IdeogramTrack()</code> . Required when <code>show.ideogram = TRUE</code> . A GRCh38 cytoband annotation is provided with GRIN2 as <code>hg38_cytoband</code> and can be loaded using <code>data(hg38_cytoband)</code> .
genome	Genome identifier passed to the Gviz annotation tracks.
transcript.height	Relative vertical space allocated to the transcript annotation track. Increasing this value provides additional space when many transcripts are displayed. This value is used only when <code>transTrack = TRUE</code> .
draw	Logical; if TRUE, draw the assembled plot on a new graphics page. The plot object is returned invisibly in either case.

Details

In gene mode, the displayed interval is derived from the matching entry in `gene.data` and can be expanded using `expand`. In regional mode, the exact interval specified by `chrom`, `plot.start`, and `plot.end` is used.

Transcript annotations can be shown in full or restricted to selected transcript IDs. Selected IDs are matched after removal of version suffixes. Unmatched IDs are reported in a warning when at least one requested transcript is available; the function stops with an informative error when none of the requested transcripts can be found.

Point-like lesions are plotted at their exact genomic start coordinates and use physical marker sizes measured in millimeters. Lesions are ordered first by lesion type and then by interval size, with source order used to break exact ties.

When transcript annotations are not displayed, the vertical space allocated to the ideogram and genomic-coordinate axis is reduced automatically so that the lesion panel occupies the available plotting area more efficiently.

For gene-centered plots, available lesion-specific subject counts and transformed p- and q-values are displayed together with ordered constellation statistics from `gene.hits`. Missing statistics are omitted rather than preventing plot construction.

Value

Invisibly returns a `gtable/grob` object representing the assembled figure. When `draw = TRUE`, the figure is also drawn.

See Also

```
grin.stats(), Gviz::GeneRegionTrack(), Gviz::GenomeAxisTrack(), Gviz::IdeogramTrack(),
ensemldb::getGeneRegionTrackForGviz()
```

Examples

```
data("grin.results")
data("hg38_cytoband")

# Plot lesions across a user-defined genomic region without transcript
# annotations. This example uses only data included with GRIN2.
```

```

lsn.transcripts.plot(
  grin.res = grin.results,
  chrom = 9,
  plot.start = 19900000,
  plot.end = 25600000,
  transTrack = FALSE,
  show.ideogram = FALSE,
  lesion.grp = "loss",
  spec.lsn.clr = "blue"
)

# Plot all lesion types across a larger genomic region without transcript
# annotations.
lsn.transcripts.plot(
  grin.res = grin.results,
  chrom = 9,
  plot.start = 1,
  plot.end = 141000000,
  transTrack = FALSE,
  show.ideogram = FALSE,
  point.size.mm = 1
)

## Not run:
# Transcript-track examples require an EnsDb annotation object retrieved
# through AnnotationHub and therefore require an internet connection.
ah <- AnnotationHub::AnnotationHub()

# Ensembl release 110 transcript annotations for human GRCh38.
hg38.transcripts <- ah[["AH113665"]]

# Define colors for the lesion types in the example dataset.
lesion.colors <- c(
  fusion = "black",
  gain = "red",
  loss = "blue",
  mutation = "olivedrab"
)

# Plot all transcripts and genomic lesions affecting JAK3.
lsn.transcripts.plot(
  grin.res = grin.results,
  gene = "JAK3",
  transcript.mode = "all",
  lsn.clrs = lesion.colors,
  hg38.transcripts = hg38.transcripts,
  hg38.cytoband = hg38_cytoband
)

# Display selected JAK1 transcripts and increase point-marker size.
lsn.transcripts.plot(
  grin.res = grin.results,

```

```

    gene = "JAK1",
    transcript.mode = "selected",
    selected.transcript = c(
      "ENST00000342505",
      "ENST00000672751"
    ),
    lsn.clrs = lesion.colors,
    point.size.mm = 0.7,
    transcript.height = 0.40,
    hg38.transcripts = hg38.transcripts,
    hg38.cytoband = hg38_cytoband
  )

# Plot transcript annotations and genomic lesions across a user-defined
# region while displaying gene names in the transcript track.
lsn.transcripts.plot(
  grin.res = grin.results,
  chrom = 9,
  plot.start = 21900000,
  plot.end = 21990000,
  transcript.mode = "all",
  show.transcript.labels = TRUE,
  lsn.clrs = lesion.colors,
  transcript.height = 0.46,
  hg38.transcripts = hg38.transcripts,
  hg38.cytoband = hg38_cytoband
)

## End(Not run)

```

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)

Lesion type names in `lsn.type` must be valid R variable names (e.g., mutation, fusion, or gain) and must not contain spaces or other special characters.

<code>clr</code>	Optional. A named vector of colors for each lesion type. User-specified colors are preserved and matched to lesion types by name. If not provided, default colors are assigned using <code>default.grin.colors</code> .
<code>hgt</code>	Optional. A named numeric vector specifying the proportion (height) of the oncoprint rectangle to be filled for each lesion type. If not provided, proportions are 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 (such as point mutations) occupy a smaller portion of the rectangle, while larger lesions (such as copy-number alterations) occupy a larger portion.

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

Colors can be manually specified using a named vector supplied to `clr`. When custom colors are provided, they are matched to lesion types by name and retained in the resulting oncoprint settings.

Value

A list with the following components:

- `alter_func`: A list of alteration-drawing functions for rendering the different lesion types in the oncoprint.
- `col`: A named vector of colors assigned to each lesion type.
- `heatmap_legend_param`: Legend parameters for the oncoprint.

Author(s)

Lakshmi Patibandla <LakshmiAnuhya.Patibandla@stjude.org>, 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.

Examples

```

data(lesion_data)

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

# 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
  )
)

# Specify custom colors for lesion types:
custom.colors <- c(
  "gain" = "red",
  "loss" = "blue",
  "mutation" = "olivedrab",
  "fusion" = "black"
)

onco.props <- onco.print.props(
  lesion_data,
  clr = custom.colors
)

```

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

Description

Orders gene annotation data by chromosome and genomic coordinates and creates an index identifying the rows corresponding to each chromosome. This helper function is used to prepare gene annotation data for downstream GRIN overlap analyses.

Usage

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

Arguments

gene.data	A data.frame containing gene annotation information, either provided by the user or obtained using <code>get.ensembl.annotation</code> . The data frame must contain the following columns: "gene" Gene identifier, typically an unversioned Ensembl gene ID.
-----------	---

"**chrom**" Chromosome identifier on which the gene is located.

"**loc.start**" Gene start position in base pairs.

"**loc.end**" Gene end position in base pairs.

Details

Genes are ordered by chromosome, start position, and end position. Consecutive rows belonging to the same chromosome are represented by a single entry in `gene.index`, which records the first and last corresponding row in the ordered gene annotation data.

A `gene.row` column is added to the ordered gene annotation data and is used internally by downstream GRIN functions.

Value

A list with two components:

gene.data The input gene annotation data ordered by chromosome, `loc.start`, and `loc.end`, with an added `gene.row` column.

gene.index A data.frame with three columns: `chrom`, `row.start`, and `row.end`, identifying the range of rows corresponding to each chromosome in the ordered gene annotation 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

[prep.gene.lsn.data](#), [get.ensembl.annotation](#)

Examples

```
data(hg38_gene_annotation)

# Order gene annotation data and create chromosome-specific row indices
ordered.genes <- order.index.gene.data(hg38_gene_annotation)
```

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

Description

Orders lesion data by lesion type, chromosome, and patient identifier and creates an index identifying contiguous rows for each lesion type-chromosome-patient combination. This helper function prepares lesion data for downstream GRIN overlap analyses.

Usage

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

Arguments

lsn.data A data.frame containing lesion data in GRIN-compatible format with the following columns:

- "ID"** Patient identifier.
- "chrom"** Chromosome identifier on which the lesion is located.
- "loc.start"** Lesion start position in base pairs.
- "loc.end"** Lesion end position in base pairs.
- "lsn.type"** Lesion type assigned to the genomic alteration.

Details

Lesions are ordered by lesion type, chromosome, and patient identifier. Consecutive rows sharing the same lesion type, chromosome, and patient are represented by a single entry in `lsn.index`.

For example, if a patient has one lesion of a given type on a chromosome, `row.start` and `row.end` are identical. If multiple lesions of the same type occur on that chromosome for the same patient, the index spans the corresponding rows.

An `lsn.row` column is added to the ordered lesion data and is used internally by downstream GRIN functions.

Value

A list with two components:

lsn.data The input lesion data ordered by lesion type, chromosome, and patient identifier, with an added `lsn.row` column.

lsn.index A data.frame with five columns: `lsn.type`, `chrom`, `ID`, `row.start`, and `row.end`, identifying the range of rows corresponding to each lesion type-chromosome-patient combination.

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](#)

Examples

```
data(lesion_data)

# Order lesion data and create lesion type-chromosome-patient row indices
ordered.lsn <- order.index.lsn.data(lesion_data)
```

pathways

Genes Annotated to Biological Pathways

Description

A dataset listing genes annotated to 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 Unversioned Ensembl gene ID.

pathway Biological pathway to which the gene is annotated.

Source

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

```
prep.binary.lsn.mtx
```

Prepare Binary Lesion Matrix

Description

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

Usage

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

Arguments

<code>ov.data</code>	GRIN gene-lesion overlap results, typically the output from the <code>find.gene.lsn.overlaps</code> function.
<code>min.ngrp</code>	Optional integer specifying the minimum number of patients required in both the affected and unaffected groups for a gene-lesion type combination to be retained. The default is 0, which retains all observed gene-lesion type combinations.

Details

The function uses the gene-lesion overlaps in `ov.data$gene.lsn.hits` to construct a binary matrix with gene-lesion type combinations as rows and patients as columns.

Each row is labeled using the format `gene_lesion.type` (for example, `ENSG00000118513_gain`). For each gene-lesion type combination, a patient receives a value of 1 if affected by that lesion type in the corresponding gene and 0 otherwise.

When `min.ngrp > 0`, a row is retained only when both the affected group (1) and unaffected group (0) contain at least `min.ngrp` patients. This can be useful when the resulting binary matrix is used for downstream analyses that require a minimum number of patients in each comparison group.

Value

A numeric binary matrix in which:

- Rows represent gene-lesion type combinations (`gene_lesion.type`).
- Columns represent patient IDs.
- Entries are 1 when the patient is affected by the specified lesion type in that gene and 0 otherwise.

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

[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(prepare.gene.lsn)

# 3) Create a binary lesion matrix requiring at least 5 patients
# in both the affected and unaffected groups
lsn.binary.mtx <- prep.binary.lsn.mtx(gene.lsn.overlap,
                                     min.ngrp = 5)
```

prep.gene.lsn.data	<i>Prepare Gene and Lesion Data for GRIN Analysis</i>
--------------------	---

Description

Prepares and indexes gene and lesion data for downstream GRIN (Genomic Random Interval) analysis. The function merges and orders gene and lesion coordinates to support efficient computation of overlaps between genes and genomic lesions. When requested, it also prepares gene- and chromosome-level exon target sizes for lesion types restricted to exonic regions.

Usage

```
prep.gene.lsn.data(lsn.data,
                  gene.data,
                  exons.annotation = NULL,
                  exon.chrom.size = NULL,
                  exon_level = NULL,
                  mess.freq = 10)
```

Arguments

lsn.data	A data.frame containing lesion data in GRIN-compatible format. The following five columns are required: 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 genomic lesion, such as mutation, breakpoint, gain, amplification, heterozygous deletion, or homozygous deletion.
gene.data	A data.frame containing gene annotation data with the following four required columns: gene Ensembl gene identifier. 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.
exons.annotation	An optional data.frame containing exon annotation data. This argument is required when exon_level is specified. The exon annotation may contain the same gene set as gene.data or a subset of those genes. Genes in gene.data without a valid matching exon annotation remain in standard GRIN analyses, but exon-level probabilities are not computed for those genes. Compatible exon annotation data for supported genome assemblies can be retrieved using get.ensembl.annotation(). The following four columns are required: gene Ensembl gene identifier of the gene to which each annotated exon belongs. Multiple rows may therefore share the same gene identifier, with each row representing a different exon of that gene. chrom Chromosome on which the exon is located. loc.start Start position of the exon in base pairs. loc.end End position of the exon in base pairs.
exon.chrom.size	An optional data.frame containing the total annotated exon target size for each chromosome. This argument is required when exon_level is specified and must contain: chrom Chromosome identifier. size Total annotated exon target size for the chromosome in base pairs. The chromosome-level exon sizes should be derived from a genome-wide exon annotation using the same genome assembly and representative-transcript selection procedure used to construct exons.annotation. A compatible chromosome-level exon-size dataset is provided with GRIN2 and can be loaded using <code>data(hg38_exon_chrom_size)</code>. Corresponding annotation data can also be obtained using get.ensembl.annotation().
exon_level	An optional character vector specifying the lesion types that should use exon-level target sizes in downstream GRIN probability calculations. Lesions belonging to these types must be restricted to exonic regions before analysis. For example, when mutations are specified for exon-level analysis, exonic variants

such as missense, nonsense, frameshift, and synonymous mutations may be included, whereas intronic and other non-coding mutations should be excluded before running the analysis. The default is NULL.

`mess.freq` Integer specifying the frequency of progress messages. The default is 10.

Details

The function first orders and indexes the lesion data by lesion type, chromosome, and subject and orders the gene annotation by chromosome and genomic position. Gene and lesion boundaries are then combined into a unified genomic position table.

The `cty` column in the combined table identifies the boundary represented by each row:

- 1 Gene start.
- 2 Lesion start.
- 3 Lesion end.
- 4 Gene end.

The resulting table and index objects are used by `find.gene.lsn.overlaps()` to identify gene-lesion overlaps.

When `exon_level` is specified, exon lengths are calculated from `exons.annotation` and summed to determine the exon target size for each matched gene. Chromosome-level exon target sizes are not calculated by this function and must instead be supplied through `exon.chrom.size`. `exons.annotation` may contain all genes in `gene.data` or only a subset. Genes without matching exon annotations remain available for lesion types analyzed using standard genomic coordinates, but their gene-level exon target sizes are recorded as missing and exon-level probabilities are not computed for those genes.

Gene-lesion overlaps continue to be determined using the complete genomic coordinates of each gene, including when `exon_level` is specified. Exon annotation is used only to calculate exon-based target sizes for downstream probability calculations. Therefore, lesion types specified in `exon_level` must contain only exonic lesions; non-exonic lesions must be removed by the user before running the analysis.

Lesion types not included in `exon_level` continue to use the standard chromosome-level GRIN workflow.

Exact duplicate exon records are removed before gene-level exon sizes are calculated. Incomplete or malformed exon records are excluded with a warning.

Value

A list with the following components:

lsn.data Processed lesion data, including indices identifying the range of rows in `gene.lsn.data` corresponding to each lesion.

gene.data Processed gene annotation data, including indices identifying the range of rows in `gene.lsn.data` corresponding to each gene.

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

gene.index Index data.frame indicating the ordered start and end rows for each chromosome in the gene data.

lsn.index Index data.frame indicating the ordered start and end rows for lesion groups defined by lesion type, chromosome, and subject.

gene.exon.size Numeric vector containing the total annotated exon target size for each gene, aligned by gene.row. Genes without a valid matching exon annotation receive NA and are excluded from probability calculations for lesion types specified in exon_level. Returned as NULL when exon_level = NULL.

exon.chrom.size A data.frame containing chromosome identifiers in the chrom column and total genome-wide annotated exon target sizes in the size column. Returned as NULL when exon_level = NULL.

exon_level The character vector of lesion types designated for exon-level analysis. Returns NULL when exon-level preprocessing is not requested.

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)
data(example_exon_annotation)
data(hg38_exon_chrom_size)

# Prepare gene and lesion data using the optional arguments
# for exon-level analysis
prep.gene.lsn <- prep.gene.lsn.data(
  lsn.data = lesion_data,
  gene.data = hg38_gene_annotation,
  exons.annotation = example_exon_annotation,
  exon.chrom.size = hg38_exon_chrom_size,
  exon_level = "mutation"
)
```

```
prep.lsn.type.matrix
```

Prepare Lesion Type Matrix

Description

Constructs a lesion type matrix summarizing the genomic lesion type(s) 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	GRIN gene-lesion overlap results, typically the output from the <code>find.gene.lsn.overlaps</code> function.
min.ngrp	Optional integer specifying the minimum number of patients affected by any lesion type in a given gene for that gene to be retained in the output matrix. The default is 0, which retains all genes represented in the gene-lesion overlap data.

Details

The function uses the gene-lesion overlaps in `ov.data$gene.lsn.hits` to construct a character matrix with genes as rows and patients as columns.

For each gene-patient combination:

- "none" indicates that no lesion overlaps the gene in that patient.
- A specific lesion type indicates that only one distinct lesion type affects the gene in that patient.
- "multiple" indicates that more than one distinct lesion type affects the same gene in that patient.

Multiple lesions of the same type affecting the same gene in a patient are represented by that lesion type rather than "multiple".

When `min.ngrp > 0`, genes affected by any lesion type in fewer than `min.ngrp` patients are excluded from the final matrix.

Value

A character matrix in which:

- Rows represent genes.
- Columns represent patient IDs.
- Entries are "none", a specific lesion type, or "multiple".

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

[prep.gene.lsn.data](#), [find.gene.lsn.overlaps](#), [prep.binary.lsn.mtx](#)

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(prepare.gene.lsn)

# 3) Create lesion type matrix for genes affected in at least 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, such as a gene or regulatory region, is affected by one or more types of genomic lesions. Statistical significance for lesion enrichment is estimated using a convolution of independent but non-identically distributed Bernoulli probabilities.

Usage

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

Arguments

hit.cnt	A list returned by <code>count.hits()</code> containing the number of subjects and lesions affecting each locus by lesion type, together with the processed lesion and gene annotations. When exon-level analysis was requested in <code>prep.gene.lsn.data()</code> , this object also contains the gene- and chromosome-level exon target sizes and the lesion types designated for exon-level analysis.
---------	--

chr.size A `data.frame` containing chromosome sizes for the 22 autosomes and the X and Y chromosomes. It must contain the following columns:

- chrom** Chromosome identifier.
- size** Chromosome length in base pairs.

Details

For each lesion type and chromosome, the function calculates the probability that each lesion overlaps each affected locus according to the relative sizes of the lesion, locus, and chromosome. These probabilities are combined across lesions and subjects using the Poisson-binomial distribution to estimate the probability of observing the recorded number of lesions or affected subjects by chance.

For lesion types not designated for exon-level analysis, locus size is calculated using the complete genomic boundaries in `gene.data`, and chromosome length is obtained from `chr.size`.

For lesion types designated for exon-level analysis, the function instead uses the total annotated exon size of each gene and the total annotated exon target size of the corresponding chromosome. Lesion types specified in `exon_level` must therefore contain only exonic lesions. For example, when mutations are analyzed at the exon level, exonic variants such as missense, nonsense, frameshift, and synonymous mutations may be included, whereas intronic and other non-coding mutations should be excluded before running the analysis.

The exon-level specification and target sizes are carried through the GRIN workflow within `hit.cnt`; users therefore specify the exon-level lesion types only once when calling `prep.gene.lsn.data()`.

Genes without a valid matching exon annotation remain in analyses of lesion types that use standard genomic coordinates. For lesion types designated for exon-level analysis, their lesion-specific p-values and q-values are returned as NA. Ordered constellation statistics are also returned as NA for genes that do not have a complete set of lesion-specific probabilities.

The function calculates lesion-specific p-values for:

Number of affected subjects The probability of observing the recorded number of unique subjects with at least one lesion affecting the locus.

Number of lesion hits The probability of observing the recorded total number of lesions affecting the locus.

False discovery rate-adjusted q-values are calculated using the method of Pounds and Cheng. When more than one lesion type is present, the function also calculates ordered p- and q-values representing constellations of one or more lesion types affecting the same locus.

Value

A list containing the following components:

gene.hits A `data.frame` containing locus annotations; the numbers of lesion hits and affected subjects by lesion type; lesion-specific p-values and q-values for hit counts (`p.nhit.*` and `q.nhit.*`) and affected-subject counts (`p.nsubj.*` and `q.nsubj.*`); and, when multiple lesion types are present, ordered constellation p-values and q-values (`p1.nhit`, `p2.nhit`, ..., `q1.nhit`, `q2.nhit`, ... and the corresponding `nsubj` columns).

lsn.data Processed input lesion data.

gene.data Processed input gene annotation data.

gene.lsn.data A data.frame in which each row represents a locus overlapped by a genomic lesion.

chr.size Chromosome size information used for standard chromosome-level calculations.

gene.index A data.frame indexing loci by chromosome.

lsn.index A data.frame indexing lesion groups by lesion type, chromosome, and subject.

gene.exon.size A numeric vector containing the exon target size for each gene. Genes without a valid matching exon annotation have a value of NA; exon-level probability calculations are not performed for those genes. Returns NULL when exon-level analysis was not requested.

exon.chrom.size A data.frame containing the genome-wide annotated exon target size for each chromosome. These values provide the chromosome-level exonic search space used for lesion types designated for exon-level analysis. Returns NULL when exon-level analysis was not requested.

exon_level A character vector specifying the lesion types evaluated using exon-level target sizes. Returns NULL when exon-level analysis was not requested.

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.

Pounds, S., and Cheng, C. (2006). Robust estimation of the false discovery rate.

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)
data(example_exon_annotation)
data(hg38_exon_chrom_size)

# 1) Prepare gene and lesion data using the optional arguments
# for exon-level analysis
prep.gene.lsn <- prep.gene.lsn.data(
  lsn.data = lesion_data,
  gene.data = hg38_gene_annotation,
  exons.annotation = example_exon_annotation,
  exon.chrom.size = hg38_exon_chrom_size,
  exon_level = "mutation"
)
```

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

# 3) Count subjects and lesions affecting each gene
count.nsubj.nhits <- count.hits(gene.lsn.overlap)

# 4) Compute p- and q-values for lesion enrichment
hits.prob <- prob.hits(count.nsubj.nhits, hg38_chrom_size)
```

```
top.alex.waterfall.plots
```

Generate Waterfall Plots for Significant Genes

Description

Generates waterfall plots for genes showing significant associations between lesion status and gene expression based on Kruskal-Wallis (KW) test results. Genes with KW q-values below a user-specified threshold are selected for plotting.

Usage

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

Arguments

out.dir	Character string specifying the directory where the waterfall plot PDF files will be saved. The directory must already exist.
alex.data	A list returned by alex.prep.lsn.expr containing the matched expression and lesion data used for lesion-expression association analyses.
alex.kw.results	Kruskal-Wallis lesion-expression association results returned by KW.hit.express.
q	Numeric; KW q-value threshold used to select genes for plotting. Genes with $q.KW < q$ are included.
lsn.data	Lesion data in GRIN-compatible format, as used in alex.prep.lsn.expr.

Details

The function selects genes from alex.kw.results with non-missing q.KW values below the specified q threshold and with available gene symbols. Duplicate gene symbols are represented once.

For each selected gene, alex.waterfall.prep prepares the corresponding lesion and expression data and alex.waterfall.plot generates a waterfall plot showing gene expression across patients according to lesion status.

Each gene is saved as a separate PDF file named <gene>_waterfall_plot.pdf in out.dir.

Value

Invisibly returns NULL. The function creates one PDF waterfall plot for each selected gene in the specified output directory. If no genes meet the specified q-value threshold, no PDF files are created.

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

[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 lesion-expression association analysis
alex.kw.results <- KW.hit.express(alex.data,
                                 hg38_gene_annotation,
                                 min.grp.size = 5)

# 3) Create a temporary output directory and generate waterfall plots
resultsFolder <- file.path(tempdir(), "temp.out")
dir.create(resultsFolder, showWarnings = FALSE, recursive = TRUE)

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 an Excel File***Description**

Writes selected components of GRIN analysis results to a multi-sheet Excel workbook. The workbook contains GRIN statistical results, input lesion and annotation data, chromosome sizes, an interpretation sheet describing the output columns, and a summary of the statistical methods and references.

The `gene.lsn.data` component is intentionally excluded from the Excel workbook because gene-lesion overlap tables can become too large for an Excel worksheet.

Usage

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

Arguments

<code>grin.result</code>	A list returned by <code>grin.stats</code> containing GRIN analysis results. The object must include <code>gene.hits</code> , <code>lsn.data</code> , <code>gene.data</code> , and <code>chr.size</code> .
<code>output.file</code>	A character string specifying the path and filename of the output Excel workbook. The filename must end with <code>".xlsx"</code> .

Details

The function exports the principal GRIN results and input data while excluding `gene.lsn.data`, which may exceed the size supported by an Excel worksheet for large analyses.

An interpretation worksheet is generated automatically to describe the exported data sheets and the columns in `gene.hits`. Descriptions are generated dynamically according to the lesion types and constellation statistics present in the GRIN results.

A `methods.paragraph` worksheet is also included with a concise description of the GRIN statistical methodology and relevant references.

Value

Invisibly returns `NULL`. The function writes an Excel workbook to `output.file` containing the following worksheets:

- `gene.hits`: GRIN statistical results, including lesion counts, subject counts, individual lesion-type p- and q-values, and available constellation statistics.
- `lsn.data`: Input genomic lesion data used in the analysis.
- `gene.data`: Input gene annotation data used in the analysis.
- `chr.size`: Input chromosome size data.
- `interpretation`: Descriptions of the exported worksheets and result columns.
- `methods.paragraph`: Summary of the GRIN statistical methods and references.

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. *Bioinformatics*, 29(17), 2088-2095.

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
)

# Write GRIN results to a temporary Excel workbook
tmp_file <- tempfile(fileext = ".xlsx")

write.grin.xlsx(
  grin.results,
  output.file = tmp_file
)

unlink(tmp_file)
```

Index

* datasets

- clin_data, 13
- example_exon_annotation, 19
- expr_data, 20
- grin.results, 44
- hg38_chrom_size, 51
- hg38_cytoband, 51
- hg38_exon_chrom_size, 52
- hg38_gene_annotation, 53
- lesion_data, 56
- pathways, 66

alex.boxplots, 3

alex.pathway, 5

alex.prep.lsn.expr, 3–6, 7, 10–12, 54, 55, 77

alex.waterfall.plot, 9, 11, 12, 77

alex.waterfall.prep, 9, 10, 11, 77

clin_data, 13

compute.gw.coordinates, 14, 23, 25

count.hits, 15, 22, 48, 75

coxph, 31, 34, 35

default.grin.colors, 5, 9, 10, 18, 23, 25, 37, 50, 62

ensembl.db::getGeneRegionTrackForGviz(), 58, 59

example_exon_annotation, 19

expr_data, 20

fct_reorder, 4

find.gene.lsn.overlaps, 8, 17, 20, 48, 68, 71, 73, 75

genomewide.log10q.plot, 22, 41

genomewide.lsn.plot, 15, 23, 24

get.chrom.length, 26

get.ensembl.annotation, 7, 8, 27, 46, 48, 54, 64

get.ensembl.annotation(), 69

glm, 31, 34, 35

grin.assoc.expr.outcome, 30

grin.assoc.lsn.outcome, 33

grin.barplt, 36

grin.logRank, 38

grin.lsn.boundaries, 23, 40

grin.oncoprint.mtx, 42, 45

grin.results, 44

grin.stats, 15, 23, 25, 29, 37, 41–45, 45, 50, 78, 79

grin.stats(), 44, 59

grin.stats.lsn.plot, 49

Gviz::GeneRegionTrack(), 59

Gviz::GenomeAxisTrack(), 59

Gviz::IdeogramTrack(), 59

hclust, 6

hg38_chrom_size, 51

hg38_cytoband, 51

hg38_exon_chrom_size, 52

hg38_gene_annotation, 53

KW.hit.express, 3, 4, 7, 8, 10–12, 54, 77

lesion_data, 56

lsn.transcripts.plot, 45, 50, 52, 57

onco.print.props, 61

oncoPrint, 42, 43

order.index.gene.data, 63, 71

order.index.lsn.data, 65, 71

p.adjust, 55

pathways, 66

prep.binary.lsn.mtx, 33, 35, 67, 73

prep.gene.lsn.data, 8, 17, 22, 48, 64, 66, 68, 68, 73, 75

prep.lsn.type.matrix, 8, 38, 39, 72

prob.hits, 17, 22, 48, 73

`read.chromInfo`, [27](#)

`Surv`, [31](#), [34](#), [39](#)

`survdiff`, [39](#)

`top.alex.waterfall.plots`, [76](#)

`write.grin.xlsx`, [45](#), [78](#)