Package ‘scLANE’

February 21, 2026

Type Package

Title Model Gene Expression Dynamics with Spline-Based NB GLMs, GEEs, & GLMMs

Version 1.0.3

Description Our scLANE model uses truncated power basis spline models to build flexible, interpretable models of single cell gene expression over pseudotime or latent time.
The modeling architectures currently supported are Negative-binomial GLMs, GEEs, & GLMMs.
Downstream analysis functionalities include model comparison, dynamic gene clustering, smoothed counts generation, gene set enrichment testing, & visualization.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends glm2, magrittr, R (>= 4.5.0)

Imports geeM, MASS, mpath, dplyr, stats, utils, withr, purrr, tidyr, furrr, doSNOW, gamlss, scales, future, Matrix, ggplot2, splines, foreach, glmmTMB, parallel, RcppEigen, bigstatsr, tidyselect, broom.mixed, Rcpp

URL `https://github.com/jr-leary7/scLANE`

BugReports `https://github.com/jr-leary7/scLANE/issues`

Suggests covr, grid, coop, uwot, scran, ggh4x, knitr, UCell, irlba, rlang, magick, igraph, scater, gtable, ggpubr, viridis, bluster, cluster, circlize, speedglm, rmarkdown, gridExtra, BiocStyle, slingshot, gprofiler2, GenomeInfoDb, BiocParallel, BiocGenerics, BiocNeighbors, ComplexHeatmap, Seurat (>= 5.0.0), testthat (>= 3.0.0), SingleCellExperiment, SummarizedExperiment

VignetteBuilder knitr

Config/testthat/edition 3

LinkingTo Rcpp, RcppEigen

biocViews RNASeq, Software, Clustering, TimeCourse, Sequencing, Regression, SingleCell, Visualization, GeneExpression, Transcriptomics, GeneSetEnrichment, DifferentialExpression

Additional_repositories `https://bioconductor.org/packages/release/bioc`

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

git_branch RELEASE_3_22

git_last_commit 70d4f40

git_last_commit_date 2026-02-12

Repository Bioconductor 3.22

Date/Publication 2026-02-20

Author Jack R. Leary [aut, cre] (ORCID: <https://orcid.org/0009-0004-8821-3269>), Rhonda Bacher [ctb, fnd] (ORCID: <https://orcid.org/0000-0001-5787-476X>)

Maintainer Jack R. Leary <j.leary@ufl.edu>

Contents

backward_sel_WIC — *Backward selection function for MARGE - uses the Wald information criterion (WIC).*

Description

Backward selection function for MARGE - uses the Wald information criterion (WIC).

Usage

```
backward_sel_WIC(
  Y = NULL,
  B_new = NULL,
  is.gee = FALSE,
  id.vec = NULL,
  cor.structure = NULL,
  theta.hat = NULL,
  sandwich.var = FALSE
)
```

Arguments

`Y`	The response variable. Defaults to NULL.
`B_new`	The model matrix. Defaults to NULL.
`is.gee`	Is the model a GEE? Defaults to FALSE.
`id.vec`	A vector of observation IDs that is necessary for fitting a GEE model. Defaults to NULL.
`cor.structure`	The specified working correlation structure of the GEE model. Must be one of "independence", "ar1", or "exchangeable". Defaults to NULL.
`theta.hat`	An initial estimate of $\hat{\theta}$ used to fit the negative-binomial model when GEE mode is being used.
`sandwich.var`	Should the sandwich variance estimator be used instead of the model-based estimator? Default to FALSE.

Value

backward_sel_WIC returns the Wald statistic from the fitted model (the penalty is applied later on).

Author(s)

Jakub Stoklosa

David I. Warton

Jack R. Leary

References

Stoklosa, J. Gibb, H. Warton, D.I. Fast forward selection for Generalized Estimating Equations With a Large Number of Predictor Variables. *Biometrics*, **70**, 110–120.

Stoklosa, J. and Warton, D.I. (2018). A generalized estimating equation approach to multivariate adaptive regression splines. *Journal of Computational and Graphical Statistics*, **27**, 245–253.

---

biasCorrectGEE *Bias-correct the GEE sandwich variance-covariance matrix.*

Description

This functions implements several bias-correction methods for the GEE sandwich variance-covariance matrix; they are to be used when the number of subjects is small or the numer of timepoints per-subject is very large.

Usage

```
biasCorrectGEE(
  fitted.model = NULL,
  correction.method = "kc",
  id.vec = NULL,
  cor.structure = "ar1",
  verbose = FALSE
)
```

Arguments

fitted.model	The fitted model of class geem returned by marge2. Defaults to NULL.
correction.method	A string specifying the correction method to be used. Currently supported options are "df" and "kc". Defaults to "kc".
id.vec	A vector of subject IDs. Defaults to NULL.
cor.structure	A string specifying the correlation structure used in fitting the model. Defaults to "ar1".
verbose	(Optional) A Boolean specifying whether or not verbose output should be printed to the console. Occasionally useful for debugging. Defaults to FALSE.

Value

An object of class matrix containing the bias-corrected variance-covariance estimates.

Author(s)

Jack R. Leary

See Also

`waldTestGEE`

`bootstrapRandomEffects` *Generate bootstrapped confidence intervals for random effects.*

Description

This function leverages the parametric bootstrap to generate empirical confidence intervals for the random effects terms of a fitted model.

Usage

```
bootstrapRandomEffects(
  glmm.mod = NULL,
  id.vec = NULL,
  Y.offset = NULL,
  n.boot = 500L,
  alpha = 0.05,
  n.cores = 4L,
  random.seed = 312,
  verbose = TRUE
)
```

Arguments

`glmm.mod`	The output from `fitGLMM`. Defaults to NULL.
`id.vec`	A vector of subject IDs. Defaults to NULL.
`Y.offset`	An offset to be included in the final model fit. Defaults to NULL.
`n.boot`	(Optional) The number of bootstrap resamples to generate. Defaults to 500.
`alpha`	(Optional) The desired confidence level. Defaults to good old 0.05.
`n.cores`	(Optional) The number of threads to use in parallel processing of the bootstrap resampling procedure. Defaults to 4.
`random.seed`	(Optional) The seed used to control stochasticity during bootstrap resampling. Defaults to 312.
`verbose`	(Optional) A boolean indicating whether a progress bar should be printed to the console. Defaults to TRUE.

Value

An object of class `data.frame` containing the upper and lower quantiles of the per-subject random effects.

Author(s)

Jack R. Leary

See Also

`fitGLMM`

`glmmTMB`

Examples

```
data(sim_counts)
data(sim_pseudotime)
cell_offset <- createCellOffset(sim_counts)
glmm_mod <- fitGLMM(
    X_pred = sim_pseudotime,
    Y = BiocGenerics::counts(sim_counts)[4, ],
    Y.offset = cell_offset,
    id.vec = sim_counts$subject,
    return.basis = TRUE
)
ranef_sumy <- bootstrapRandomEffects(glmm_mod,
    id.vec = sim_counts$subject,
    Y.offset = cell_offset,
    n.boot = 10L,  # in practice use a larger number such as 500
    n.cores = 1L
)
```

`chooseCandidateGenes` *Choose candidate genes for trajectory DE analysis.*

Description

This function identifies good gene candidates for trajectory differential expression modeling by ranking genes based on their mean expression, SD of expression, and sparsity across cells.

Usage

```
chooseCandidateGenes(
  obj = NULL,
  group.by.subject = TRUE,
  id.vec = NULL,
  n.desired.genes = 2000L
)
```

Arguments

`obj`	An object of class `SingleCellExperiment`, `Seurat`, or `cell_data_set` (from `monocle3`), or a gene-by-cell matrix (sparse or dense). Defaults to NULL.
`group.by.subject`	Boolean specifying whether or not the summary statistics should be computed per-subject and then mean-aggregated. Defaults to TRUE.
`id.vec`	A vector of subject IDs. Defaults to NULL.

`n.desired.genes`	An integer specifying the number of candidate genes to return. Defaults to 2000.

Value

A vector of candidate gene names.

Author(s)

Jack R. Leary

Examples

```
data(sim_counts)
candidate_genes <- chooseCandidateGenes(sim_counts,
                                        id.vec = sim_counts$subject)
```

`clusterGenes` *Cluster the fitted values from a set of* `scLANE` *models.*

Description

This function takes as input the output from `testDynamic` and clusters the fitted values from the model for each gene using one of several user-chosen algorithms. An approximately optimal clustering is determined by iterating over reasonable hyperparameter values & choosing the value with the highest mean silhouette score based on the cosine distance.

Usage

```
clusterGenes(
  test.dyn.res = NULL,
  pt = NULL,
  size.factor.offset = NULL,
  clust.algo = "leiden",
  use.pca = FALSE,
  n.PC = 15L,
  lineages = NULL
)
```

Arguments

`test.dyn.res`	The list returned by `testDynamic` - no extra processing required. Defaults to NULL.
`pt`	A data.frame containing the pseudotime or latent time estimates for each cell. Defaults to NULL.
`size.factor.offset`	(Optional) An offset to be used to rescale the fitted values. Can be generated easily with `createCellOffset`. No need to provide if the GEE backend was used. Defaults to NULL.
`clust.algo`	The clustering method to use. Can be one of "hclust", "kmeans", "leiden". Defaults to "leiden".

`use.pca`	Should PCA be performed prior to clustering? Defaults to FALSE.
`n.PC`	The number of principal components to use when performing dimension reduction prior to clustering. Defaults to 15.
`lineages`	Should one or more lineages be isolated? If so, specify which one(s). Otherwise, all lineages will be clustered independently. Defaults to NULL.

Details

- Due to some peculiarities of how the fitted values (on the link scale) are generated for `geeM` models, it's not necessary to multiply them by the offset as this is done internally. For GLM & GEE models, the opposite is true, and `size.factor.offset` must be provided in order to rescale the fitted values correctly.

Value

A data.frame of with three columns: `Gene`, `Lineage`, and `Cluster`.

Author(s)

Jack Leary

See Also

`testDynamic`

`embedGenes`

`plotClusteredGenes`

Examples

```
data(sim_counts)
data(scLANE_models)
data(sim_pseudotime)
cell_offset <- createCellOffset(sim_counts)
gene_clusters <- clusterGenes(scLANE_models,
    pt = sim_pseudotime,
    size.factor.offset = cell_offset
)
```

`createCellOffset` *Create an offset vector before modeling.*

Description

Creates a vector of per-cell size factors to be used as input to `testDynamic` as a model offset given a variety of inputs.

Usage

```
createCellOffset(expr.mat = NULL, scale.factor = 10000)
```

Arguments

`expr.mat`	Either a (sparse or dense) matrix of raw integer counts (cells as columns), a `Seurat` object, or a `SingleCellExperiment` object. Defaults to NULL.
`scale.factor`	The scaling factor use to multiply the sequencing depth factor for each cell. The default value is 1e4, which returns counts-per-10k.

Value

A named numeric vector containing the computed size factor for each cell.

Author(s)

Jack R. Leary

See Also

`testDynamic`

`marge2`

`LogNormalize`

`computeLibraryFactors`

Examples

```
data(sim_counts)
cell_offset <- createCellOffset(sim_counts)
```

`createSlopeTestData` *A helper function to create a dataframe of breakpoints and associated* p*-values from a* `marge` *model.*

Description

Creates a data.frame of `marge` model breakpoints, *p*-values, and other info.

Usage

```
createSlopeTestData(
  marge.model = NULL,
  pt = NULL,
  is.gee = FALSE,
  is.glmm = FALSE
)
```

Arguments

`marge.model`	A `marge` model object, like those returned from `marge2`. Defaults to NULL.
`pt`	A data.frame containing pseudotime or latent time values. Defaults to NULL.
`is.gee`	Was the GEE mode used? Defaults to FALSE.
`is.glmm`	Was the GLMM mode used? Defaults to FALSE.

Value

A data.frame containing model data.

Author(s)

Jack R. Leary

See Also

marge2

testSlope

---

embedGenes *Generate PCA & UMAP embeddings of fitted gene dynamics.*

Description

Embed genes in dimension-reduced space given a smoothed counts matrix.

Usage

```
embedGenes(
  smoothed.counts = NULL,
  genes = NULL,
  pca.init = FALSE,
  pc.embed = 30,
  pc.return = 2,
  cluster.genes = TRUE,
  gene.meta.data = NULL,
  k.param = 20,
  resolution.param = NULL,
  random.seed = 312,
  n.cores = 2L
)
```

Arguments

`smoothed.counts`	The output from smoothedCountsMatrix. Defaults to NULL.
`genes`	A character vector of genes to embed. If not specified, all genes in `smoothed.counts` are used. Defaults to NULL.
`pca.init`	A boolean specifying whether or not the embedded PCs should be used as initialization for clustering and UMAP. The default is to cluster/embed the raw dynamics i.e., defaults to FALSE.
`pc.embed`	(Optional) How many PCs should be used to cluster the genes and run UMAP? Defaults to 30.
`pc.return`	(Optional) How many principal components should be included in the output? Defaults to 2.
`cluster.genes`	(Optional) Should genes be clustered in PCA space using the Leiden algorithm? Defaults to TRUE.

`gene.meta.data`	(Optional) A data.frame of metadata values for each gene (HVG status, Ensembl ID, gene biotype, etc.) that will be included in the result table. Defaults to NULL.
`k.param`	(Optional) The value of nearest-neighbors used in creating the SNN graph prior to clustering & in running UMAP. Defaults to 20.
`resolution.param`	(Optional) The value of the resolution parameter for the Leiden algorithm. If unspecified, silhouette scoring is used to select an optimal value. Defaults to NULL.
`random.seed`	(Optional) The random seed used to control stochasticity in the clustering algorithm. Defaults to 312.
`n.cores`	(Optional) Integer specifying the number of threads used by `umap` and in `makeSNNGraph`. Defaults to 2.

Value

A data.frame containing embedding coordinates, cluster IDs, and metadata for each gene.

Author(s)

Jack R. Leary

Examples

```
data(sim_pseudotime)
data(scLANE_models)
smoothed_dynamics <- smoothedCountsMatrix(scLANE_models,
    pt = sim_pseudotime,
    n.cores = 1L
)
gene_embed <- embedGenes(smoothed_dynamics$Lineage_A, n.cores = 1L)
```

`enrichDynamicGenes` *Perform GSEA on dynamic genes identified by* `scLANE`.

Description

This function uses the `gprofiler2` package to perform pathway analysis on a set of genes from one or more lineages that were determined to be dynamic with `testDynamic`.

Usage

```
enrichDynamicGenes(scLANE.de.res = NULL, lineage = NULL, species = "hsapiens")
```

Arguments

`scLANE.de.res`	The output from `getResultsDE`. Defaults to NULL.
`lineage`	A character vector specifying lineages to isolate. Defaults to NULL.
`species`	The species against which to run enrichment analysis. Defaults to "hsapiens".

Value

The output from `gost`.

Author(s)

Jack R. Leary

See Also

`gost`

Examples

```
data(scLANE_models)
scLANE_de_res <- getResultsDE(scLANE_models)
enr_res <- enrichDynamicGenes(scLANE_de_res)
```

`extractBreakpoints` *Identify breakpoints in a* `marge` *model.*

Description

Extracts the breakpoints from a fitted `marge` model. Note - this function relies on the name of the pseudotime variable not having any numeric characters in it e.g., "pseudotime" or "PT" would be fine but "pseudotime1" would not. If multiple lineages exist, use letters to denote lineages instead of numbers e.g., "Lineage_A" and "Lineage_B". This is currently handled automatically in `testDynamic`, so don't change anything.

Usage

```
extractBreakpoints(model = NULL, directions = TRUE)
```

Arguments

`model`	The `marge` model to analyze. Defaults to NULL.
`directions`	Should the directions of the hinge functions also be extracted? Defaults to TRUE.

Value

A data.frame of breakpoints & their directions.

Author(s)

Jack R. Leary

Examples

```
data(sim_counts)
data(sim_pseudotime)
cell_offset <- createCellOffset(sim_counts)
marge_model <- marge2(sim_pseudotime,
    Y = BiocGenerics::counts(sim_counts)[4, ],
    Y.offset = cell_offset
)
breakpoint_df <- extractBreakpoints(model = marge_model)
```

`fitGLMM` *Build an NB GLMM using truncated power basis functions.*

Description

Fits a negative-binomial generalized linear mixed model using truncated power basis function splines as input. The basis matrix can be created adaptively using subject-specific estimation of optimal knots using `marge2`, or basis functions can be evenly spaced across quantiles. The resulting model can output subject-specific and population-level fitted values.

Usage

```
fitGLMM(
  X_pred = NULL,
  Y = NULL,
  Y.offset = NULL,
  id.vec = NULL,
  adaptive = TRUE,
  approx.knot = TRUE,
  M.glm = 3,
  reg.penalty = "snet",
  return.basis = FALSE,
  return.GCV = FALSE,
  verbose = FALSE
)
```

Arguments

`X_pred`	A matrix with one column containing cell ordering values. Defaults to NULL.
`Y`	A vector of raw single cell counts. Defaults to NULL.
`Y.offset`	(Optional) An offset to be included in the final model fit. Defaults to NULL.
`id.vec`	A vector of subject IDs. Defaults to NULL.
`adaptive`	Should basis functions be chosen adaptively? Defaults to TRUE.
`approx.knot`	Should knot approximation be used in the calls to `marge2`? This speeds up computation somewhat. Defaults to TRUE.
`M.glm`	The number of possible basis functions to use in the calls to `marge2` when choosing basis functions adaptively. Defaults to 3.
`reg.penalty`	(Optional) String specifying the penalty type to be used when fitting a regularized negative-binomial model to select optimal basis functions. Defaults to "snet".

`return.basis`	(Optional) Whether the basis model matrix (denoted `basis_mtx`) should be returned as part of the `marge` model object. Defaults to FALSE.
`return.GCV`	(Optional) Whether the final GCV value should be returned as part of the `marge` model object. Defaults to FALSE.
`verbose`	(Optional) Should intermediate output be printed to the console? Defaults to FALSE.

Value

An object of class `marge` containing the fitted model & other optional quantities of interest (basis function matrix, GCV, etc.).

Author(s)

Jack R. Leary

See Also

`glmregNB`

`glmmTMB`

`testDynamic`

`modelLRT`

Examples

```
data(sim_counts)
data(sim_pseudotime)
cell_offset <- createCellOffset(sim_counts)
glmm_mod <- fitGLMM(
    X_pred = sim_pseudotime,
    Y = BiocGenerics::counts(sim_counts)[4, ],
    Y.offset = cell_offset,
    id.vec = sim_counts$subject
)
```

`geneProgramDrivers` *Identify driver genes for a given gene program.*

Description

This function computes the correlation between smoothed gene expression and gene program scores in order to identify genes are significantly associated with program scores i.e., the "drivers" of the gene program.

Usage

```
geneProgramDrivers(
  expr.mat = NULL,
  genes = NULL,
  gene.program = NULL,
  cor.method = "spearman",
  fdr.cutoff = 0.01,
  p.adj.method = "holm",
  n.cores = 2L,
  verbose = TRUE
)
```

Arguments

`expr.mat`	Either a `SingleCellExperiment` or `Seurat` object from which counts can be extracted, or a matrix of normalized counts with genes as rows & cells as columns. Defaults to NULL.
`genes`	A character vector of genes to test. Defaults to NULL.
`gene.program`	A vector of program scores as returned by `geneProgramScoring`. Defaults to NULL.
`cor.method`	(Optional) The correlation method to be used. Defaults to "spearman".
`fdr.cutoff`	(Optional) The FDR threshold for determining statistical significance. Defaults to 0.01.
`p.adj.method`	(Optional) The method used to adjust *p*-values for multiple hypothesis testing. Defaults to "holm".
`n.cores`	(Optional) The number of cores used when iterating over genes to perform testing. Defaults to 2.
`verbose`	(Optional) Should a progress bar be printed to the console during processing? Defaults to TRUE.

Value

Either a `Seurat` or `SingleCellExperiment` object if `expr.mat` is in either form, or a data.frame containing per-cell program scores if `expr.mat` is a matrix.

Author(s)

Jack R. Leary

See Also

`geneProgramScoring`

`cor.test`

Examples

```
data(sim_counts)
data(scLANE_models)
data(sim_pseudotime)
smoothed_dynamics <- smoothedCountsMatrix(scLANE_models,
    pt = sim_pseudotime,
    n.cores = 1L
)
gene_embed <- embedGenes(smoothed_dynamics$Lineage_A, n.cores = 1L)
sim_counts <- geneProgramScoring(sim_counts,
    genes = gene_embed$gene,
    gene.clusters = gene_embed$leiden,
    n.cores = 1L
)
program_drivers <- geneProgramDrivers(sim_counts,
    genes = gene_embed$gene,
    gene.program = sim_counts$cluster_0,
    fdr.cutoff = 0.05,
    n.cores = 1L
)
```

geneProgramScoring *Add per-cell module scores for gene programs.*

Description

This function uses ScoreSignatures_UCell to create a per-cell module score for each of the provided gene clusters. If the input matrix is a Seurat or SingleCellExperiment object, then the resulting scores will be added to the meta.data or the colData slot, respectively. Otherwise, a data.frame of the per-program scores is returned.

Usage

```
geneProgramScoring(
  expr.mat = NULL,
  genes = NULL,
  gene.clusters = NULL,
  program.labels = NULL,
  minmax.norm = TRUE,
  minmax.epsilon = 0.01,
  n.cores = 2L
)
```

Arguments

expr.mat	Either a SingleCellExperiment or Seurat object from which counts can be extracted, or a matrix of integer-valued counts with genes as rows & cells as columns. Defaults to NULL.
genes	A character vector of gene IDs. Defaults to NULL.
gene.clusters	A factor containing the cluster assignment of each gene in genes. Defaults to NULL.
program.labels	(Optional) A character vector specifying a label for each gene cluster. Defaults to NULL.
minmax.norm	(Optional) Should each program's score be min-max normalized to be on (0, 1)? Defaults to TRUE.
minmax.epsilon	(Optional) The tolerance used to ensure that program scores equal to 0 or 1 do not occur. Defaults to 0.01.
n.cores	(Optional) The number of cores used under the hood in ScoreSignatures_UCell. Defaults to 2.

Value

Either a Seurat or SingleCellExperiment object if expr.mat is in either form, or a data.frame containing per-cell program scores if expr.mat is a matrix.

Author(s)

Jack R. Leary

See Also

ScoreSignatures_UCell

geneProgramDrivers

Examples

```
data(sim_counts)
data(scLANE_models)
data(sim_pseudotime)
smoothed_dynamics <- smoothedCountsMatrix(scLANE_models,
    pt = sim_pseudotime,
    n.cores = 1L
)
gene_embed <- embedGenes(smoothed_dynamics$Lineage_A, n.cores = 1L)
sim_counts <- geneProgramScoring(sim_counts,
    genes = gene_embed$gene,
    gene.clusters = gene_embed$leiden,
    n.cores = 1L
)
```

---

geneProgramSignificance

Test significance of gene program enrichment across a trajectory.

Description

This function fits a Beta GAM with pseudotime as a covariate and gene program score as the response. The fitted model is then compared to a null, intercept-only model in order to determine the significance of the relationship between gene program and pseudotime.

Usage

```
geneProgramSignificance(
  gene.programs = NULL,
  pt = NULL,
  program.labels = NULL,
  p.adj.method = "holm"
)
```

Arguments

`gene.programs`	A list of vectors of program scores as returned by `geneProgramScoring`. Defaults to NULL.
`pt`	A vector of pseudotime values for each cell. May contain NAs, which are handled internally. Defaults to NULL.
`program.labels`	(Optional) A character vector specifying a label for each gene cluster. Defaults to NULL.
`p.adj.method`	(Optional) The method used to adjust *p*-values for multiple hypothesis testing. Defaults to "holm".

Details

- This function assumes that the gene program scores have been min-max normalized to (0, 1) i.e., *not* including the values 0 or 1. This is necessary to fit the Beta distribution additive model. This normalization can be easily generated by setting the argument `minmax.norm = TRUE` in the `geneProgramScoring` function.

Value

A table of statistical output showing the significance of the association between pseudotime and program scores.

Author(s)

Jack R. Leary

See Also

`geneProgramScoring`

Examples

```
data(sim_counts)
data(scLANE_models)
data(sim_pseudotime)
smoothed_dynamics <- smoothedCountsMatrix(scLANE_models,
    pt = sim_pseudotime,
    n.cores = 1L
)
gene_embed <- embedGenes(smoothed_dynamics$Lineage_A, n.cores = 1L)
sim_counts <- geneProgramScoring(sim_counts,
    genes = gene_embed$gene,
    gene.clusters = gene_embed$leiden,
    n.cores = 1L
)
program_enrichment_stats <- geneProgramSignificance(list(sim_counts$cluster_0),
    pt = sim_pseudotime$PT,
    program.labels = c("Program Name")
)
```

getFittedValues — *Generate a table of fitted values and celltype metadata for genes of interest.*

Description

Generate a table of expression counts, model fitted values, celltype metadata, etc. in order to create custom plots of gene dynamics.

Usage

```
getFittedValues(
  test.dyn.res = NULL,
  genes = NULL,
  pt = NULL,
  expr.mat = NULL,
  size.factor.offset = NULL,
  log1p.norm = TRUE,
  cell.meta.data = NULL,
  is.gee = FALSE,
  id.vec = NULL,
  ci.alpha = 0.05,
  filter.lineage = NULL
)
```

Arguments

`test.dyn.res`	The output from `testDynamic`. Defaults to NULL.
`genes`	A character vector of genes to generate fitted values for. Defaults to NULL.
`pt`	A data.frame of pseudotime values for each cell. Defaults to NULL.
`expr.mat`	Either a `SingleCellExperiment`, `Seurat`, or `cell_data_set` object from which counts can be extracted, or a matrix of integer-valued counts with genes as rows & cells as columns. Defaults to NULL.
`size.factor.offset`	(Optional) An offset to be used to rescale the fitted values. Can be generated easily with `createCellOffset`. No need to provide if the GEE backend was used. Defaults to NULL.
`log1p.norm`	(Optional) Should log1p-normalized versions of expression & model predictions be returned as well? Defaults to TRUE.
`cell.meta.data`	(Optional) A data.frame of metadata values for each cell (celltype label, subject characteristics, tissue type, etc.) that will be included in the result table. Defaults to NULL.
`is.gee`	Was the GEE mode used to fit the models? Defaults to FALSE.
`id.vec`	(Optional) A vector of subject IDs used in fitting GEE or GLMM models. Defaults to NULL.
`ci.alpha`	(Optional) The pre-specified Type I Error rate used in generating $(1-\alpha)\%$ CIs. Defaults to good old 0.05.
`filter.lineage`	(Optional) A character vector of lineages to filter out. Should be letters, i.e. lineage "A" or "B". Defaults to NULL.

Value

A data.frame containing depth- and log1p-normalized expression, model predictions, and cell-level metadata.

Author(s)

Jack R. Leary

Examples

```
data(sim_counts)
data(sim_pseudotime)
data(scLANE_models)
fitted_vals <- getFittedValues(scLANE_models,
    genes = sample(names(scLANE_models), 5),
    pt = sim_pseudotime,
    expr.mat = sim_counts
)
```

`getKnotDist` *Pull the set of knots for dynamic genes across each lineage.*

Description

Pulls knot locations for dynamic genes across each lineage, allowing comparisons of where transcriptional switches occur between lineages.

Usage

```
getKnotDist(test.dyn.res = NULL, dyn.genes = NULL)
```

Arguments

`test.dyn.res`	The output from `testDynamic`. Defaults to NULL.
`dyn.genes`	The set of genes to pull knots for. If unspecified, pulls knots for all modeled genes. Defaults to NULL.

Value

A data.frame containing gene name, lineage ID, and knot location in pseudotime.

Author(s)

Jack R. Leary

Examples

```
data(scLANE_models)
knot_dist <- getKnotDist(scLANE_models)
```

`getResultsDE`	*Tidy the results of* `testDynamic`.

Description

This function turns the nested list differential expression results of `testDynamic` and turns them into a tidy data.frame.

Usage

```
getResultsDE(test.dyn.res = NULL, p.adj.method = "fdr", fdr.cutoff = 0.01)
```

Arguments

`test.dyn.res`	The nested list returned by `testDynamic`. Defaults to NULL.
`p.adj.method`	(Optional) The method used to adjust *p*-values for multiple hypothesis testing. Defaults to "fdr".
`fdr.cutoff`	(Optional) The FDR threshold for determining statistical significance. Defaults to 0.01.

Value

A data.frame containing differential expression results & test statistics for each gene.

Author(s)

Jack R. Leary

Rhonda Bacher

See Also

`testDynamic`

`p.adjust`

Examples

```
data(scLANE_models)
scLANE_de_res <- getResultsDE(scLANE_models)
```

marge2 — *Fit* `MARGE` *models of single cell counts.*

Description

MARS fitting function for negative binomial generalized linear models (GLMs) & generalized estimating equations (GEEs).

Usage

```
marge2(
  X_pred = NULL,
  Y = NULL,
  Y.offset = NULL,
  M = 5,
  is.gee = FALSE,
  is.glmm = FALSE,
  id.vec = NULL,
  cor.structure = "ar1",
  sandwich.var = FALSE,
  approx.knot = TRUE,
  n.knot.max = 25,
  glm.backend = "MASS",
  tols_score = 1e-05,
  minspan = NULL,
  return.basis = FALSE,
  return.WIC = FALSE,
  return.GCV = FALSE
)
```

Arguments

`X_pred`	A matrix of the predictor variables. Defaults to NULL.
`Y`	The response variable. Defaults to NULL.
`Y.offset`	(Optional) An vector of per-cell size factors to be included in the final model fit as an offset. Defaults to NULL.
`M`	A set threshold for the maximum number of basis functions to be chosen. Defaults to 5.
`is.gee`	Should the `geeM` package be used to fit a negative binomial GEE? Defaults to FALSE.
`is.glmm`	Is the overall model to be fit a GLMM? Defaults to FALSE.
`id.vec`	If `is.gee = TRUE`, must be a vector of ID values for the observations. Data must be sorted such that the subjects are in order! Defaults to NULL.
`cor.structure`	If `is.gee = TRUE`, a string specifying the desired correlation structure for the NB GEE. Defaults to "ar1".
`sandwich.var`	(Optional) Should the sandwich variance estimator be used instead of the model-based estimator? Default to FALSE.

`approx.knot`	(Optional) Should the set of candidate knots be subsampled in order to speed up computation? This has little effect on the final fit, but can improve computation time somewhat. Defaults to TRUE.
`n.knot.max`	(Optional) The maximum number of candidate knots to consider. Uses uniform sampling to select this number of unique values from the reduced set of all candidate knots. Defaults to 25.
`glm.backend`	(Optional) Character specifying which GLM-fitting backend should be used. Must be one of "MASS" or "speedglm". Defaults to "MASS".
`tols_score`	(Optional) The set tolerance for monitoring the convergence for the difference in score statistics between the parent and candidate model (this is the lack-of-fit criterion used for MARGE). Defaults to 0.00001.
`minspan`	(Optional) A set minimum span value. Defaults to NULL.
`return.basis`	(Optional) Whether the basis model matrix should be returned as part of the `marge` model object. Defaults to FALSE.
`return.WIC`	(Optional) Whether the WIC matrix should be returned as part of the `marge` model object. Defaults to FALSE.
`return.GCV`	(Optional) Whether the final GCV value should be returned as part of the `marge` model object. Defaults to FALSE.

Details

- If models are being fit using an offset (as is recommended), it is assumed that the offset represents a library size factor (or similar quantity) generated using e.g., `createCellOffset` or `computeLibraryFactors`. Since this quantity represents a scaling factor divided by sequencing depth, the offset is formulated as `offset(log(1 / cell_offset))`. The inversion is necessary because the rate term, i.e. the sequencing depth, is the denominator of the estimated size factors.

Value

An object of class `marge` containing the fitted model & other optional quantities of interest (basis function matrix, GCV, etc.).

Author(s)

Jakub Stoklosa

David I. Warton.

Jack R. Leary

Rhonda Bacher

References

Friedman, J. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, **19**, 1–67.

Stoklosa, J., Gibb, H. and Warton, D.I. (2014). Fast forward selection for generalized estimating equations with a large number of predictor variables. *Biometrics*, **70**, 110–120.

Stoklosa, J. and Warton, D.I. (2018). A generalized estimating equation approach to multivariate adaptive regression splines. *Journal of Computational and Graphical Statistics*, **27**, 245–253.

See Also

backward_sel_WIC

testDynamic

createCellOffset

glm.nb

geem

Examples

```
data(sim_counts)
data(sim_pseudotime)
cell_offset <- createCellOffset(sim_counts)
marge_model <- marge2(sim_pseudotime,
    Y = BiocGenerics::counts(sim_counts)[4, ],
    Y.offset = cell_offset
)
```

---

max_span *Truncates the predictor variable value to exclude extreme values in knots selection.*

Description

Truncates the predictor variable value to exclude extreme values in knots selection.

Usage

```
max_span(X_red = NULL, q = NULL, alpha = 0.05)
```

Arguments

X_red	A vector of values for the predictor variable.
q	The number of predictors used.
alpha	See Friedman (1991) equation (45). Defaults to 0.05.

Details

Note that this equation comes from Friedman (1991) equation (45).

Value

max_span returns a vector of truncated predictor variable values.

Author(s)

Jakub Stoklosa

David I. Warton.

References

Friedman, J. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, **19**, 1–67.

Stoklosa, J. and Warton, D.I. (2018). A generalized estimating equation approach to multivariate adaptive regression splines. *Journal of Computational and Graphical Statistics*, **27**, 245–253.

`min_span` *A truncation function applied on the predictor variable for knot selection.*

Description

A truncation function applied on the predictor variable for knot selection.

Usage

```
min_span(X_red = NULL, q = NULL, minspan = NULL, alpha = 0.05)
```

Arguments

`X_red`	A vector of reduced predictor variable values. Defaults to NULL.
`q`	The number of predictor variables used. Defaults to NULL.
`minspan`	The set minimum span value. Defaults to `round((-log2(-(1 / (q * N)) * log(1 - alpha)) / 2.5))`.
`alpha`	See Friedman (1991) equation (43). Defaults to 0.05.

Details

This function selects a minimum span between the knots to mitigate runs of correlated noise in the input data and hence avoiding estimation issues, this equation comes from Friedman (1991) equation 43.

Value

`min_span` returns a vector of truncated predictor variable values.

Author(s)

Jakub Stoklosa

David I. Warton.

References

Friedman, J. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, **19**, 1–67.

Stoklosa, J. and Warton, D.I. (2018). A generalized estimating equation approach to multivariate adaptive regression splines. *Journal of Computational and Graphical Statistics*, **27**, 245–253.

modelLRT *Perform a likelihood ratio test for one model against another.*

Description

This function compares two models using a likelihood ratio test (LRT) under the assumption that the test statistic is asymptotically Chi-squared with degrees freedom equal to the difference in the number of parameters between the larger and smaller model.

Usage

```
modelLRT(mod.1 = NULL, mod.0 = NULL, is.glmm = FALSE)
```

Arguments

`mod.1`	The model corresponding to the alternative hypothesis. Defaults to NULL.
`mod.0`	The model corresponding to the null hypothesis. Defaults to NULL.
`is.glmm`	Are the models being compared GLMMs? Defaults to FALSE.

Value

A list containing the LRT test statistic, degrees freedom, and the *p*-value computed using the Chi-squared assumption.

Author(s)

Jack R. Leary

nbGAM *Fit a negative-binomial GAM.*

Description

Fits a negative-binomial family GAM using a cubic basis spline on pseudotime. If data are multi-subject in nature, a random intercept is included for each subject.

Usage

```
nbGAM(
  expr = NULL,
  pt = NULL,
  Y.offset = NULL,
  id.vec = NULL,
  penalize.spline = FALSE,
  spline.df = 5
)
```

Arguments

`expr`	A vector of integer counts. Defaults to NULL.
`pt`	A dataframe or vector of pseudotime values. Defaults to NULL.
`Y.offset`	(Optional) An offset to be included in the final model fit. Defaults to NULL.
`id.vec`	(Optional) A vector of subject IDs to be used in creating random intercepts in the GAM. Useful for comparing GAMs to GLMMs. Defaults to NULL.
`penalize.spline`	(Optional) Should a P-spline be used to fit the GAM? Otherwise the default cubic basis spline is used instead. Defaults to FALSE.
`spline.df`	(Optional) Degrees of freedom of the cubic basis spline. Unused if a P-spline is being fit, since it's estimated internally. Defaults to 5.

Value

An object of class `gamlss`

Author(s)

Jack R. Leary

See Also

`gamlss`

`random`

`bs`

`pb`

`NBI`

Examples

```
data(sim_counts)
data(sim_pseudotime)
cell_offset <- createCellOffset(sim_counts)
gam_mod <- nbGAM(BiocGenerics::counts(sim_counts)[4, ],
    pt = sim_pseudotime,
    Y.offset = cell_offset,
    penalize.spline = TRUE,
    spline.df = 10
)
```

`npConvolve` *Convolution that matches* `np.convolve`.

Description

Convolve a vector with a user-specified kernel. Can be useful for heatmap smoothing, weighted moving means, etc.

Usage

```
npConvolve(x = NULL, conv.kernel = NULL)
```

Arguments

`x`	The vector to be convolved. Defaults to NULL.
`conv.kernel`	The kernel to be used in the convolution. If unspecified, defaults to a vector of $1/n$ where n is the length of the input vector. Defaults to NULL.

Details

- The convolution here uses `convolve`, but creates the kernel and padding in such a way that it matches the output from `np.convolve` in Python's `numpy` matrix algebra package.

Value

A convolution with same length as the input vector.

Author(s)

Jack R. Leary

See Also

`convolve`

Examples

```
convolved_vec <- npConvolve(x = rnorm(20), conv.kernel = rep(1 / 5, 5))
```

`plotClusteredGenes` *Generate tidy results from* `clusterGenes` *to use in plotting.*

Description

Generate a table of per-lineage, per-cluster fitted values from `scLANE` to be used in visualizations.

Usage

```
plotClusteredGenes(
  test.dyn.res = NULL,
  gene.clusters = NULL,
  pt = NULL,
  size.factor.offset = NULL,
  n.cores = 2L
)
```

Arguments

`test.dyn.res`	The list returned by `testDynamic` - no extra processing required. Defaults to NULL.
`gene.clusters`	The data.frame returned by `clusterGenes`. Defaults to NULL.
`pt`	A data.frame containing the pseudotime or latent time estimates for each cell. Defaults to NULL.
`size.factor.offset`	(Optional) An offset to be used to rescale the fitted values. Can be generated easily with `createCellOffset`. No need to provide if the GEE backend was used. Defaults to NULL.
`n.cores`	If parallel execution is desired, how many cores should be utilized? Defaults to 2.

Details

- Due to some peculiarities of how the fitted values (on the link scale) are generated for `geeM` models, it's not necessary to multiply them by the offset as this is done internally. For GLM & GEE models, the opposite is true, and `size.factor.offset` must be provided in order to rescale the fitted values correctly.

Value

A `data.frame` with ready-to-plot tidy data. Includes columns for gene name, pseudotime lineage, cell name, fitted values on link & response scale, pseudotime, & gene cluster.

Author(s)

Jack R. Leary

See Also

`clusterGenes`

Examples

```
data(sim_counts)
data(scLANE_models)
data(sim_pseudotime)
cell_offset <- createCellOffset(sim_counts)
gene_clusters <- clusterGenes(scLANE_models,
    pt = sim_pseudotime,
    size.factor.offset = cell_offset
)
library(ggplot2)
plotClusteredGenes(
    test.dyn.res = scLANE_models,
    gene.clusters = gene_clusters,
    pt = sim_pseudotime,
    n.cores = 1L
) %>%
    ggplot(aes(x = PT, y = FITTED, color = CLUSTER, group = GENE)) +
    facet_wrap(~ LINEAGE + CLUSTER) +
    geom_line() +
    theme_classic()
```

plotModelCoefs *Plot gene dynamics with estimated coefficients.*

Description

Generate a plot of gene dynamics over a single pseudotime lineage, along with a table of coefficients across pseudotime intervals.

Usage

```
plotModelCoefs(
  test.dyn.res = NULL,
  gene = NULL,
  pt = NULL,
  expr.mat = NULL,
  size.factor.offset = NULL,
  lineage = "A",
  log1p.norm = TRUE
)
```

Arguments

`test.dyn.res`	The output from `testDynamic`. Defaults to NULL.
`gene`	A character specifying which gene's dynamics should be plotted. Defaults to NULL.
`pt`	A data.frame of pseudotime values for each cell. Defaults to NULL.
`expr.mat`	Either a `SingleCellExperiment` or `Seurat` object from which counts can be extracted, or a matrix of integer-valued counts with genes as rows & cells as columns. Defaults to NULL.
`size.factor.offset`	(Optional) An offset to be used to rescale the fitted values. Can be generated easily with `createCellOffset`. No need to provide if the GEE backend was used. Defaults to NULL.
`lineage`	A character vector specifying which lineage should be plotted. Should be letters, i.e. lineage "A" or "B". Defaults to "A".
`log1p.norm`	(Optional) Should log1p-normalized versions of expression & model predictions be returned as well? Defaults to TRUE.

Value

A `ggplot2` object displaying a gene dynamics plot & a table of coefficients across pseudotime intervals.

Author(s)

Jack R. Leary

Examples

```
data(sim_counts)
data(sim_pseudotime)
data(scLANE_models)
cell_offset <- createCellOffset(sim_counts)
scLANE_de_res <- getResultsDE(scLANE_models)
plotModelCoefs(scLANE_models,
    gene = "ACLY",
    pt = sim_pseudotime,
    expr.mat = sim_counts,
    size.factor.offset = cell_offset
)
```

`plotModels` *Plot results of* `marge` *and other models using* `ggplot2`.

Description

This function visualizes the fitted values of several types of models over the expression and pseudotime values of each cell.

Usage

```
plotModels(
  test.dyn.res = NULL,
  gene = NULL,
  pt = NULL,
  expr.mat = NULL,
  size.factor.offset = NULL,
  log1p.norm = TRUE,
  is.gee = FALSE,
  is.glmm = FALSE,
  id.vec = NULL,
  cor.structure = "ar1",
  ci.alpha = 0.05,
  plot.null = FALSE,
  plot.glm = FALSE,
  plot.gam = FALSE,
  plot.scLANE = TRUE,
  filter.lineage = NULL,
  gg.theme = theme_scLANE()
)
```

Arguments

`test.dyn.res`	The output from `testDynamic`. Defaults to NULL.
`gene`	The name of the gene that's being analyzed. Used as the title of the `ggplot` object & to subset the counts matrix. Defaults to NULL.
`pt`	A data.frame of pseudotime values for each cell. Defaults to NULL.
`expr.mat`	Either a `SingleCellExperiment` or `Seurat` object from which counts can be extracted, or a matrix of integer-valued counts. Defaults to NULL.

`size.factor.offset`	(Optional) An offset to be included in the final model fit. Can be generated easily with `createCellOffset`. Defaults to NULL.
`log1p.norm`	(Optional) Should log1p-normalized versions of expression & model predictions be returned instead of raw counts? Defaults to TRUE.
`is.gee`	Should a GEE framework be used instead of the default GLM? Defaults to FALSE.
`is.glmm`	Should a GLMM framework be used instead of the default GLM? Defaults to FALSE.
`id.vec`	If the GEE or GLMM framework is being used, a vector of subject IDs to use as input to `geem` or `glmmTMB`. Defaults to NULL.
`cor.structure`	If the GEE framework is used, specifies the desired working correlation structure. Must be one of "ar1", "independence", or "exchangeable". Defaults to "ar1".
`ci.alpha`	(Optional) The pre-specified Type I Error rate used in generating $(1-\alpha)$% CIs. Defaults to good old 0.05.
`plot.null`	(Optional) Should the fitted values from the intercept-only null model be plotted? Defaults to FALSE.
`plot.glm`	(Optional) Should the fitted values from an NB GLM be plotted? If the data are multi-subject, the "GLM" model can be a GEE or GLMM depending on the desired framework. See Examples for more detail. Defaults to FALSE.
`plot.gam`	(Optional) Should the fitted values from an NB GAM be plotted? Defaults to FALSE.
`plot.scLANE`	(Optional) Should the fitted values from the `scLANE` model be plotted? Defaults to TRUE.
`filter.lineage`	(Optional) A character vector of lineages to filter out before generating the final plot. Should be letters, i.e. lineage "A" or "B". Defaults to NULL.
`gg.theme`	(Optional) A `ggplot2` theme to be added to the plot. Defaults to `theme_scLANE`.

Value

A `ggplot` object.

Author(s)

Jack R. Leary

Examples

```
data(sim_counts)
data(scLANE_models)
data(sim_pseudotime)
cell_offset <- createCellOffset(sim_counts)
model_plot <- plotModels(scLANE_models,
    gene = names(scLANE_models)[2],
    pt = sim_pseudotime,
    expr.mat = sim_counts,
    size.factor.offset = cell_offset
)
```

print.summary.scLANE *Print method for summary.scLANE objects.*

Description

Print method for summary.scLANE objects.

Usage

```
## S3 method for class 'summary.scLANE'
print(x, ...)
```

Arguments

x	An object of class summary.scLANE.
...	Other options passed to `print.summary.scLANE`.

Value

A printed summary of overall scLANE results

Author(s)

Jack R. Leary

pullMARGESummary *Generate a summary of the MARGE model.*

Description

This function takes in the `MARGE` model fitted during the running of `marge2` and summarizes it.

Usage

```
pullMARGESummary(
  marge.model = NULL,
  is.gee = FALSE,
  sandwich.var = FALSE,
  is.glmm = FALSE
)
```

Arguments

marge.model	The `MARGE` model from `marge2`. Defaults to NULL.
is.gee	Boolean specifying whether GEE mode was used in fitting the null model. Defaults to FALSE.
sandwich.var	Boolean specifying whether the robust sandwich variance-covariance matrix should be used. Defaults to FALSE.
is.glmm	Boolean specifying whether the GLMM mode was used in fitting the model. Defaults to FALSE.

Value

A list containing a coefficient summary, fitted values and their standard errors, and the log-likelihood and deviance of the model.

Author(s)

Jack R. Leary

Rhonda Bacher

See Also

`marge2`

`pullNullSummary` *Generate a summary of the null model.*

Description

This function takes in the null model fitted during the running of `marge2` and summarizes it.

Usage

```
pullNullSummary(
  null.model = NULL,
  is.gee = FALSE,
  sandwich.var = FALSE,
  is.glmm = FALSE
)
```

Arguments

`null.model`	The null model from `marge2`. Defaults to NULL.
`is.gee`	Boolean specifying whether GEE mode was used in fitting the null model. Defaults to FALSE.
`sandwich.var`	Boolean specifying whether the robust sandwich variance-covariance matrix should be used. Defaults to FALSE.
`is.glmm`	Boolean specifying whether the GLMM mode was used in fitting the model. Defaults to FALSE.

Value

A list containing a coefficient summary, fitted values and their standard errors, and the log-likelihood and deviance of the model.

Author(s)

Jack R. Leary

Rhonda Bacher

See Also

`marge2`

`scLANE_models` *An object of class* `scLANE`.

Description

Contains the results from running `testDynamic` on `sim_counts`.

Usage

```
data(scLANE_models)
```

Format

An object of class `scLANE`.

Value

scLANE results on small simulated trajectory dataset.

`scoreTestGEE` *Use a Lagrange multiplier (score) test to compare nested GEE models.*

Description

Performs a basic Lagrange multiplier test to determine whether an alternate model is significantly better than a nested null model. This is the GEE equivalent (kind of) of `modelLRT`. Be careful with small sample sizes.

Usage

```
scoreTestGEE(
  mod.1 = NULL,
  mod.0 = NULL,
  alt.df = NULL,
  null.df = NULL,
  id.vec = NULL,
  cor.structure = "ar1"
)
```

Arguments

`mod.1`	The model under the alternative hypothesis. Must be of class `geem`. Defaults to NULL.
`mod.0`	The model under the null hypothesis. Must be of class `geem`. Defaults to NULL.
`alt.df`	The dataframe used to fit the alternative model. Defaults to NULL.
`null.df`	The dataframe used to fit the null model. Defaults to NULL.
`id.vec`	A vector of subject IDs to use as input to `marge2`. Defaults to NULL.
`cor.structure`	A string specifying the working correlation structure used to fit each model. Must be one of "ar1", "independence", or "exchangeable". Defaults to "ar1".

Details

- Calculating the test statistic involves taking the inverse of the variance of the score vector. Ideally this would be done using the true inverse, but in practice this can cause issues when the matrix is near-singular. With this in mind, we use the Moore-Penrose pseudoinverse if the original matrix inversion fails.
- The p-value is calculated using an asymptotic Chi-squared distribution, with the degrees of freedom equal to the number of non-intercept coefficients in the alternative model.

Value

A list containing the Score test statistic, a p-value, and the degrees of freedom used in the test.

Author(s)

Jack R. Leary

See Also

`geem`

`waldTestGEE`

`modelLRT`

`score_fun_gee` *Given estimates from the null model fit and the design matrix for alternative model, find the score statistic (this is used for GEEs only).*

Description

Calculate the score statistic for a GEE model.

Usage

```
score_fun_gee(
  Y = NULL,
  N = NULL,
  n_vec = NULL,
  VS.est_list = NULL,
  AWA.est_list = NULL,
  J2_list = NULL,
  Sigma2_list = NULL,
  J11.inv = NULL,
  JSigma11 = NULL,
  mu.est = NULL,
  V.est = NULL,
  B1 = NULL,
  XA = NULL
)
```

Arguments

`Y`	The response variable. Defaults to NULL.
`N`	The number of clusters. Defaults to NULL.
`n_vec`	A vector consisting of the cluster sizes for each cluster. Defaults to NULL.
`VS.est_list`	A product of matrices. Defaults to NULL.
`AWA.est_list`	A product of matrices. Defaults to NULL.
`J2_list`	A product of matrices. Defaults to NULL.
`Sigma2_list`	A product of matrices. Defaults to NULL.
`J11.inv`	A product of matrices. Defaults to NULL.
`JSigma11`	A product of matrices. Defaults to NULL.
`mu.est`	Estimates of the fitted mean under the null model. Defaults to NULL.
`V.est`	Estimates of the fitted variance under the null model. Defaults to NULL.
`B1`	Design matrix under the null model. Defaults to NULL.
`XA`	Design matrix under the alternative model. Defaults to NULL.

Value

A calculated score statistic for the null and alternative model when fitting a GEE.

Author(s)

Jakub Stoklosa

David I. Warton

Jack R. Leary

References

Stoklosa, J., Gibb, H. and Warton, D.I. (2014). Fast forward selection for generalized estimating equations with a large number of predictor variables. *Biometrics*, **70**, 110–120.

Stoklosa, J. and Warton, D.I. (2018). A generalized estimating equation approach to multivariate adaptive regression splines. *Journal of Computational and Graphical Statistics*, **27**, 245–253.

See Also

`score_fun_glm`

score_fun_glm *Given estimates from the null model fit and the design matrix for alternative model, find the score statistic (this is used for GLMs only).*

Description

Calculate the score statistic for a GLM model.

Usage

```
score_fun_glm(
  Y = NULL,
  VS.est_list = NULL,
  A_list = NULL,
  B1_list = NULL,
  mu.est = NULL,
  V.est = NULL,
  B1 = NULL,
  XA = NULL
)
```

Arguments

`Y`	The response variable. Defaults to NULL.
`VS.est_list`	A product of matrices. Defaults to NULL.
`A_list`	A product of matrices. Defaults to NULL.
`B1_list`	A product of matrices. Defaults to NULL.
`mu.est`	Estimates of the fitted mean under the null model. Defaults to NULL.
`V.est`	Estimates of the fitted variance under the null model. Defaults to NULL.
`B1`	Design matrix under the null model. Defaults to NULL.
`XA`	Design matrix under the alternative model. Defaults to NULL.

Value

A calculated score statistic for the null and alternative model when fitting a GLM.

Author(s)

Jakub Stoklosa

David I. Warton

Jack R. Leary

References

Stoklosa, J., Gibb, H. and Warton, D.I. (2014). Fast forward selection for generalized estimating equations with a large number of predictor variables. *Biometrics*, **70**, 110–120.

Stoklosa, J. and Warton, D.I. (2018). A generalized estimating equation approach to multivariate adaptive regression splines. *Journal of Computational and Graphical Statistics*, **27**, 245–253.

See Also

`score_fun_gee`

`sim_counts` *A* `SingleCellExperiment` *object containing simulated counts.*

Description

Data simulated using the `scaffold` R package for 50 dynamic and 50 static genes across 1200 cells from 3 subjects.

Usage

```
data(sim_counts)
```

Format

An object of class `SingleCellExperiment`.

Value

Small simulated trajectory dataset.

Source

https://www.rhondabacher.com/scaffold-vignette.pdf

`sim_pseudotime` *A data.frame containing ground-truth pseudotime.*

Description

The true ordering of the 1200 cells contained in `sim_counts`.

Usage

```
data(sim_pseudotime)
```

Format

An object of class `data.frame` with 1200 rows and one variable:

- PT: the true pseudotime (0.0025–1)

Value

Small simulated trajectory pseudotime.

smoothedCountsMatrix *Generate a smoothed matrix of gene expression using* scLANE *models.*

Description

This function takes as input the output from testDynamic and returns the fitted values from each model in a wide format, with one column per-gene and one row-per cell. This matrix can be use as input to cell or gene clustering and / or visualizations such as heatmaps.

Usage

```
smoothedCountsMatrix(
  test.dyn.res = NULL,
  size.factor.offset = NULL,
  pt = NULL,
  genes = NULL,
  log1p.norm = FALSE,
  n.cores = 2L
)
```

Arguments

test.dyn.res	The list returned by testDynamic - no extra processing required. Defaults to NULL.
size.factor.offset	(Optional) An offset to be used to rescale the fitted values. Can be generated easily with createCellOffset. No need to provide if the GEE backend was used. Defaults to NULL.
pt	A data.frame of pseudotime values for each cell. Defaults to NULL.
genes	(Optional) A character vector of genes with which to subset the results. Defaults to NULL.
log1p.norm	A boolean specifying whether the smoothed counts should be log1p-transformed after depth normalization. Defaults to FALSE.
n.cores	If parallel execution is desired, how many cores should be utilized? Defaults to 2.

Value

A list of matrices of smoothed counts, with each element of the list being a single pseudotime lineage.

Author(s)

Jack R. Leary

See Also

testDynamic

Examples

```
data(sim_pseudotime)
data(scLANE_models)
smoothed_dynamics <- smoothedCountsMatrix(scLANE_models,
    pt = sim_pseudotime,
    n.cores = 1L
)
```

`sortGenesHeatmap` *Sort genes by where their peak expression occurs across pseudotime.*

Description

Sort genes such that genes with peak expression occurring earlier in pseudotime are first, and vice versa for genes with late peak expression. Useful for ordering genes in order to create heatmaps of expression cascades.

Usage

```
sortGenesHeatmap(heatmap.mat = NULL, pt.vec = NULL)
```

Arguments

`heatmap.mat`	A matrix of raw or smoothed expression values with genes as columns and cells as rows. Defaults to NULL.
`pt.vec`	A numeric vector of pseudotime values for each cell i.e., for each row in the heatmap matrix. Defaults to NULL.

Value

A character vector of genes sorted by their peak expression values over pseudotime.

Author(s)

Jack R. Leary

See Also

`smoothedCountsMatrix`

Examples

```
data(sim_pseudotime)
data(scLANE_models)
smoothed_counts <- smoothedCountsMatrix(scLANE_models,
    pt = sim_pseudotime,
    n.cores = 1L
)
sorted_genes <- sortGenesHeatmap(smoothed_counts$Lineage_A,
                        pt.vec = sim_pseudotime$PT)
```

`sortObservations` *Sort observations by sample ID and pseudotime.*

Description

Since the GEE & GLMM modes require data to be sorted by sample ID and pseudotime, this function provides a simple way to do so for a range of inputs.

Usage

```
sortObservations(expr.mat = NULL, pt.vec = NULL, id.vec = NULL)
```

Arguments

`expr.mat`	Either a `SingleCellExperiment`, `Seurat`, or `cell_data_set` object from which cell-level metadata can be extracted, or a matrix of integer-valued counts with genes as rows & cells as columns. Defaults to NULL.
`pt.vec`	A vector of pseudotime values used to sort the observations. May contain NA values. Defaults to NULL.
`id.vec`	A vector of subject IDs used to sort the observations. Defaults to NULL.

Details

- If the input is a matrix, it is assumed that the columns are cells - and are named as such - and the rows are genes.
- If the input is a `Seurat` object, sorting requires converting to `SingleCellExperiment` object first, then ordering, then converting back to a `Seurat` object. Some information might be lost, so it is recommended not to overwrite your original `Seurat` object.

Value

An object of the same class as the input `expr.mat`, but sorted by sample ID & pseudotime.

Author(s)

Jack R. Leary

Examples

```
data(sim_counts)
data(sim_pseudotime)
sorted_counts <- sortObservations(sim_counts,
    pt.vec = sim_pseudotime$PT,
    id.vec = sim_counts$subject
)
```

`stat_out` *Fits a linear regression model and calculates RSS/GCV measures (used for MARS linear models).*

Description

Calculate the final RSS / GCV for a fitted model.

Usage

```
stat_out(Y = NULL, B1 = NULL, TSS = NULL, GCV.null = NULL, pen = 2)
```

Arguments

`Y`	The response variable. Defaults to NULL.
`B1`	The model matrix of predictor variables. Defaults to NULL.
`TSS`	Total sum of squares. Defaults to NULL.
`GCV.null`	GCV value for the intercept model. Defaults to NULL.
`pen`	The set/fixed penalty used for the GCV. Defaults to 2.

Value

A list consisting of the RSS, RSSq1, GCV1 and GCVq1 values for the fitted model.

Author(s)

Jakub Stoklosa

David I. Warton

Jack R. Leary

References

Friedman, J. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, **19**, 1–67.

Stoklosa, J. and Warton, D.I. (2018). A generalized estimating equation approach to multivariate adaptive regression splines. *Journal of Computational and Graphical Statistics*, **27**, 245–253.

See Also

`stat_out_score_glm_null`

stat_out_score_gee_null

Calculate part of the score statistic for a GEE.

Description

A function that calculates parts of the score statistic for GEEs only (it is used for the full path for forward selection).

Usage

```
stat_out_score_gee_null(
  Y = NULL,
  B_null = NULL,
  id.vec = NULL,
  cor.structure = NULL,
  theta.hat = NULL
)
```

Arguments

`Y`	The response variable Defaults to NULL.
`B_null`	The design matrix matrix under the null model Defaults to NULL.
`id.vec`	A vector of ID values for the observations. Defaults to NULL.
`cor.structure`	A string specifying the desired correlation structure for the NB GEE. Defaults to NULL.
`theta.hat`	Estimated value to be treated as the "known" theta when passing the negative binomial family to `geeM`. Defaults to NULL.

Value

A list of values (mainly products of matrices) that make up the final score statistic calculation (required for another function).

Author(s)

Jakub Stoklosa

David I. Warton

Jack R. Leary

References

Stoklosa, J., Gibb, H. and Warton, D.I. (2014). Fast forward selection for generalized estimating equations with a large number of predictor variables. *Biometrics*, **70**, 110–120.

Stoklosa, J. and Warton, D.I. (2018). A generalized estimating equation approach to multivariate adaptive regression splines. *Journal of Computational and Graphical Statistics*, **27**, 245–253.

See Also

`stat_out`

`stat_out_score_glm_null`

`stat_out_score_glm_null` *Calculate part of the score statistic for a GLM.*

Description

A function that calculates parts of the score statistic for GLMs only (it is used for the full path for forward selection).

Usage

```
stat_out_score_glm_null(Y = NULL, B_null = NULL)
```

Arguments

`Y`	: The response variable Defaults to NULL.
`B_null`	: Design matrix under the null model. Defaults to NULL.

Value

A list of values (mainly products of matrices) that make up the final score statistic calculation (required for another function).

Author(s)

Jakub Stoklosa

David I. Warton

Jack R. Leary

References

Stoklosa, J., Gibb, H. and Warton, D.I. (2014). Fast forward selection for generalized estimating equations with a large number of predictor variables. *Biometrics*, **70**, 110–120.

Stoklosa, J. and Warton, D.I. (2018). A generalized estimating equation approach to multivariate adaptive regression splines. *Journal of Computational and Graphical Statistics*, **27**, 245–253.

See Also

`stat_out`

`stat_out_score_gee_null`

`stripGLM` *Make GLM objects much smaller.*

Description

This function removes a *lot* of components from the default GLM object in order to make it take up less memory. It does however retain enough pieces for `predict()` to still work. No promises beyond that.

Usage

```
stripGLM(glm.obj = NULL)
```

Arguments

`glm.obj`	An object of class GLM from which you'd like to strip out unnecessary components. Defaults to NULL.

Value

A slimmed-down `glm` object.

Author(s)

Jack R. Leary

See Also

`glm.nb`

`summarizeModel` *Represent a `marge` model as a series of piecewise equations.*

Description

This function summarizes the model for each gene and allows for quantitative interpretation of fitted gene dynamics.

Usage

```
summarizeModel(marge.model = NULL, pt = NULL, is.glmm = FALSE)
```

Arguments

`marge.model`	The fitted model output from `marge2` (this function is internally called by `testDynamic`). Defaults to NULL.
`pt`	The predictor matrix of pseudotime. Defaults to NULL.
`is.glmm`	Flag specifying whether or not data were generated using GLMM mode. Defaults to FALSE.

Value

A data.frame of the model coefficients, cutpoint intervals, and formatted equations.

Author(s)

Jack R. Leary

Rhonda Bacher

See Also

`marge2`

Examples

```
data(sim_counts)
data(sim_pseudotime)
cell_offset <- createCellOffset(sim_counts)
marge_model <- marge2(sim_pseudotime,
    Y = BiocGenerics::counts(sim_counts)[4, ],
    Y.offset = cell_offset
)
model_summary <- summarizeModel(marge.model = marge_model,
                            pt = sim_pseudotime)
```

`summary.scLANE` *Summary method for scLANE objects.*

Description

Summary method for scLANE objects.

Usage

```
## S3 method for class 'scLANE'
summary(object, ...)
```

Arguments

`object`	The nested list returned by `testDynamic`.
`...`	Other options passed to `summary.scLANE`. Defaults to NULL.

Value

A summary list with aggregated statistics concerning the trajectory DE tests from `scLANE`.

Author(s)

Jack R. Leary

Examples

```
data(scLANE_models)
summary(scLANE_models)
```

`testDynamic` *Test whether a gene is dynamic over pseudotime.*

Description

This function tests whether a NB `marge` model is better than a null (intercept-only) model using the Likelihood Ratio Test. In effect, the test tells us whether a gene's expression changes (in any way) over pseudotime.

Usage

```
testDynamic(
  expr.mat = NULL,
  pt = NULL,
  genes = NULL,
  size.factor.offset = NULL,
  is.gee = FALSE,
  cor.structure = "ar1",
  gee.bias.correction.method = NULL,
  gee.test = "wald",
  is.glmm = FALSE,
  glmm.adaptive = TRUE,
  id.vec = NULL,
  n.potential.basis.fns = 5,
  n.cores = 4L,
  approx.knot = TRUE,
  verbose = TRUE,
  random.seed = 312
)
```

Arguments

`expr.mat`	Either a `SingleCellExperiment`, `Seurat`, or `CellDataSet` object from which counts can be extracted, or a matrix of integer-valued counts with genes as rows & cells as columns. Defaults to NULL.
`pt`	Either the output from `SlingshotDataSet` object from which pseudotime can be generated, or a data.frame containing the pseudotime or latent time estimates for each cell (can be multiple columns / lineages). Defaults to NULL.
`genes`	A character vector of genes to model. If not provided, defaults to all genes in `expr.mat`. Defaults to NULL.
`size.factor.offset`	(Optional) An offset to be included in the final model fit. Can be generated easily with `createCellOffset`. Defaults to NULL.
`is.gee`	Should a GEE framework be used instead of the default GLM? Defaults to FALSE.
`cor.structure`	If the GEE framework is used, specifies the desired working correlation structure. Must be one of "ar1", "independence", or "exchangeable". Defaults to "ar1".

`gee.bias.correction.method`	(Optional) Specify which small-sample bias correction to be used on the sandwich variance-covariance matrix prior to test statistic estimation. Options are "kc" and "df". Defaults to NULL, indicating the use of the model-based variance.
`gee.test`	A string specifying the type of test used to estimate the significance of the full model. Must be one of "wald" or "score". Defaults to "wald".
`is.glmm`	Should a GLMM framework be used instead of the default GLM? Defaults to FALSE.
`glmm.adaptive`	(Optional) Should the basis functions for the GLMM be chosen adaptively? If not, uses 4 evenly spaced knots. Defaults to TRUE.
`id.vec`	If a GEE or GLMM framework is being used, a vector of subject IDs to use as input to `geem` or `glmmTMB`. Defaults to NULL.
`n.potential.basis.fns`	(Optional) The maximum number of possible basis functions. See the parameter `M` in `marge2`. Defaults to 5.
`n.cores`	(Optional) If running in parallel, how many cores should be used? Defaults to 4L.
`approx.knot`	(Optional) Should the knot space be reduced in order to improve computation time? Defaults to TRUE.
`verbose`	(Optional) A boolean indicating whether a progress bar should be printed to the console. Defaults to TRUE.
`random.seed`	(Optional) The random seed used to initialize RNG streams in parallel. Defaults to 312.

Details

- If `expr.mat` is a `Seurat` object, counts will be extracted from the output of `DefaultAssay`. If using this functionality, check to ensure the specified assay is correct before running the function. If the input is a `SingleCellExperiment` or `CellDataSet` object, the raw counts will be extracted with `counts`.
- If using the GEE or GLMM model architectures, ensure that the observations are sorted by subject ID (this is assumed by the underlying fit implementations). If they are not, the models will error out.
- If `gee.bias.correction.method` is set to "kc" or "df", a bias adjustment will be used to inflate the robust variance-covariance matrix prior to estimating the Wald test statistic. This is useful when the number of subjects is small and / or the number of per-subject observations is very large. Doing so will remove the bias in the sandwich estimator in small-sample cases. Currently, we suggest keeping this NULL and using the model-based variance estimates and specifying the "ar1" correlation structure.

Value

A list of lists, where each element is a gene and each gene contains sublists for each lineage. Each gene-lineage sublist contains a gene name, lineage number, default `marge` vs. null model test results, model statistics, and fitted values. Use `getResultsDE` to tidy the results.

Author(s)

Jack R. Leary

See Also

getResultsDE

testSlope

marge2

glm.nb

geem

glmmTMB

Examples

```
data(sim_counts)
data(sim_pseudotime)
cell_offset <- createCellOffset(sim_counts)
scLANE_models <- testDynamic(sim_counts,
    pt = sim_pseudotime,
    size.factor.offset = cell_offset,
    n.cores = 1L
)
```

---

testSlope *Test whether a gene is dynamic over a pseudotime interval.*

Description

This function tests whether each gene's estimated β for pseudotime differs significantly from 0 over each empirically estimated sets of knots / pseudotime intervals using a Wald test.

Usage

```
testSlope(test.dyn.res = NULL, p.adj.method = "fdr", fdr.cutoff = 0.01)
```

Arguments

test.dyn.res	The list returned by testDynamic - no extra processing required. Defaults to NULL.
p.adj.method	The method used to adjust the *p*-values for each coefficient. Defaults to "fdr".
fdr.cutoff	The FDR threshold for determining statistical significance. Defaults to 0.01.

Value

A dataframe containing the genes, breakpoints, and coefficient *p*-values from each model.

Author(s)

Jack R. Leary

See Also

testDynamic

p.adjust

Examples

```
data(scLANE_models)
slope_test_res <- testSlope(scLANE_models)
```

---

theme_scLANE — *A* `ggplot2` *theme for* `scLANE`.

Description

A publication-ready theme for creating gene dynamics plots, embedding plots, etc.

Usage

```
theme_scLANE(
  base.size = 12,
  base.lwd = 0.75,
  base.family = "sans",
  umap = FALSE
)
```

Arguments

`base.size`	The base font size. Defaults to 12.
`base.lwd`	The base linewidth. Defaults to 0.75.
`base.family`	The font family to be used throughout. Defaults to "sans".
`umap`	(Optional) If set to TRUE, removes axis text and ticks for a cleaner look. Defaults to FALSE.

Value

A `ggplot2` theme.

Author(s)

Jack R. Leary

Examples

```
data(sim_counts)
data(scLANE_models)
data(sim_pseudotime)
cell_offset <- createCellOffset(sim_counts)
model_plot <- plotModels(scLANE_models,
    gene = names(scLANE_models)[1],
    pt = sim_pseudotime,
    expr.mat = sim_counts,
    size.factor.offset = cell_offset
) +
    theme_scLANE()
```

tp1 *Truncated p-th power function (positive part).*

Description

Truncated p-th power function (positive part).

Usage

```
tp1(x = NULL, t = NULL, p = 1)
```

Arguments

x	A predictor variable value. Defaults to NULL.
t	A specified knot value. Defaults to NULL.
p	The p^{th} degree of the polynomial considered. Defaults to 1.

Value

A vector of values that have been transformed using a truncated p^{th} power function (positive part) for a specified knot value.

Author(s)

Jakub Stoklosa

David I. Warton

References

Friedman, J. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, **19**, 1–67.

Stoklosa, J. and Warton, D.I. (2018). A generalized estimating equation approach to multivariate adaptive regression splines. *Journal of Computational and Graphical Statistics*, **27**, 245–253.

See Also

tp2

tp2 *Truncated p-th power function (negative part).*

Description

Truncated p-th power function (negative part).

Usage

```
tp2(x = NULL, t = NULL, p = 1)
```

Arguments

x	A predictor variable value. Defaults to NULL.
t	A specified knot value. Defaults to NULL.
p	The p^{th} degree of the polynomial considered. Defaults to 1.

Value

A vector of values that have been transformed using a truncated p^{th} power function (negative part) for a specified knot value.

Author(s)

Jakub Stoklosa

David I. Warton

References

Friedman, J. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, **19**, 1–67.

Stoklosa, J. and Warton, D.I. (2018). A generalized estimating equation approach to multivariate adaptive regression splines. *Journal of Computational and Graphical Statistics*, **27**, 245–253.

See Also

tp1

---

waldTestGEE *Use a Wald test to compare nested GEE models.*

Description

Performs a basic Wald test to determine whether an alternate model is significantly better than a nested null model. This is the GEE equivalent (kind of) of modelLRT. Be careful with small sample sizes.

Usage

```
waldTestGEE(
  mod.1 = NULL,
  mod.0 = NULL,
  correction.method = NULL,
  id.vec = NULL,
  verbose = FALSE
)
```

Arguments

`mod.1`	The model under the alternative hypothesis. Must be of class `geem`. Defaults to NULL.
`mod.0`	The model under the null hypothesis. Must be of class `geem`. Defaults to NULL.
`correction.method`	A string specifying the correction method to be used. Currently supported options are "df", "kc", and NULL. Defaults to NULL.
`id.vec`	A vector of subject IDs. Required when `correction.method` is "kc". Defaults to NULL.
`verbose`	(Optional) A Boolean specifying whether or not verbose output should be printed to the console. Occasionally useful for debugging. Defaults to FALSE.

Details

- Calculating the test statistic involves taking the inverse of the variance-covariance matrix of the coefficients. Ideally this would be done using the "true" inverse with something like `solve`, `qr.solve`, or `chol2inv`, but in practice this can cause issues when the variance-covariance matrix is near-singular. With this in mind, we use the Moore-Penrose pseudoinverse as implemented in `ginv` instead, which leads to more stable results.
- The *p*-value is calculated using an asymptotic Chi-squared distribution, with the degrees of freedom equal to the number of non-intercept coefficients in the alternative model.

Value

A list containing the Wald test statistic, a *p*-value, and the degrees of freedom used in the test.

Author(s)

Jack R. Leary

See Also

`geem`

`biasCorrectGEE`

`modelLRT`

Index