Package 'CLAMP'

June 23, 2026

Type Package

Title Curated Latent-variable Analysis with Molecular Priors

Version 0.99.3

Description CLAMP performs prior-informed latent variable decomposition of high-dimensional transcriptomic data. It integrates curated gene sets to learn biologically interpretable latent variables, supports file-backed matrices for large datasets, and provides tools for preprocessing, normalization, projection, and evaluation of latent structures. CLAMP is designed to scale to tens of thousands of samples, making it suitable for large public resources such as recount3 and ARCHS4. It enables researchers to uncover biologically meaningful patterns that connect genes, pathways, and complex traits in transcriptomics studies.

License GPL-3

URL https://chikinalab.github.io/CLAMP/, https://github.com/chikinalab/CLAMP

BugReports https://github.com/chikinalab/CLAMP/issues

Depends R (>= 4.6.0),

Imports bigstatsr, circlize, ComplexHeatmap, dplyr, ggplot2, ggrepel, glmnet, grid, irlba, Matrix, matrixStats, methods, patchwork, Rcpp, rlang, rsvd, stats, utils

Suggests AnnotationDbi, BiocStyle, CLAMPData, data.table, DiagrammeR, DT, rhdf5, here, knitr, org.Hs.eg.db, PCAtools, magick, rmarkdown, testthat (>= 3.0.0)

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

biocViews Software, GeneExpression, RNASeq, Transcriptomics, DimensionReduction, Visualization, Normalization, Pathways, Preprocessing, GenePrediction, GeneTarget

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

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

git_branch devel

git_last_commit 1768d3b

git_last_commit_date 2026-06-08

Repository Bioconductor 3.24

Date/Publication 2026-06-23

Author Marc Subirana-Granes [aut, cre] (ORCID:
<https://orcid.org/0000-0003-3934-839X>),
Maria Chikina [aut],
National Human Genome Research Institute [fnd] (R00 HG011898 to M.P.;
R01 HG009299-6A1 to M.C.),
Eunice Kennedy Shriver National Institute of Child Health and Human
Development [fnd] (R01 HD109765 to M.P.),
National Science Foundation [fnd] (NSF 2238125 to M.C.),
National Eye Institute [fnd] (NIH R01 EY030546-01A1 to M.C.)

Maintainer Marc Subirana-Granes <mb2subi@gmail.com>

Contents

`allAgainstAllAUCs` *Compute all-vs-all AUC matrix*

Description

Calculates the area under the ROC curve (AUC) for all pairs of columns between a prediction matrix `B` and binary targets in `target`. Each column of `B` is ranked, and AUC is computed based on how well the ranks separate positive vs. negative samples in each target column.

Usage

```
allAgainstAllAUCs(B, target)
```

Arguments

`B`	A numeric matrix of predictions (samples × features).
`target`	A binary matrix of the same number of rows as `B` (samples × targets), where 1 indicates positive and 0 indicates negative.

Value

A numeric matrix of AUC values (features × targets).

Examples

```
set.seed(1)
B <- matrix(rnorm(100), nrow = 20)
target <- matrix(sample(0:1, 40, replace = TRUE), nrow = 20)
allAgainstAllAUCs(B, target)
```

AUC *Compute AUC using Wilcoxon rank-sum test*

Description

Computes the area under the ROC curve (AUC) by applying a Wilcoxon rank-sum test between predicted values for positive and negative labels. This is equivalent to computing the Mann-Whitney U statistic.

Usage

```
AUC(labels, values)
```

Arguments

labels	A numeric or logical vector indicating class labels. Values > 0 are treated as positive.
values	A numeric vector of prediction scores corresponding to labels.

Value

A list with:

auc Estimated AUC, or 0.5 if one class is missing

pval Wilcoxon test p-value, or NA if one class is missing

BH *Adjust p-values using Benjamini-Hochberg method*

Description

Applies the BH (Benjamini-Hochberg) correction for multiple hypothesis testing.

Usage

```
BH(p)
```

Arguments

p	Numeric vector of p-values.

Value

Adjusted p-values.

---

binarizeTop *Binarize matrix by top-k values per column*

Description

Keeps only the top `top` values in each column of a matrix, setting others to 0.

Usage

```
binarizeTop(Z, top, keepVals = TRUE)
```

Arguments

Z	A numeric matrix.
top	Number of top entries to keep in each column.
keepVals	If `TRUE`, retains original values above the cutoff; otherwise, sets them to 1.

Value

A modified matrix with only top entries retained per column.

---

celltypeTargets *Cell-type deconvolution matrix*

Description

A numeric matrix of estimated cell-type proportions for whole-blood samples. Rows correspond to sample IDs and columns to major immune cell types. This dataset can be used for validation or illustrative purposes in CLAMP analyses.

Usage

```
data(celltypeTargets)
```

Format

A numeric matrix with samples as rows and cell types as columns. Row names are sample identifiers.

Value

A numeric matrix of cell-type proportions.

Examples

```
data(celltypeTargets)
```

`CLAMPbase` *CLAMP base matrix factorization*

Description

Runs the core matrix factorization procedure of CLAMP, decomposing the gene expression matrix `Y` into latent variables `Z` and loadings `B`. It supports sparse, dense, and Filebacked Big Matrices (FBM) as input and includes options for adaptive sparsity, positive constraints, and regularization.

Usage

```
CLAMPbase(
  Y,
  clamp_k = NULL,
  svd_k = NULL,
  svdres = NULL,
  L1 = NULL,
  L2 = NULL,
  Zpos = TRUE,
  max.iter = 200,
  tol = 5e-04,
  trace = FALSE,
  rseed = NULL,
  B = NULL,
  scale = 1,
  pos.adj = 3,
  adaptive.p = 0.05,
  adaptive.iter = 20,
  cutoff = 0,
  ncores = 1,
  clamp_k_method = c("elbow", "permutation", "gavish_donoho", "scaleSVs")
)
```

Arguments

`Y`	Input gene expression matrix (genes x samples). Can be dense, sparse (`dgCMatrix`), or FBM.
`clamp_k`	Number of latent variables for CLAMP (final model rank). If `NULL`, it is chosen automatically via `select_clamp_k()`.
`svd_k`	Number of singular values/components to compute in the SVD. If `NULL`, defaults to `max(2, min(n_genes, n_samples) - 1)`.
`svdres`	Optional precomputed SVD result. If not supplied, it is computed internally.
`L1`	L1 regularization strength for Z. Defaults to scaled singular value.
`L2`	L2 regularization strength for B. Defaults to scaled singular value.
`Zpos`	Logical; if `TRUE`, negative entries in Z are zeroed. Default is `TRUE`.
`max.iter`	Maximum number of optimization iterations. Default is 200.

tol	Convergence tolerance for B update. Default is 5e-4.
trace	Logical; if TRUE, prints progress. Default is FALSE.
rseed	Optional integer for reproducible random initialization of B.
B	Optional initial matrix for B. If not provided, initialized from SVD.
scale	Scaling factor for L1 and L2 when not provided. Default is 1.
pos.adj	Positive constraint adjustment divisor for L1. Default is 3.
adaptive.p	Controls adaptive sparsity in Z. After each ALS update, negative entries in Z are assumed to reflect noise. The cutoff for thresholding is set according to the probability of positive values under a reflected negative distribution-effectively zeroing out small positive entries likely to be noise. Smaller values lead to more sparsity. Default is 0.05.
adaptive.iter	Number of iterations before adaptive sparsity is applied. Default is 20.
cutoff	Scalar threshold to zero Z values when Zpos = TRUE and adaptive thresholding is not used. Default is 0.
ncores	Number of cores to use for parallel computation (only used if Y is an FBM). Default is 1.
clamp_k_method	Method for selecting clamp_k when not provided. One of "elbow" (default), "permutation", "gavish_donoho", or "scaleSVs". Passed to select_clamp_k().

Details

This function is the low-level implementation of CLAMP. It alternates between solving for Z given B and solving for B given Z, with optional sparsity and non-negativity constraints on Z. Convergence is assessed via relative change in B.

Value

A list with components:

B Latent variable loadings (LVs x genes)

Z Latent variable scores (LVs x samples)

Zraw Raw Z matrix before thresholding

L1 Final value of L1 used

L2 Final value of L2 used

Examples

```
# small toy dataset: 5 genes x 4 samples
Y <- matrix(rnorm(5 * 4), nrow = 5, ncol = 4)
# run a single iteration for speed
res <- CLAMPbase(Y, clamp_k = 2, max.iter = 1, trace = FALSE)
# inspect dimensions of B and Z
dim(res$B)
dim(res$Z)
```

CLAMPdotplot *Dot plot of top pathways for a single latent variable*

Description

Lollipop-style dot plot showing the top pathways associated with one selected LV. Dot size encodes AUC; dot colour encodes `-log10(FDR)`. The x-axis and pathway ordering can use either AUC or `-log10(FDR)`.

Usage

```
CLAMPdotplot(
  clampRes,
  lv = 1,
  top = 20,
  auc.cutoff = 0.6,
  fdr.cutoff = 0.05,
  max.name.len = 50,
  x.axis = c("AUC", "-log10(FDR)", "log10FDR"),
  order.by = x.axis
)
```

Arguments

`clampRes`	A CLAMP result list containing a `summary` data frame, or the summary data frame itself.
`lv`	LV to plot: either a numeric index (e.g. `1` -> `"LV1"`) or a character name (e.g. `"LV3"`). Default `1`.
`top`	Maximum number of pathways to display, chosen by `order.by`. Default `20`.
`auc.cutoff`	Minimum AUC to display. Default `0.6`.
`fdr.cutoff`	Maximum FDR to display. Default `0.05`.
`max.name.len`	Maximum characters for pathway label trimming. Default `50`.
`x.axis`	Metric to place on the x-axis. Use `"AUC"` or `"-log10(FDR)"`. `"log10FDR"` is also accepted. Default `"AUC"`.
`order.by`	Metric used to choose the top pathways and order the y-axis. Use `"AUC"` or `"-log10(FDR)"`. `"log10FDR"` is also accepted. Defaults to `x.axis`.

Value

Invisibly returns a `ggplot2::ggplot()` object.

Examples

```
set.seed(9)
pathways <- paste0("Pathway_", 1:20)
lvs <- paste0("LV", 1:5)
nr <- length(pathways) * length(lvs)

summ <- data.frame(
    pathway = rep(pathways, length(lvs)),
```

```
    LV = rep(lvs, each = length(pathways)),
    AUC = runif(nr, 0.5, 1.0),
    FDR = runif(nr, 0, 0.05),
    stringsAsFactors = FALSE
)

CLAMPdotplot(list(summary = summ), lv = 1, top = 10)
CLAMPdotplot(list(summary = summ),
    lv = "LV2", x.axis = "-log10(FDR)",
    order.by = "-log10(FDR)"
)
```

`CLAMPdotplotAll` *Dot plot of pathway-LV associations across all latent variables*

Description

Produces a dot plot where each point represents one pathway × LV pair. Dot size encodes the AUC value and dot colour encodes `-log10(FDR)`. Only associations passing `auc.cutoff` and `fdr.cutoff` are shown.

Usage

```
CLAMPdotplotAll(
  clampRes,
  auc.cutoff = 0.6,
  fdr.cutoff = 0.05,
  top.per.lv = NULL,
  max.name.len = 40
)
```

Arguments

`clampRes`	A CLAMP result list containing a `summary` data frame with columns `pathway`, `LV`, `AUC`, and `FDR`. Alternatively, `clampRes` may be the summary data frame itself.
`auc.cutoff`	Minimum AUC to display. Default `0.6`.
`fdr.cutoff`	Maximum FDR to display. Default `0.05`.
`top.per.lv`	Maximum number of pathways to display per LV, chosen by highest AUC. `NULL` shows all. Default `NULL`.
`max.name.len`	Maximum characters for pathway label trimming. Default `40`.

Value

Invisibly returns a `ggplot2::ggplot()` object.

Examples

```
set.seed(9)
pathways <- paste0("Pathway_", 1:20)
lvs <- paste0("LV", 1:5)
nr <- length(pathways) * length(lvs)
```

```
summ <- data.frame(
    pathway = rep(pathways, length(lvs)),
    LV = rep(lvs, each = length(pathways)),
    AUC = runif(nr, 0.5, 1.0),
    FDR = runif(nr, 0, 0.2),
    stringsAsFactors = FALSE
)

CLAMPdotplotAll(list(summary = summ), auc.cutoff = 0.6, fdr.cutoff = 0.15)
```

`CLAMPfull` *Runs the streamlined full CLAMP model.*

Description

This version performs latent-variable decomposition of a gene expression matrix `Y` guided by prior pathway annotations `priorMat`, with simplified and lighter regularization compared to the original extended CLAMP variant. The algorithm alternates updates of `Z`, `B`, and `U`, where `U` captures pathway-latent variable associations inferred directly from the data without ridge-regularized projections (`Chat` is not used).

Usage

```
CLAMPfull(
  Y,
  priorMat,
  Chat = NULL,
  svdres = NULL,
  clamp.base.result = NULL,
  clamp_k = NULL,
  svd_k = NULL,
  L1 = NULL,
  L2 = NULL,
  cvn = 5,
  max.iter = 30,
  trace = TRUE,
  maxPath = 10,
  doCrossval = TRUE,
  penalty.factor = rep(1, ncol(priorMat)),
  glm_alpha = 0.9,
  minGenes = 0,
  tol = 5e-04,
  seed = 123456,
  allGenes = FALSE,
  rseed = NULL,
  max.U.updates = Inf,
  pathwaySelection = c("fast", "complete"),
  multiplier = 5,
  adaptive.p = 0.05,
  useNNLS = TRUE,
  useRaw = TRUE,
  refitEvery = 3,
```

```
  useSE = FALSE,
  var.prior = TRUE,
  Uscale = FALSE,
  robust.vp = TRUE,
  use_cpp = FALSE,
  clamp_k_method = c("elbow", "permutation", "gavish_donoho", "scaleSVs")
)
```

Arguments

Y	Gene expression matrix (genes x samples). Can be dense, sparse (dgCMatrix), or FBM.
priorMat	Binary or weighted prior matrix (genes x pathways) linking genes to pathways.
Chat	Ignored in this version (kept for interface compatibility).
svdres	Optional precomputed SVD result for initialization.
clamp.base.result	Optional result from CLAMPbase() providing initial values.
clamp_k	Number of latent variables for CLAMP (final model rank). If NULL, it is chosen automatically via select_clamp_k().
svd_k	Number of singular values/components to compute in the SVD. If NULL, defaults to max(2, min(n_genes, n_samples) - 1).
L1, L2	Regularization parameters for Z and B. Defaults use values from clamp.base.result.
cvn	Number of folds for pathway-level cross-validation. Default: 5.
max.iter	Maximum number of outer iterations. Default: 30.
trace	Logical; print iteration progress. Default: TRUE.
maxPath	Maximum number of pathways per LV during U-fitting. Default: 10.
doCrossval	Whether to mask prior entries for CV evaluation. Default: TRUE.
penalty.factor	Optional vector of per-pathway penalties for glmnet. Default: 1.
glm_alpha	Elastic net mixing parameter for U estimation. Default: 0.9.
minGenes	Minimum number of genes per pathway. Default: 0.
tol	Convergence tolerance for B updates. Default: 5e-4.
seed	Random seed for CV masking. Default: 123456.
allGenes	If TRUE, adds zero-filled rows for missing genes. Default: FALSE.
rseed	Reproducibility, coordinate descent updates are done in random order.
max.U.updates	Maximum number of U updates (capped by max.iter).
pathwaySelection	Pathway selection mode for U fitting ("fast" or "complete").
multiplier	Variance-prior scaling factor. Default: 5.
adaptive.p	Quantile of negative Z values used to define adaptive thresholding. Default: 0.05.
useNNLS	Whether to use non-negative least squares for U estimation. Default: TRUE.
useRaw	If TRUE, uses unthresholded Z in U updates. Default: TRUE.
refitEvery	Frequency (in U updates) of full refits. Default: 3.
useSE	Logical; whether to use the 1-standard-error rule for internal glmnet fitting. Default is FALSE.

var.prior	Logical; if TRUE, enables adaptive variance prior updates for Z. Default: TRUE.
Uscale	Logical; whether to scale U columns. Default: FALSE.
robust.vp	Logical; winsorize prior-predicted Z2 values to reduce outlier effects. Default: TRUE.
use_cpp	Logical; if TRUE, use C++ implementation for Z updates. Default is FALSE.
clamp_k_method	Method for selecting clamp_k when not provided. One of "elbow" (default), "permutation", "gavish_donoho", or "scaleSVs". Passed to select_clamp_k().

Details

Cross-validation can be used to evaluate pathway-LV specificity, and a variance-based prior (var.prior = TRUE) introduces adaptive shrinkage of Z based on how strongly each latent component aligns with prior pathways. The scaling factor multiplier (default 5) controls the strength of this adaptive shrinkage.

This implementation omits ridge-projected priors (Chat) and uses a lighter variance prior with a lower default multiplier = 5, allowing more flexible latent representations. Setting var.prior = FALSE reproduces standard CLAMP-like updates. Cross-validation, if enabled, masks 20% of gene-pathway associations per column to estimate pathway-LV specificity (reported via AUC and p-values).

Value

A list with elements:

B LV loadings on samples (k × samples)

Z Gene loadings (genes × k)

U Pathway loadings (pathways × k)

C Prior matrix used during training (masked if CV)

Z2 Predicted Z from pathway priors

heldOutGenes Held-out gene lists per pathway (CV mode)

Uauc, Up, summary CV evaluation metrics if CV is enabled

priorMat, priorMatCV Final and masked prior matrices

call Function call

Examples

```
set.seed(1)
mat <- matrix(rnorm(100), nrow = 10, ncol = 10)
base <- CLAMPbase(mat, clamp_k = 5, trace = FALSE, max.iter = 5)
prior <- matrix(sample(0:1, 10 * 6, TRUE, prob = c(0.9, 0.1)),
    nrow = 10, ncol = 6
)
fit <- CLAMPfull(
    Y = mat,
    priorMat = prior,
    clamp.base.result = base,
    doCrossval = FALSE,
    adaptive.p = 0,
    max.U.updates = 0,
    max.iter = 1,
    trace = FALSE
)
```

CLAMPfullnVP *Full CLAMP model with prior information and cross-validation*

Description

Runs the full CLAMP model using a gene expression matrix and prior pathway annotation matrix. This function performs latent variable decomposition guided by prior knowledge and includes optional cross-validation to evaluate pathway associations.

Usage

```
CLAMPfullnVP(
  Y,
  priorMat,
  svdres = NULL,
  clamp.base.result = NULL,
  clamp_k = NULL,
  svd_k = NULL,
  L1 = NULL,
  L2 = NULL,
  top = NULL,
  cvn = 5,
  max.iter = 350,
  trace = FALSE,
  Chat = NULL,
  maxPath = 10,
  doCrossval = TRUE,
  penalty.factor = rep(1, ncol(priorMat)),
  glm_alpha = 0.9,
  minGenes = 10,
  tol = 5e-04,
  seed = 123456,
  allGenes = FALSE,
  rseed = NULL,
  max.U.updates = 5,
  pathwaySelection = c("fast", "complete"),
  multiplier = 1,
  adaptive.p = 0.05,
  useNNLS = TRUE,
  useRaw = TRUE,
  refitAll = FALSE,
  useSE = FALSE,
  ncores = 1,
  clamp_k_method = c("elbow", "permutation", "gavish_donoho", "scaleSVs")
)
```

Arguments

Y	Gene expression matrix (genes x samples). Can be dense, sparse (dgCMatrix), or FBM.
priorMat	Binary matrix (genes x pathways) representing prior annotations.

svdres	Optional SVD result used for initialization.
clamp.base.result	Optional result from CLAMPbase() to initialize B.
clamp_k	Number of latent variables for CLAMP (final model rank). If NULL, it is chosen automatically via select_clamp_k().
svd_k	Number of singular values/components to compute in the SVD. If NULL, defaults to max(2, min(n_genes, n_samples) - 1).
L1	Regularization strength for Z. If NULL, initialized from SVD or clamp.base.result.
L2	Regularization strength for B. If NULL, initialized from SVD or clamp.base.result.
top	If set, keeps only top-n values per column in Z during U updates.
cvn	Number of folds for cross-validation in U updates. Default is 5.
max.iter	Maximum number of iterations. Default is 350.
trace	Logical; if TRUE, prints iteration progress.
Chat	Optional precomputed matrix for solving U.
maxPath	Maximum number of pathways/features selected per LV. Default is 10.
doCrossval	Whether to perform pathway-level cross-validation. Default is TRUE.
penalty.factor	Vector of feature-specific penalties for glmnet. Default: all ones.
glm_alpha	Elastic net mixing parameter for glmnet. Default is 0.9.
minGenes	Minimum number of genes per pathway to retain. Default is 10.
tol	Convergence tolerance on relative change in B. Default is 5e-4.
seed	Seed for reproducibility of cross-validation masking. Default is 123456.
allGenes	If TRUE, zero-fills priorMat for genes not present. Default is FALSE.
rseed	Optional seed for randomly reinitializing B and Z.
max.U.updates	Maximum number of U updates. Default is 5.
pathwaySelection	Pathway selection mode: "fast" or "complete".
multiplier	Scaling factor for adjusting L1 and L2.
adaptive.p	Quantile threshold for adaptively zeroing small Z values. After each update, the adaptive.p-quantile of negative entries in Z is used (flipped positive) to threshold small positive values, assuming they reflect noise. Default is 0.05.
useNNLS	If TRUE, uses non-negative least squares in U estimation. Default is TRUE.
useRaw	If TRUE, uses unthresholded Z for solving U. Default is TRUE.
refitAll	If TRUE, refits all U columns every update. Default is FALSE.
useSE	Logical; passed to the internal solveU() call. If TRUE, enables standard-error-aware selection when fitting U (pathway coefficients). Default is FALSE.
ncores	Number of cores to use for parallel computation (only used if Y is an FBM). Default is 1.
clamp_k_method	Method for selecting clamp_k when not provided. One of "elbow" (default), "permutation", "gavish_donoho", or "scaleSVs". Passed to select_clamp_k().

Details

The model alternates between solving Z, B, and U. Adaptive sparsity is applied to Z using a dynamic threshold based on the negative tail of its distribution. Cross-validation is used to hold out gene annotations in priorMat and evaluate latent variable specificity.

Value

A list with the following components:

`B` Latent variable loadings (LVs x genes)

`Z` Latent variable matrix (LVs x samples)

`U` Pathway loadings matrix (pathways x LVs)

`C` Masked prior matrix used for training

`L1, L2` Regularization parameters

`heldOutGenes` List of held-out genes per pathway (if CV is enabled)

`Uauc` AUC matrix from CV evaluation (if enabled)

`Up -log10(p)` values from CV evaluation (if enabled)

`summary` Data frame of AUC, p-values, and FDR per pathway x LV (if enabled)

`priorMatCV` Masked prior matrix used during CV

`priorMat` Final filtered prior matrix

`withPrior` Indices of LVs with non-zero pathway loadings

`call` Function call

Examples

```
mat <- matrix(rnorm(100), 10, 10)
svdres <- rsvd::rsvd(mat, k = 5)
base <- CLAMPbase(Y = mat, clamp_k = 5, svdres = svdres, trace = FALSE)
priorMat <- matrix(1, nrow(mat), 5)
full <- CLAMPfullnVP(
    Y = mat, priorMat = priorMat, svdres = svdres,
    clamp.base.result = base, clamp_k = 5,
    doCrossval = FALSE, trace = FALSE, max.U.updates = 0
)
```

CLAMPplotTopZ *Plot top genes per LV by Z loading*

Description

For each selected latent variable, ranks genes by their Z loading and plots the top genes as a loading-versus-rank scatter plot. The highest ranking genes are labelled with `ggrepel`.

Usage

```
CLAMPplotTopZ(
  clampRes,
  data = NULL,
  priorMat = NULL,
  top = 50,
  index = NULL,
  allLVs = FALSE,
  label.top = min(10, top),
  max.name.len = 50
)
```

Arguments

clampRes	A CLAMP result list containing at least Z, and optionally U for selecting LVs with pathway support.
data	Deprecated; retained for backward compatibility and ignored.
priorMat	Deprecated; retained for backward compatibility and ignored.
top	Number of top genes to plot per LV. Default 50.
index	Integer or character vector of LV columns to include. NULL keeps LVs with non-zero U entries when U is present.
allLVs	Logical; if TRUE, all LVs are eligible when index = NULL. Default FALSE.
label.top	Number of top genes to label per LV. Default min(10, top).
max.name.len	Maximum characters for displayed gene labels.

Value

A ggplot2::ggplot() object for one LV, or a patchwork object for multiple LVs.

Examples

```
set.seed(1)
genes <- paste0("Gene", 1:80)
lvs <- paste0("LV", 1:4)
paths <- paste0("Path", 1:10)
Z <- matrix(abs(rnorm(80 * 4)),
    nrow = 80,
    dimnames = list(genes, lvs)
)
U <- matrix(abs(rnorm(10 * 4)),
    nrow = 10,
    dimnames = list(paths, lvs)
)
clampRes <- list(Z = Z, U = U)
CLAMPplotTopZ(clampRes, top = 20, index = 1:2)
```

CLAMPplotU *Plot the U matrix (pathway-LV associations) as a heatmap*

Description

Displays the pathway loading matrix U after filtering by AUC and FDR thresholds. Only the top-top pathways per LV are shown.

Usage

```
CLAMPplotU(
  clampRes,
  auc.cutoff = 0.6,
  fdr.cutoff = 0.05,
  indexCol = NULL,
  indexRow = NULL,
  top = 3,
```

```
  sort.row = FALSE,
  cluster.rows = TRUE
)
```

Arguments

clampRes	A CLAMP result list containing at least U, Uauc, Up, and summary.
auc.cutoff	Minimum AUC threshold; entries below this are set to zero. Default 0.6.
fdr.cutoff	Maximum FDR threshold for pathway significance filtering. Default 0.05.
indexCol	Integer vector of LV column indices to include. NULL uses all LVs.
indexRow	Integer vector of pathway row indices to include. NULL uses all pathways.
top	Number of top pathways to retain per LV. Default 3.
sort.row	Logical; if TRUE, rows are sorted by the dominant LV. Default FALSE.
cluster.rows	Logical; if TRUE (default), rows are reordered by hierarchical clustering (overridden when sort.row = TRUE).

Value

Invisibly returns a ggplot2::ggplot() object.

Examples

```
set.seed(42)
pathways <- paste0("Path", 1:30)
lvs <- paste0("LV", 1:5)

U <- matrix(abs(rnorm(30 * 5)),
    nrow = 30,
    dimnames = list(pathways, lvs)
)
Uauc <- matrix(runif(30 * 5, 0.5, 1.0),
    nrow = 30,
    dimnames = list(pathways, lvs)
)
Up <- matrix(runif(30 * 5, 0, 3),
    nrow = 30,
    dimnames = list(pathways, lvs)
)

# Build a minimal summary table
nr <- 30 * 5
summ <- data.frame(
    pathway = rep(pathways, 5),
    LV = rep(lvs, each = 30),
    AUC = as.vector(Uauc),
    p_value = runif(nr, 0, 0.1),
    FDR = runif(nr, 0, 0.1),
    stringsAsFactors = FALSE
)

clampRes <- list(U = U, Uauc = Uauc, Up = Up, summary = summ)
CLAMPplotU(clampRes, auc.cutoff = 0.6, fdr.cutoff = 0.1, top = 3)
```

cleanFBM — *Clean a Filebacked Big Matrix (FBM) by log-transforming and handling NAs*

Description

This function inspects an FBM to determine if log-transformation is needed (based on value range) and whether NA values are present. If the maximum value is >= 100, it applies a log2(x + 1) transformation in-place. If any NA values are detected, they are replaced with 0.

Usage

```
cleanFBM(fbm, ncores = 1)
```

Arguments

fbm	A bigmemory::FBM or bigstatsr::FBM object.
ncores	Integer; number of cores to use for parallel operations (default 1).

Details

Modifies the FBM in place. Uses bigstatsr::big_apply() to process in parallel-safe chunks.

Value

A list with:

max_value The maximum value encountered in the FBM (after log transformation if applied).

had_na Logical indicating whether any NA values were found and filled.

Examples

```
fbm <- bigstatsr::FBM(3, 4, init = matrix(
    c(0, 1, 2, NA, 100, 200, 300, 400, 5, 6, 7, 8),
    nrow = 3
))
cleanFBM(fbm, ncores = 1)
```

commonRows — *Find common row names between two matrices or data frames*

Description

Returns the intersection of row names shared by two input objects.

Usage

```
commonRows(data1, data2)
```

Arguments

data1	A matrix, data frame, or similar object with row names.
data2	A matrix, data frame, or similar object with row names.

Value

A character vector of row names common to both inputs.

---

compareBs *Compare two sets of factor loadings or embeddings*

Description

Compares correspondence between two result matrices (e.g., factor loadings) across multiple targets using correlation, AUC, or t-statistics. Produces a paired comparison plot and summary statistics.

Usage

```
compareBs(
  res1,
  res2,
  target,
  method = c("p", "s", "a", "t"),
  xlab = "1",
  ylab = "2",
  stat.method = c("t", "wilcox"),
  oneToOne = TRUE
)
```

Arguments

res1	res2 Result objects or matrices containing factor loadings. If a list, must contain element B; if of class "rsvd", t(res$v) is used.
res2	Second result object or matrix to compare against.
target	Numeric matrix of target variables (samples × targets).
method	Character, one of "p", "s", "a", or "t", indicating Pearson, Spearman, AUC, or t-statistic comparison.
xlab	ylab Labels for x and y axes in the plot.
ylab	Label for y-axis (used in plot).
stat.method	Statistical test to compare correlations ("t" or "wilcox").
oneToOne	Logical, whether to apply one-to-one masking of associations.

Value

A list with:

plot A ggplot object comparing maximal correlations across targets.

df A data.frame with per-target metrics and (if available) top genes.

Examples

```
set.seed(123)
# Simulated example with 50 genes × 20 samples
Y <- matrix(rnorm(50 * 20), nrow = 50, ncol = 20)

# Run two CLAMP-like decompositions (here using simple SVD)
svd1 <- rsvd::rsvd(Y, k = 5)
svd2 <- rsvd::rsvd(Y + matrix(rnorm(50 * 20, 0, 0.1), 50, 20), k = 5)

# Define a target variable (e.g., binary or continuous trait)
target <- matrix(rnorm(20 * 3), ncol = 3)
colnames(target) <- c("Trait1", "Trait2", "Trait3")

# Compare the two sets of embeddings
res <- compareBs(svd1, svd2, target,
    method = "p", xlab = "SVD1", ylab = "SVD2"
)
```

---

computeRowStatsFBM *Compute row-wise sum and sum of squares for a Filebacked Big Matrix*

Description

Efficiently computes row sums and row sum-of-squares for a `bigstatsr::FBM` using column-wise chunking, suitable for large datasets that cannot be loaded fully into memory.

Usage

```
computeRowStatsFBM(fbm, ncores = 1)
```

Arguments

`fbm`	A `bigstatsr::FBM` object.
`ncores`	Integer; number of cores to use for parallel operations (default 1).

Value

A list with two numeric vectors:

row_sums Sum of each row.

row_sums_sq Sum of squares of each row.

compute_svd *Compute a truncated SVD for a CLAMP input matrix*

Description

By default, the SVD backend is auto-detected from the class of Y: bigstatsr::big_SVD for FBM objects, irlba::irlba for sparse dgCMatrix objects, and rsvd::rsvd otherwise. A specific backend can be forced via method. Used by the CLAMP solvers so that the SVD step is handled in one place.

Usage

```
compute_svd(Y, k = NULL, method = NULL)
```

Arguments

Y	A matrix-like object (dense matrix, dgCMatrix, or FBM).
k	Integer number of components to compute. If NULL (the default), select_svd_k(Y) is used.
method	One of "rsvd", "irlba", or "big_SVD". If NULL (the default), the backend is auto-detected from the class of Y.

Value

A list with d, u, v components (structure depends on the backend but these three fields are always present).

Examples

```
set.seed(1)
Y <- matrix(rnorm(100), nrow = 20, ncol = 5)
res <- compute_svd(Y, k = 3)
length(res$d)

# Force a specific backend:
res2 <- compute_svd(Y, k = 3, method = "rsvd")
```

cpmCLAMP *Compute counts-per-million (CPM) for CLAMP pipelines*

Description

Compute counts-per-million (CPM) for CLAMP pipelines

Usage

```
cpmCLAMP(counts)
```

Arguments

counts	A numeric matrix or data.frame of raw counts (genes x samples).

Value

A numeric matrix of CPM values (same dimensions), ready for CLAMP input.

Examples

```
mat <- matrix(seq_len(12), nrow = 3)
cpmCLAMP(mat)
```

---

cpmCLAMPFBM *Compute CPM on a file-backed matrix for CLAMP (in-place)*

Description

Compute CPM on a file-backed matrix for CLAMP (in-place)

Usage

```
cpmCLAMPFBM(fbm_counts, block_size = 1000, ncores = 1)
```

Arguments

fbm_counts	A bigstatsr::FBM of raw counts (genes x samples).
block_size	Integer; columns per block (default 1000).
ncores	Integer; number of cores to use for parallel operations (default 1).

Value

Invisibly returns the modified FBM (now holding CPM values).

Examples

```
library(bigstatsr)
mat <- matrix(c(10, 20, 30, 40, 50, 60),
    nrow = 2,
    dimnames = list(c("gene1", "gene2"), paste0("sample", seq_len(3)))
)
fbm <- FBM(nrow(mat), ncol(mat), init = mat)
cpmCLAMPFBM(fbm, block_size = 1)
```

crossVal *Cross-validation AUC for CLAMP latent variables and pathways*

Description

Evaluates how well each latent variable in a CLAMP model captures held-out pathway annotations, using cross-validation over the prior matrix. For each latent variable and associated pathway, held-out genes are selected and the AUC is computed using their scores in `clampRes$Z`.

Usage

```
crossVal(clampRes, priorMat, priorMatcv)
```

Arguments

`clampRes`	A list containing `U` (loadings) and `Z` (scores) from a CLAMP model.
`priorMat`	A binary matrix (genes x pathways) indicating original pathway annotations.
`priorMatcv`	A version of `priorMat` used to mask held-out annotations for cross-validation.

Value

A list with:

`Uauc` Matrix of AUC values (pathways x LVs)

`Upval` Matrix of `-log10(p)` values (pathways x LVs)

`summary` Data frame with pathway, LV index, AUC, p-value, and FDR

cross_ZY *Cross-product Z^T Y with FBM or dense matrices*

Description

Computes $Z^T Y$, handling FBM objects from bigstatsr as well as base R matrices.

Usage

```
cross_ZY(Y, Z)
```

Arguments

`Y`	Gene expression matrix (genes x samples), dense or FBM.
`Z`	Latent variable matrix (genes x k).

Value

A numeric matrix giving Z^T Y.

Examples

```
set.seed(123)

genes <- 40
samples <- 10
k <- 4

Y <- matrix(rnorm(genes * samples), nrow = genes)
Z <- matrix(rnorm(genes * k), nrow = genes)

# Compute Z^T Y
res1 <- cross_ZY(Y, Z)
dim(res1) # k × samples
```

---

dataWholeBlood *Whole-blood reference expression matrix*

Description

A numeric matrix of whole-blood gene expression where rows correspond to genes and columns to samples.

Usage

```
data(dataWholeBlood)
```

Format

A numeric matrix with G genes (rows) and N samples (columns). Row names are gene symbols, and column names are sample IDs.

Details

This object is a whole-blood RNA-seq expression matrix. The source data are publicly available from NCBI GEO under accession GSE130824 (Homo sapiens whole-blood RNA-seq, 36 samples). Raw sequencing data are deposited in SRA, and the processed normalized expression file is provided as `GSE130824_dataNormedFiltered.txt.gz`. The matrix bundled with CLAMP is derived from that processed file and serves as a compact example dataset for package demonstrations and unit tests.

Value

A numeric matrix of expression values.

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE130824

Examples

```
data(dataWholeBlood)
```

`differentialLVActivity` *Differential latent-variable activity between sample groups*

Description

For each row of a CLAMP `B` matrix, compares mean activity in a reference sample group against all other samples using a Wilcoxon rank-sum test, with Benjamini–Hochberg FDR adjustment.

Usage

```
differentialLVActivity(
  x,
  metadata,
  sample_col = "id",
  group_col = "type",
  reference
)
```

Arguments

`x`	A CLAMP result list (with element `B`) or a numeric matrix with latent variables in rows and samples in columns.
`metadata`	Data frame with sample identifiers and group labels.
`sample_col`	Name of the column in `metadata` holding sample identifiers matching `colnames(B)`.
`group_col`	Name of the column in `metadata` holding group labels.
`reference`	Label of the reference group. All other samples are treated as the comparison group.

Value

A data frame with one row per latent variable, ordered by FDR. Columns: `LV`, `Mean_Reference`, `Mean_Comparison`, `Mean_Diff`, `P_Value`, `FDR`.

Examples

```
B <- matrix(rnorm(30), nrow = 3)
rownames(B) <- paste0("LV", seq_len(3))
colnames(B) <- paste0("S", seq_len(10))
meta <- data.frame(
    id = colnames(B),
    type = rep(c("Control", "Case"), each = 5)
)
differentialLVActivity(B, meta, reference = "Control")
```

`filterFBM` *Filter rows of a Filebacked Big Matrix based on mean and variance*

Description

Filters an FBM based on row-level mean and variance thresholds, returning a new FBM with only the selected rows.

Usage

```
filterFBM(
  fbm,
  rowStats,
  keep_samples_idx = NULL,
  mean_cutoff = NULL,
  var_cutoff = NULL,
  backingfile = "filtered_fbm"
)
```

Arguments

`fbm`	A `bigstatsr::FBM` object.
`rowStats`	A list with numeric vectors `row_means` and `row_variances`.
`keep_samples_idx`	Optional integer vector of column indices to retain. Default is `filtered_fbm`.
`mean_cutoff`	Optional minimum mean threshold; rows with means below this are removed.
`var_cutoff`	Optional minimum variance threshold; rows with variances below this are removed.
`backingfile`	A character string specifying the filename (without extension) for the new FBM.

Details

This function creates a new FBM and copies over only the rows that pass the filtering criteria. The original FBM is unchanged.

Value

A list with:

fbm_filtered A new FBM object containing only filtered rows.

kept_rows Indices of rows retained in the filtering step.

Examples

```
fbm <- bigstatsr::FBM(5, 3, init = matrix(rnorm(15), nrow = 5))
rs <- list(
    row_means = rowMeans(fbm[]),
    row_variances = apply(fbm[], 1, var)
)
out <- filterFBM(fbm, rs,
    mean_cutoff = -0.2, var_cutoff = 0.5,
```

```
    backingfile = tempfile()
)
```

`findSplineMax` *Find the location of the maximum of a smoothing spline*

Description

Find the location of the maximum of a smoothing spline

Usage

```
findSplineMax(x, y, n = 1000, spar = NULL)
```

Arguments

`x`	Numeric vector of x values.
`y`	Numeric vector of y values (same length as `x`).
`n`	Integer, number of grid points to evaluate (default 1000).
`spar`	Smoothing parameter passed to `stats::smooth.spline`.

Value

A named list with components:

x The x coordinate at which the spline reaches its maximum.

y The corresponding maximum fitted y value.

Examples

```
x <- seq_len(10)
y <- sin(x) + rnorm(10, 0, 0.1)
findSplineMax(x, y)
```

`getAUCstats` *Count number of latent variables exceeding AUC thresholds*

Description

Given a summary data frame from cross-validation, reports the number of latent variables with maximum AUC exceeding 0.7, 0.8, and 0.9.

Usage

```
getAUCstats(summary)
```

Arguments

summary	A data frame with LV index and AUC columns.

Value

A named numeric vector with counts for thresholds 0.7, 0.8, and 0.9.

---

getChat *Compute Chat matrix from prior annotation*

Description

Computes the transformation matrix Chat used to map from observed data to latent space, based on a pseudo-inverse of the prior annotation matrix. Optionally standardizes the columns of priorMat before computing.

Usage

```
getChat(priorMat, scale = TRUE)
```

Arguments

priorMat	A numeric or sparse matrix (features x pathways) containing prior annotations.
scale	Logical; if TRUE (default), standardizes the columns of priorMat before computing Chat.

Value

A numeric matrix Chat of dimensions (pathways x features).

Examples

```
# simple toy prior: 3 features x 2 pathways
priorMat <- matrix(
    c(
        1, 0, 1,
        0, 1, 0
    ),
    nrow = 3, ncol = 2,
    dimnames = list(
        paste0("gene", seq_len(3)),
        paste0("path", seq_len(2))
    )
)
# compute Chat (2 pathways x 3 features)
Chat <- getChat(priorMat)
```

getGMT *Download and read a GMT file from a URL*

Description

Downloads a Gene Matrix Transposed (GMT) file from a specified URL, reads it into R as a list, and removes the temporary file afterward.

Usage

```
getGMT(url, name = NULL, cache_dir = NULL, redownload = FALSE)
```

Arguments

url	A character string specifying the URL to a GMT file.
name	Optional name for the GMT file (defaults to the portion after the last '=' in the URL).
cache_dir	Optional directory in which to cache/download the GMT file.
redownload	Logical; if TRUE, forces re-download even if cached.

Value

A named list where each element is a character vector of gene names for a given gene set.

Examples

```
url <- paste0(
    "https://maayanlab.cloud/Enrichr/geneSetLibrary",
    "?mode=text&libraryName=KEGG_2019_Human"
)
gmt_list <- getGMT(url)
# list available gene sets
names(gmt_list)
# inspect the first few genes in the first gene set
head(gmt_list[[1]])
```

getMatchedPathwayMat *Subset and filter pathway matrix to match target genes*

Description

Filters a gene-by-pathway annotation matrix to retain only pathways with sufficient overlap with a given gene set. The result is a sparse matrix aligned to new.genes, with columns (pathways) retained only if they have at least min.genes matched genes.

Usage

```
getMatchedPathwayMat(pathMat, new.genes, min.genes = 10)
```

Arguments

pathMat	A sparse binary matrix of genes (rows) x pathways (columns).
new.genes	Character vector of gene names to match.
min.genes	Minimum number of overlapping genes required to keep a pathway.

Value

A sparse matrix of dimensions length(new.genes) x filtered pathways.

Examples

```
library(Matrix)
# create a toy gene-by-pathway sparse matrix
genes <- paste0("g", seq_len(6))
pathways <- c("Path1", "Path2", "Path3")
# Path1: g1, g2; Path2: g2, g3, g4; Path3: g5
pathMat <- sparseMatrix(
    i = c(1, 2, 2, 3, 4, 5),
    j = c(1, 1, 2, 2, 2, 3),
    dims = c(length(genes), length(pathways)),
    dimnames = list(genes, pathways)
)
new.genes <- genes
filtered <- getMatchedPathwayMat(pathMat, new.genes, min.genes = 2)
```

---

getMatchedPathwayMat2 *Subset and filter multiple pathway matrices to match target genes*

Description

Filters gene-by-pathway annotation matrices to retain only pathways with sufficient overlap with a given gene set. The result is a sparse matrix aligned to new.genes, combining all inputs column-wise.

Usage

```
getMatchedPathwayMat2(..., new.genes, min.genes = 10)
```

Arguments

...	One or more sparse binary matrices (genes x pathways).
new.genes	Character vector of gene names to match.
min.genes	Minimum number of overlapping genes required to keep a pathway.

Value

A sparse matrix with rows = new.genes and columns = filtered pathways from all inputs.

getMatchedPathwayMatList *Subset and filter multiple pathway matrices to match target genes*

Description

Filters one or more gene-by-pathway annotation matrices to retain only pathways with sufficient overlap with a given target gene set. Each input matrix is restricted to `new.genes`, and pathways with fewer than `min.genes` overlapping genes are removed. The resulting matrices are column-bound into a single sparse matrix aligned to `new.genes`.

Usage

```
getMatchedPathwayMatList(..., new.genes, min.genes = 10)
```

Arguments

`...`	One or more binary matrices (genes x pathways), either base `matrix` or sparse `Matrix` objects. Row names must be gene identifiers.
`new.genes`	Character vector of gene names to align all pathway matrices to.
`min.genes`	Integer; minimum number of overlapping genes required for a pathway to be retained. Default is 10.

Value

A sparse binary matrix with rows equal to `new.genes` and columns equal to the union of filtered pathways from all input matrices.

Examples

```
set.seed(123)
library(Matrix)

# Simulate two small pathway matrices (genes × pathways)
genes <- paste0("Gene", 1:100)
pathways1 <- paste0("Path", 1:5)
pathways2 <- paste0("Path", 6:10)

mat1 <- matrix(sample(c(0, 1), 100 * 5, replace = TRUE, prob = c(0.9, 0.1)),
    nrow = 100, ncol = 5,
    dimnames = list(genes, pathways1)
)
mat2 <- matrix(sample(c(0, 1), 100 * 5, replace = TRUE, prob = c(0.9, 0.1)),
    nrow = 100, ncol = 5,
    dimnames = list(genes, pathways2)
)

# Define target genes (subset of total)
new.genes <- sample(genes, 50)

# Match and filter pathways with at least 5 genes
matched <- getMatchedPathwayMatList(mat1, mat2,
```

```
    new.genes = new.genes, min.genes = 5
)
```

`getMatchedPathwayMatOld` *Subset and filter pathway matrix to match target genes*

Description

Filters a gene-by-pathway annotation matrix to retain only pathways with sufficient overlap with a given gene set. The result is a sparse matrix aligned to `new.genes`, with columns (pathways) retained only if they have at least `min.genes` matched genes.

Usage

```
getMatchedPathwayMatOld(pathMat, new.genes, min.genes = 10)
```

Arguments

`pathMat`	A sparse binary matrix of genes (rows) x pathways (columns).
`new.genes`	Character vector of gene names to match.
`min.genes`	Minimum number of overlapping genes required to keep a pathway.

Value

A sparse matrix of dimensions `length(new.genes)` x filtered pathways.

`getMaxAUC` *Get maximum AUC per latent variable*

Description

Summarizes a cross-validation results data frame to extract the highest AUC value associated with each latent variable (LV).

Usage

```
getMaxAUC(summary, verbose = FALSE)
```

Arguments

`summary`	A data frame (e.g., from `crossVal()`) .
`verbose`	Logical; if `TRUE`, prints counts of LVs exceeding AUC thresholds. Default is `FALSE`.

Value

A data frame with columns LV index and max_AUC.

getScaleFromSVs *Estimate noise scale from singular values with linear tail extrapolation*

Description

This function estimates a characteristic scale from a vector of singular values by fitting a linear model to the tail and extrapolating to length n. The median of the extrapolated values is returned as the estimate. If no sufficiently linear tail is detected (based on min_r2) or the extrapolation produces negative values, a fallback estimate is returned using the 75\

Usage

```
getScaleFromSVs(sv, n, min_r2 = 0.95)
```

Arguments

sv	Numeric vector of singular values sorted in decreasing order.
n	Integer, total length to which the linear tail is extrapolated.
min_r2	Numeric, minimum R-squared value required for accepting the linear tail fit. Defaults to 0.95.

Value

A numeric scalar giving the estimated scale. If linear extrapolation is unreliable, returns sv[ceiling(0.75 * length(sv))].

Examples

```
sv <- exp(-seq(0, 5, length.out = 50)) + rnorm(50, 0, 0.01)
getScaleFromSVs(sv, n = 100)
```

gmtListToSparseMat *Convert a list of GMT gene sets to a sparse matrix*

Description

Converts a list of named gene sets (e.g., from getGMT()) into a sparse binary matrix where rows are genes, columns are gene sets, and entries are 1 if the gene is in the set.

Usage

```
gmtListToSparseMat(gmtList)
```

Arguments

gmtList	A nested list of gene sets. Outer names are gene set names; each entry is a character vector of gene names.

Value

A sparse binary matrix with genes as rows and gene sets as columns.

Examples

```
# define a simple nested GMT list
gmt1 <- list(
    PathwayA = c("Gene1", "Gene2", "Gene3"),
    PathwayB = c("Gene2", "Gene4")
)
gmt2 <- list(
    PathwayC = c("Gene1", "Gene4"),
    PathwayD = c("Gene3", "Gene5")
)
# combine into a nested list
nestedList <- list(gmt1 = gmt1, gmt2 = gmt2)
# convert to sparse matrix
sparseMat <- gmtListToSparseMat(nestedList)
```

---

majorCellTypes *Major cell-type annotations*

Description

A factor vector annotating samples by major cell type. This dataset provides cell-type labels corresponding to each sample in the reference expression matrix.

Usage

```
data(majorCellTypes)
```

Format

A factor (or character) vector of length N, where each element corresponds to a sample and indicates its major cell type.

Value

A factor (or character) vector of major cell-type labels.

Examples

```
data(majorCellTypes)
```

mat_mult *Matrix multiplication with support for FBM objects*

Description

Multiplies two matrices, using optimized multiplication if the first is a Filebacked Big Matrix (FBM).

Usage

```
mat_mult(mat1, mat2, ncores = 1)
```

Arguments

mat1	A matrix or an object of class FBM.
mat2	A numeric matrix.
ncores	Number of cores to use for parallel computation (only used if mat1 is an FBM). Default is 1.

Value

Matrix product of mat1 and mat2.

Examples

```
set.seed(123)
mat1 <- matrix(rnorm(20), nrow = 4)
mat2 <- matrix(rnorm(15), nrow = 5)
res1 <- mat_mult(mat1, mat2)
res1
```

max_correspondence_greedy *Greedy maximum correspondence from correlation matrix*

Description

Finds a one-to-one assignment (permutation) between rows and columns of a square correlation matrix that maximizes the total correlation score, using a greedy algorithm.

Usage

```
max_correspondence_greedy(cor_mat)
```

Arguments

cor_mat	A square numeric matrix of pairwise correlations (rows = items, cols = items).

Value

A vector of assignments (integer indices).

mymessage *Print a concatenated message*

Description

Wrapper around `message()` that pastes arguments together into a single string.

Usage

```
mymessage(...)
```

Arguments

`...`	Character strings to concatenate and print.

Value

Invisibly returns NULL. Called for side effects (messages).

num.pc *Estimate number of principal components via elbow or permutation method*

Description

Estimate number of principal components via elbow or permutation method

Usage

```
num.pc(data, method = c("elbow", "permutation"), B = 20, seed = NULL)
```

Arguments

`data`	Either a matrix (e.g. z-scored data) or an SVD result (list with $d).
`method`	One of "elbow" (fast) or "permutation" (slower).
`B`	Number of permutations (for method = "permutation").
`seed`	Seed for reproducibility.

Value

`Estimated number of PCs.`

Examples

```
# generate a small random matrix: 5 features x 10 samples
mat <- matrix(rnorm(5 * 10), nrow = 5)
# fast elbow estimate
num.pc(mat, method = "elbow")
# slower permutation estimate (use fewer perms for example speed)
num.pc(mat, method = "permutation", B = 5)
```

oneToOneMask *One-to-one masking of maximum associations*

Description

Selects the highest-scoring one-to-one pairs between rows and columns of a matrix, similar to a greedy bipartite matching. All other entries are set to a sentinel value (default `-100`).

Usage

```
oneToOneMask(cc)
```

Arguments

cc	A numeric matrix of association scores.

Value

A numeric matrix of the same dimensions as `cc`, where only the selected one-to-one maxima are retained and all other entries are set to `-100`.

Examples

```
set.seed(1)
m <- matrix(runif(16), 4, 4)
oneToOneMask(m)
```

panDB *panDB gene-set database*

Description

A list of curated pathway and biological process gene sets used as prior knowledge for CLAMP and other latent-variable models.

Usage

```
data(panDB)
```

Format

A named list of length M, where each element is a gene-set collection.

Details

Each element corresponds to a functional collection (e.g., KEGG, Reactome, GO), where each entry contains a character vector of gene symbols.

Value

A list of gene-set collections used as priors for pathway-informed modeling.

Examples

```
data(panDB)
```

`pinv.ridge` *Ridge-regularized pseudoinverse via SVD*

Description

Computes a stable pseudoinverse of a symmetric positive semi-definite matrix using singular value decomposition (SVD) and ridge regularization. This is useful when the matrix is ill-conditioned or rank-deficient.

Usage

```
pinv.ridge(m, alpha = 0)
```

Arguments

`m`	A symmetric numeric matrix (e.g., from `crossprod()`).
`alpha`	Non-negative scalar specifying the ridge penalty. A small positive value stabilizes the inversion by shrinking large singular values.

Value

A numeric matrix representing the ridge-regularized pseudoinverse of `m`.

`plotTopZ_Complex` *ComplexHeatmap visualization of top genes by latent variable*

Description

Produces a combined heatmap showing expression, pathway membership, and optionally Z-loadings for top genes per LV using ComplexHeatmap.

Usage

```
plotTopZ_Complex(
  clampRes,
  data,
  priorMat,
  top = 10,
  top.pathway = 5,
  index = NULL,
  allLVs = FALSE,
  Zheat = FALSE,
```

```
  LV.names = NULL,
  max.genes = 100,
  max.col = 50,
  seed = 1234
)
```

Arguments

clampRes	A CLAMP result list containing matrices Z, U, etc.
data	Expression matrix with genes as rows.
priorMat	Binary gene × pathway matrix.
top	Integer, number of top genes per LV.
top.pathway	Integer, number of top pathways per LV to annotate.
index	Optional vector of LVs to include.
allLVs	Logical; include all LVs.
Zheat	Logical; whether to include the Z matrix as an additional heatmap.
LV.names	Optional character vector for LV names.
max.genes	Maximum number of genes allowed in the plot.
max.col	Maximum number of columns (samples).
seed	Random seed for column subsampling.

Value

Invisibly returns the drawn ComplexHeatmap object.

Examples

```
library(ComplexHeatmap)
library(Matrix)

# Simulate small CLAMP-like results
set.seed(123)
genes <- paste0("Gene", 1:100)
samples <- paste0("S", 1:20)
lvs <- paste0("LV", 1:3)

# Simulated Z (gene loadings) and U (pathway loadings)
Z <- matrix(rnorm(100 * 3), nrow = 100, dimnames = list(genes, lvs))
U <- matrix(abs(rnorm(50 * 3)),
    nrow = 50,
    dimnames = list(paste0("Path", 1:50), lvs)
)

# Expression data
expr_data <- matrix(rnorm(100 * 20),
    nrow = 100,
    dimnames = list(genes, samples)
)

# Binary gene × pathway matrix
priorMat <- matrix(sample(0:1, 100 * 50, replace = TRUE, prob = c(0.9, 0.1)),
    nrow = 100, ncol = 50,
```

```
    dimnames = list(genes, paste0("Path", 1:50))
)

# Create a CLAMP-like result list
clampRes <- list(Z = Z, U = U)

# Plot top genes and pathway memberships
plotTopZ_Complex(clampRes, expr_data, priorMat,
    top = 5, top.pathway = 3, index = 1:2, Zheat = TRUE
)
```

`preprocessCLAMP` *Preprocess an expression matrix for CLAMP*

Description

Filters genes by mean expression and variance, returning the filtered matrix and per-gene statistics.

Usage

```
preprocessCLAMP(Y, mean_cutoff = 0, var_cutoff = 0)
```

Arguments

`Y`	Numeric matrix of gene expression (rows = genes, cols = samples)
`mean_cutoff`	Numeric. Minimum row-mean required to keep a gene (default 0).
`var_cutoff`	Numeric. Minimum row-variance required to keep a gene (default 0).

Value

A list with components:

- Y_filtered: filtered matrix (genes x samples)
- rowStats: data.frame with columns mean and variance for each kept gene
- kept_rows: integer vector of the original row indices that were kept

Examples

```
# construct a small example matrix
mat <- matrix(
    c(
        1, 5, 10,
        2, 6, 11,
        3, 7, 12,
        4, 8, 13
    ),
    nrow = 4, byrow = FALSE,
    dimnames = list(paste0("gene", seq_len(4)), paste0("sample", seq_len(3)))
)

# keep genes with mean >= 6 and variance >= 2
res <- preprocessCLAMP(mat, mean_cutoff = 6, var_cutoff = 2)
```

`preprocessCLAMPFBM` *Preprocess a bigstatsr FBM for CLAMP*

Description

Makes a writable copy of the input FBM, cleans it (log2 transform if needed, fill NAs), filters rows by mean/variance, and returns the filtered FBM plus stats and indices.

Usage

```
preprocessCLAMPFBM(
  fbm,
  mean_cutoff = NULL,
  var_cutoff = NULL,
  backingfile = NULL,
  block_size = 1000,
  ncores = 1
)
```

Arguments

`fbm`	A bigstatsr::FBM (genes x samples), possibly read-only.
`mean_cutoff`	Numeric or NULL. Minimum row mean to keep (NULL = no mean filter).
`var_cutoff`	Numeric or NULL. Minimum row variance to keep (NULL = no var filter).
`backingfile`	Character or NULL. Base name for the *copy* FBM and filtered FBM on disk. If NULL, defaults to paste0(fbm$backingfile, '_preproc') and '_filtered'.
`block_size`	Number of rows to process at a time when copying data. Default is 1000.
`ncores`	Integer; number of cores to use for parallel operations (default 1).

Value

A list with:

`fbm_filtered`	The filtered FBM (writable).
`rowStats`	List with row_means & row_variances for fbm_filtered.
`kept_rows`	Integer vector of original row indices that were retained.

Examples

```
library(bigstatsr)
# create a toy matrix and back it with an FBM
mat <- matrix(
    c(
        1, 2, 3, # geneA
        10, 20, 30, # geneB
        100, 200, 300 # geneC
    ),
    nrow = 3, byrow = TRUE,
    dimnames = list(c("geneA", "geneB", "geneC"), paste0("s", seq_len(3)))
)
```

```
fbm <- FBM(nrow(mat), ncol(mat), init = mat)

# preprocess without filtering (all genes kept)
res_all <- preprocessCLAMPFBM(fbm)
```

`projectCLAMP` *Project new data into CLAMP latent space*

Description

Computes the latent loadings `B` for new gene expression data using the latent variables `Z` from a fitted CLAMP model. This allows transfer of the learned latent structure to new datasets with matched genes.

Usage

```
projectCLAMP(
  CLAMPres,
  newdata,
  scale = 1,
  ncores = 1,
  align = TRUE,
  verbose = TRUE
)
```

Arguments

`CLAMPres`	A result object from `CLAMPfull()` or `CLAMPbase()`, containing at least `Z` and `L2`.
`newdata`	A gene expression matrix (genes x samples) to be projected. Can be a standard matrix, sparse matrix, or FBM/big.matrix.
`scale`	Optional numeric multiplier for the L2 regularization terms. Default is 1.
`ncores`	Number of cores to use for parallel computation (only used if newdata is an FBM). Default is 1.
`align`	Logical; if `TRUE` (default), row names shared by `CLAMPres$Z` and `newdata` are used to align both matrices to the same genes in the same order before projection. If row names are not available, dimensions must already match.
`verbose`	Logical; if `TRUE` (default), report how many common rows are used for projection.

Details

This function uses ridge-regularized least squares to compute `B = solve(Z'Z + L2 * I) * Z'Y`, where `Z` is the latent matrix from the trained CLAMP model and `Y` is the new dataset. If `newdata` is a File-backed Big Matrix (FBM) and does not need row-name subsetting, the computation is optimized using `bigstatsr::big_cprodMat()`.

Value

A matrix `B` of projected latent loadings (LVs x samples) for the new dataset.

Examples

```
# fit a tiny CLAMP model for projection
Y0 <- matrix(rnorm(5 * 3),
    nrow = 5,
    dimnames = list(paste0("Gene", 1:5), paste0("S", 1:3))
)
base <- CLAMPbase(Y0, clamp_k = 2, max.iter = 1, trace = FALSE)
# new data can be provided in a different row order
newY <- matrix(rnorm(5 * 2),
    nrow = 5,
    dimnames = list(rev(rownames(Y0)), paste0("N", 1:2))
)
projB <- projectCLAMP(base, newdata = newY)
# check dimensions: 2 latent vars x 2 samples
dim(projB)
```

---

read_gmt *Read a GMT file into a list*

Description

Parses a local GMT file and returns a list of gene sets. Each gene set is represented as a character vector of unique gene names.

Usage

```
read_gmt(filename)
```

Arguments

`filename`	A `character(1)` string giving the path to a .gmt file.

Value

A `list` where each element is a character vector of gene names, named by the gene set ID.

Examples

```
# Bioconductor requires runnable examples.
# We create a dummy GMT file for this example:
gmt_file <- tempfile(fileext = ".gmt")
writeLines(
    c(
        "PATHWAY_A\thttp://link.com\tGENE1\tGENE2\tGENE3",
        "PATHWAY_B\thttp://link.com\tGENE2\tGENE4"
    ),
    con = gmt_file
)

# Run the function
gs <- read_gmt(gmt_file)
```

```
# Inspect results
length(gs)
names(gs)
gs[["PATHWAY_A"]]
```

`ridge_B` *Ridge regression update for B*

Description

Solves $B = (Z^T Z + L2)^{-1} Z^T Y$ with ridge penalty matrix L2.

Usage

```
ridge_B(Y, Z, L2k)
```

Arguments

`Y`	Gene expression matrix (genes x samples), dense or FBM.
`Z`	Latent variable matrix (genes x k).
`L2k`	Ridge penalty matrix (k x k).

Value

A numeric matrix of size k x samples.

Examples

```
set.seed(123)

genes <- paste0("Gene", 1:50)
samples <- paste0("S", 1:20)
k <- 5

Y <- matrix(rnorm(50 * 20), nrow = 50, dimnames = list(genes, samples))
Z <- matrix(rnorm(50 * k),
    nrow = 50,
    dimnames = list(genes, paste0("LV", 1:k))
)

lambda <- 0.1
L2k <- diag(lambda, k)

# Solve for B = (Z'Z + L2)^(-1) Z'Y
B <- ridge_B(Y, Z, L2k)
```

rotateSVD *Rotate SVD components to make dominant directions positive*

Description

Ensures consistency in SVD output by flipping signs so that each left singular vector has a majority of positive entries.

Usage

```
rotateSVD(svdres)
```

Arguments

svdres	A list as returned by svd(), with components u, d, and v.

Value

A modified svd-result list where each column of $u has been sign-flipped so that its entries sum to a nonnegative value; $v is flipped correspondingly.

row_cor *Row-wise correlation between two matrices*

Description

Computes the Pearson correlation for each row between matrices A and B.

Usage

```
row_cor(A, B)
```

Arguments

A	A numeric matrix.
B	A numeric matrix of the same dimensions as A.

Value

A numeric vector of correlations, one per row.

run_elbow *Run elbow method to estimate number of PCs*

Description

Run elbow method to estimate number of PCs

Usage

```
run_elbow(d)
```

Arguments

d	Vector of singular values

Value

Estimated number of PCs via elbow

run_permutation *Run permutation method to estimate number of PCs*

Description

Run permutation method to estimate number of PCs

Usage

```
run_permutation(data, d, B = 20)
```

Arguments

data	Raw data matrix (row-normalized)
d	Vector of singular values
B	Number of permutations

Value

Estimated number of PCs via permutation test

`select_clamp_k` *Select default number of CLAMP latent variables from an SVD*

Description

Chooses the default `clamp_k` used by the CLAMP solvers when the user does not provide one, and returns the scale used for downstream L1/L2 regularization. Multiple methods are available via `method`:

Usage

```
select_clamp_k(
  svdres,
  n_samples,
  svd_k,
  method = c("elbow", "permutation", "gavish_donoho", "scaleSVs"),
  data = NULL,
  B = 20
)
```

Arguments

`svdres`	An SVD result with a `d` component (output of `compute_svd`).
`n_samples`	Integer number of samples in the original matrix (i.e. `ncol(Y)`). Used by `"scaleSVs"` and `"gavish_donoho"`.
`svd_k`	Integer upper bound on `clamp_k` (number of components actually computed in the SVD).
`method`	One of `"elbow"`, `"permutation"`, `"gavish_donoho"`, `"scaleSVs"`. Defaults to `"elbow"`.
`data`	Raw data matrix. Required for `"permutation"` (row-normalized internally) and `"gavish_donoho"` (used for `n_genes`).
`B`	Number of permutations for `"permutation"`.

Details

`"elbow"` **(default)** Elbow heuristic on the singular-value spectrum via `num.pc(svdres, method = "elbow")`. `scale = svdres$d[clamp_k]`.

`"permutation"` Permutation test via `num.pc(data, method = "permutation", B = B)`. Requires the raw row-normalized `data` matrix. `scale = svdres$d[clamp_k]`.

`"gavish_donoho"` Gavish-Donoho optimal singular-value threshold via `PCAtools::chooseGavishDonoho()`. Requires the raw `data` matrix (used for `n_genes`). `scale = svdres$d[clamp_k]`.

`"scaleSVs"` Previous behavior: `getScaleFromSVs()` linear-tail fit, `clamp_k <- min(floor(k * 1.5), svd_k)`, scale from the fit.

Value

An integer: the selected number of latent variables.

Examples

```
set.seed(1)
Y <- matrix(rnorm(100 * 30), nrow = 100, ncol = 30)
svdres <- compute_svd(Y, k = 25)
select_clamp_k(svdres, n_samples = ncol(Y), svd_k = 25)
```

`select_svd_k` *Select default number of components for a CLAMP solver SVD*

Description

Returns the default number of components to compute in a truncated SVD for the given input matrix. Used by the CLAMP solvers when `svd_k` is not provided explicitly. Other SVD contexts use their own heuristics.

Usage

```
select_svd_k(Y)
```

Arguments

`Y`	A matrix-like object (dense matrix, `dgCMatrix`, or `FBM`).

Value

An integer: `max(2, floor((min(nrow(Y), ncol(Y)) - 1) / 4))`.

Examples

```
select_svd_k(matrix(0, nrow = 100, ncol = 20))
```

`solveU` *Fit the loading matrix Z using sparse regression of prior information U*

Description

For each column of a target matrix `Z` using pathway or prior annotation `priorMat`. It performs regularization selection using cross-validation and can apply either Supports continuous or binary response models. Relaxed refitting is supported for final coefficient estimation.

Usage

```
solveU(
  Z,
  Chat = NULL,
  priorMat,
  penalty.factor,
  pathwaySelection = c("fast", "complete"),
  alpha = 0.9,
  maxPath = 10,
  nfolds = 5,
  useSE = FALSE,
  top = NULL,
  binary = FALSE,
  nlambda = 20,
  scale = TRUE,
  refit = TRUE,
  Uprev = NULL,
  useAUC = TRUE,
  intercept = TRUE,
  ...
)
```

Arguments

Z	A numeric matrix with features (rows) and samples (columns).
Chat	(Optional) Precomputed pseudo-inverse of priorMat; if NULL, it is calculated using ridge regularization.
priorMat	A numeric matrix with prior information (features x pathways).
penalty.factor	Optional penalty weights for features in priorMat.
pathwaySelection	Method to select candidate pathways: "fast" (default) or "complete".
alpha	Elastic net mixing parameter (0 = ridge, 1 = lasso). Default is 0.9.
maxPath	Maximum number of pathways/features selected per column. Default is 10.
nfolds	Number of cross-validation folds. Default is 5.
useSE	Whether to use the 1-standard-error rule for lambda selection. Default is FALSE.
top	If set, sets to 0 all but the top entries of Z per column before fitting.
binary	If TRUE, fits a binomial model (e.g., classification) to Z>0. Can be used in combination with top. Default is FALSE.
nlambda	Number of lambda values for glmnet. Default is 20.
scale	Whether to standardize predictors in glmnet. Default is TRUE.
refit	Whether to perform relaxed refitting using selected predictors. Default is TRUE.
Uprev	(Optional) Previous U matrix to reuse. In this mode only the columns of U that are all zero are estimated. Used internally in CLAMP.
useAUC	Logical; whether to compute pathway-LV associations using AUC (default TRUE) instead of OLS.
intercept	Logical; whether to include an intercept term in glmnet models. Default is TRUE.
...	Additional arguments passed to glmnet() or cv.glmnet().

Value

A list with one element:

`U` A matrix of loadings (features x components). Columns are named `LV1`, `LV2`, ...

Examples

```
set.seed(123)
genes <- paste0("G", 1:200)
lvs <- paste0("LV", 1:4)
paths <- paste0("Path", 1:60)

Z <- matrix(rnorm(200 * 4), nrow = 200, dimnames = list(genes, lvs))

priorMat <- matrix(rbinom(200 * 60, 1, 0.07),
    nrow = 200, dimnames = list(genes, paths)
)

fit1 <- solveU(
    Z = Z,
    priorMat = priorMat,
    pathwaySelection = "fast",
    alpha = 0.9,
    maxPath = 10,
    nfolds = 5,
    binary = FALSE,
    refit = TRUE
)
```

---

squashZscore *Squash extreme z-scores*

Description

Squash extreme z-scores

Usage

```
squashZscore(zdata, maxScore = 2)
```

Arguments

`zdata`	Numeric vector or matrix of z-scores.
`maxScore`	Numeric scalar, maximum absolute score (default 2).

Value

A numeric object of same dimensions as `zdata`, with values shrunk by a hyperbolic tangent transformation.

Examples

```
z <- rnorm(10, 0, 5)
squashZscore(z)
```

tscale *Row-wise scaling (mean 0, sd 1)*

Description

Standardizes each row of a numeric matrix to have mean 0 and standard deviation 1. Missing values are ignored in the computation of the mean and standard deviation.

Usage

```
tscale(x)
```

Arguments

x	A numeric matrix. Each row will be scaled independently.

Value

A numeric matrix of the same dimensions as x, where each row has mean 0 and standard deviation 1 (ignoring NAs). If a row has zero variance, it is returned unchanged.

See Also

base::scale()

Examples

```
mat <- matrix(seq_len(9), nrow = 3)
tscale(mat)
```

winsor_topk *Winsorize matrix columns by capping the top-k values*

Description

For each column, replaces values greater than the k-th largest entry with that threshold.

Usage

```
winsor_topk(M, k)
```

Arguments

M	A numeric matrix.
k	Integer; number of top elements to cap. Must be >= 1 and <= nrow(M).

Value

A numeric matrix of the same dimensions as M, winsorized per column.

Examples

```
set.seed(123)
M <- matrix(rnorm(20 * 5, mean = 0, sd = 2),
    nrow = 20,
    dimnames = list(paste0("Gene", 1:20), paste0("S", 1:5))
)

# Display column maxima before winsorization
apply(M, 2, max)

# Winsorize each column by capping top 3 values
M_winsor <- winsor_topk(M, k = 3)
```

---

xCell *xCell cell-signature matrix*

Description

A numeric matrix of cell-type signatures derived from xCell, used to represent the transcriptional profiles of distinct immune and stromal cell populations.

Usage

```
data(xCell)
```

Format

A numeric matrix with G genes (rows) and C cell types (columns). Row names are gene symbols; column names are xCell cell-type labels.

Value

A matrix of xCell-derived reference expression signatures.

Examples

```
data(xCell)
```

---

zscoreCLAMP *Z-score a filtered expression matrix for CLAMP*

Description

Centers each gene to mean 0 and scales to unit variance.

Usage

```
zscoreCLAMP(Y_filtered, rowStats)
```

Arguments

Y_filtered	Numeric matrix (genes x samples) returned by preprocessCLAMP
rowStats	Data frame with numeric columns mean and variance, row-named to match rownames(Y_filtered)

Value

Numeric matrix of the same dimensions as Y_filtered, with each row centered and scaled.

Examples

```
# simple 2 genes x 3 samples matrix
Y <- matrix(
    c(
        2, 4, 6, # gene1 counts
        8, 10, 12 # gene2 counts
    ),
    nrow = 2, byrow = TRUE,
    dimnames = list(c("gene1", "gene2"), paste0("sample", seq_len(3)))
)

# compute per-gene mean and variance
rowStats <- data.frame(
    mean = rowMeans(Y),
    variance = apply(Y, 1, var),
    row.names = rownames(Y)
)

# z-score each row
Y_z <- zscoreCLAMP(Y, rowStats)
```

---

zscoreCLAMPFBM *Z-score a filtered FBM in-place*

Description

Standardizes each row of an FBM using provided row means and variances.

Usage

```
zscoreCLAMPFBM(fbm_filtered, rowStats, chunk_size = 1000, ncores = 1)
```

Arguments

fbm_filtered	A bigstatsr::FBM produced by preprocessCLAMPFBM().
rowStats	A list with row_means and row_variances from that FBM.
chunk_size	Columns per block (default 1000).
ncores	Integer; number of cores to use for parallel operations (default 1).

Value

A normalized FBM with z-scored rows.

Examples

```
library(bigstatsr)
fbm <- FBM(
    nrow = 2, ncol = 4,
    init = matrix(seq_len(8), nrow = 2)
)
stats <- list(
    row_means     = rowMeans(fbm[]),
    row_variances = apply(fbm[], 1, var)
)
zscoreCLAMPFBM(fbm, stats, chunk_size = 2)
```

Index