Package: bigPCAcpp (via r-universe)

May 24, 2026

Type Package

Title Principal Component Analysis for 'bigmemory' Matrices

Version 0.9.1

Date 2026-03-25

Author Frederic Bertrand [aut, cre]

Maintainer Frederic Bertrand <`frederic.bertrand@lecnam.net`>

Description High performance principal component analysis routines that operate directly on bigmemory::big.matrix() objects. The package avoids materialising large matrices in memory by streaming data through 'BLAS' and 'LAPACK' kernels and provides helpers to derive scores, loadings, correlations, and contribution diagnostics, including utilities that stream results into 'bigmemory'-backed matrices for file-based workflows. Additional interfaces expose 'scalable' singular value decomposition, robust PCA, and robust SVD algorithms so that users can explore large matrices while tempering the influence of outliers. 'Scalable' principal component analysis is also implemented, Elgamal, Yabandeh, Aboulnaga, Mustafa, and Hefeeda (2015) <doi:10.1145/2723372.2751520>.

License GPL (>= 2)

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.12), methods, withr

Suggests bench, bigmemory, ggplot2, irlba, knitr, rmarkdown, testthat (>= 3.0.0)

LinkingTo Rcpp, bigmemory, BH

Encoding UTF-8

VignetteBuilder knitr

LazyLoad yes

NeedsCompilation yes

URL `https://fbertran.github.io/bigPCAcpp/`, `https://github.com/fbertran/bigPCAcpp`

BugReports https://github.com/fbertran/bigPCAcpp/issues

RoxygenNote 7.3.3

Config/testthat/edition 3

Repository https://cran.r-universe.dev

Date/Publication 2026-03-25 05:10:05 UTC

RemoteUrl https://github.com/cran/bigPCAcpp

RemoteRef HEAD

RemoteSha 27311fd781107b79ddc94e76cf7e405b3d966a9a

Contents

`bigPCAcpp-package` *bigPCAcpp: Principal Component Analysis for bigmemory Matrices*

Description

The **bigPCAcpp** package provides high-performance principal component analysis routines that work directly with `bigmemory::big.matrix` objects. Data are streamed through BLAS and LAPACK kernels so large, file-backed matrices can be analysed without materialising dense copies in R. Companion helpers compute scores, loadings, correlations, and contributions, including streaming variants that write results to `bigmemory::big.matrix` destinations used by file-based pipelines.

Author(s)

Maintainer: Frederic Bertrand <`frederic.bertrand@lecnam.net`>

See Also

`pca_bigmatrix()`, `pca_stream_bigmatrix()`

Examples

```
library(bigmemory)
mat <- as.big.matrix(matrix(rnorm(20), nrow = 5))
result <- pca_bigmatrix(mat)
result$sdev
```

`benchmark_results` *Benchmark timings for bigPCAcpp methods*

Description

A dataset summarising wall-clock performance for the main PCA entry points in **bigPCAcpp** across matrices of increasing size. The benchmarks compare the classical block-based decomposition, the streaming variant that writes rotations as it progresses, the scalable stochastic PCA implementation, and the base R `stats::prcomp()` routine for reference.

Format

A data frame with 360 rows and 14 columns:

dataset Human-readable size label ("small", "medium", "large", "xlarge").

rows Number of rows in the simulated matrix.

cols Number of columns in the simulated matrix.

ncomp Number of components requested.

method Computation strategy ("classical", "streaming", "scalable", or "prcomp").

replicate Replication index for repeated runs.

user_time User CPU time in seconds returned by `base::system.time()`.

system_time System CPU time in seconds.

elapsed Elapsed (wall-clock) time in seconds.

success Logical flag indicating whether the run completed without errors.

backend Name of the backend reported by the result object when the computation succeeded.

iterations Recorded iteration count for iterative methods when available (otherwise `NA`).

converged Logical convergence flag for iterative methods when available.

error Error message captured for failed runs (otherwise `NA`).

Source

Generated by `scripts/run_benchmark.R` using randomly simulated in-memory matrices (no file-backed storage).

Examples

```
data(benchmark_results)
```

`bigpca` *BigPCA result objects*

Description

Results returned by `pca_bigmatrix()`, `pca_stream_bigmatrix()`, and `pca_robust()` inherit from the `bigpca` class. The objects store the component standard deviations, rotation/loadings, and optional scores while recording which computational backend produced them. Standard S3 generics such as `summary()` and `plot()` are implemented for convenience.

`bigpca` objects are lists produced by `pca_bigmatrix()`, `pca_stream_bigmatrix()`, `pca_robust()`, and related helpers. They mirror the structure of base R's `prcomp()` outputs while tracking additional metadata for large-scale and streaming computations.

#' @seealso `pca_bigmatrix()`, `pca_stream_bigmatrix()`, `pca_robust()`, `pca_plot_scree()`, `pca_plot_scores()`, `pca_plot_contributions()`, `pca_plot_correlation_circle()`, and `pca_plot_biplot()`.

Components

`sdev` Numeric vector of component standard deviations.

`rotation` Numeric matrix whose columns contain the variable loadings (principal axes).

`center`, `scale` Optional numeric vectors describing the centring and scaling applied to each variable when fitting the model.

`scores` Optional numeric matrix of principal component scores when computed alongside the decomposition.

`column_sd` Numeric vector of marginal standard deviations for each input variable.

`eigenvalues` Numeric vector of eigenvalues associated with the retained components.

`explained_variance`, `cumulative_variance` Numeric vectors summarising the fraction of variance explained by individual components and the corresponding cumulative totals.

`covariance` Sample covariance matrix used to derive the components.

`nobs` Number of observations used in the decomposition.

The class also records the computation backend via `attr(x, "backend")`, enabling downstream methods to adjust their behaviour for streamed or robust results.

See Also

`pca_bigmatrix()`, `pca_stream_bigmatrix()`, `summary.bigpca()`, `print.summary.bigpca()`, `plot.bigpca()`

---

pca_bigmatrix *Principal component analysis for* `bigmemory::big.matrix` *inputs*

Description

Perform principal component analysis (PCA) directly on a `bigmemory::big.matrix` without copying the data into R memory. The exported helpers mirror the structure of base R's `prcomp()` while avoiding the need to materialise large matrices.

Usage

```
resolve_big_pointer(x, arg, allow_null = FALSE)

pca_scores_bigmatrix(
  xpMat,
  rotation,
  center,
  scale,
  ncomp = -1L,
  block_size = 1024L
)

pca_variable_loadings(rotation, sdev)

pca_variable_correlations(rotation, sdev, column_sd, scale = NULL)

pca_variable_contributions(loadings)

pca_individual_contributions(scores, sdev, total_weight = NA_real_)

pca_individual_cos2(scores)

pca_variable_cos2(correlations)

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

## S3 method for class 'summary.bigpca'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'bigpca'
plot(
  x,
  y,
  type = c("scree", "contributions", "correlation_circle", "biplot"),
  max_components = 25L,
  component = 1L,
```

```
  top_n = 20L,
  components = c(1L, 2L),
  data = NULL,
  draw = TRUE,
  ...
)
```

Arguments

`x`	A `summary.bigpca` object.
`arg`	Character string naming the argument being validated. Used to construct informative error messages.
`allow_null`	Logical flag indicating whether `NULL` is accepted for the argument. When `TRUE`, a `NULL` input is returned unchanged.
`xpMat`	Either a `bigmemory::big.matrix` or an external pointer such as `mat@address` that references the source `big.matrix`.
`rotation`	A rotation matrix such as the `rotation` element returned by `pca_bigmatrix()`.
`center`	For `pca_scores_bigmatrix()`, a numeric vector of column means (optional).
`scale`	Optional numeric vector of scaling factors returned by `pca_bigmatrix()`. If supplied, it indicates the PCA was performed on standardised variables.
`ncomp`	Number of components to retain. Use a non-positive value to keep all components returned by the decomposition.
`block_size`	Number of rows to process per block when streaming data through BLAS kernels. Larger values improve throughput at the cost of additional memory.
`sdev`	A numeric vector of component standard deviations, typically the `sdev` element from `pca_bigmatrix()`.
`column_sd`	A numeric vector with the marginal standard deviation of each original variable. When `scale` is supplied, correlations are computed on the standardised scale without rescaling by `column_sd`.
`loadings`	A numeric matrix such as the result of `pca_variable_loadings()`.
`scores`	For `pca_individual_contributions()` and `pca_individual_cos2()`, a numeric matrix of component scores where rows correspond to observations and columns to components.
`total_weight`	Optional positive scalar giving the effective number of observations to use when computing contributions. Defaults to the number of rows in `scores`.
`correlations`	For `pca_variable_cos2()`, a numeric matrix of correlations between variables and components.
`object`	A `bigpca` object created by `pca_bigmatrix()`, `pca_stream_bigmatrix()`, or related helpers.
`...`	Additional arguments passed to plotting helpers.
`digits`	Number of significant digits to display when printing importance metrics.
`y`	Currently unused.

`type`	The plot to draw. Options include "scree" (variance explained), "contributions" (top contributing variables), "correlation_circle" (variable correlations with selected components), and "biplot" (joint display of scores and loadings).
`max_components`	Maximum number of components to display in scree plots.
`component`	Component index to highlight when drawing contribution plots.
`top_n`	Number of variables to display in contribution plots.
`components`	Length-two integer vector selecting the components for correlation circle and biplot views.
`data`	Optional data source (matrix, data frame, `bigmemory::big.matrix`, or external pointer) used to compute scores for biplots when `x$scores` is unavailable.
`draw`	Logical; if `FALSE`, return the data prepared for the selected plot instead of drawing it.

Value

For `pca_bigmatrix()`, a `bigpca` object mirroring a `prcomp` result with elements `sdev`, `rotation`, optional `center` and `scale` vectors, `column_sd`, `eigenvalues`, `explained_variance`, `cumulative_variance`, and the sample covariance matrix. The object participates in S3 generics such as `summary()` and `plot()`.

A numeric matrix of scores with rows corresponding to observations and columns to retained components.

A numeric matrix containing variable loadings for each component.

A numeric matrix of correlations between variables and components.

A numeric matrix where each entry represents the contribution of a variable to a component.

For `summary.bigpca()`, a `summary.bigpca` object containing component importance measures.

Functions

- `pca_scores_bigmatrix()`: Project observations into principal component space while streaming from a `big.matrix`.
- `pca_variable_loadings()`: Compute variable loadings (covariances between original variables and components).
- `pca_variable_correlations()`: Compute variable-component correlations given column standard deviations.
- `pca_variable_contributions()`: Derive the relative contribution of each variable to the retained components.
- `pca_individual_contributions()`: Compute the relative contribution of individual observations to each component.
- `pca_individual_cos2()`: Compute squared cosine values measuring the quality of representation for individual observations.
- `pca_variable_cos2()`: Compute squared cosine values measuring the quality of representation for variables.
- `summary(bigpca)`: Summarise the component importance metrics for a `bigpca` result.

- `print(summary.bigpca)`: Print the component importance summary produced by `summary.bigpca()`.
- `plot(bigpca)`: Visualise PCA diagnostics such as scree, correlation circle, contribution, and biplot displays.

See Also

bigpca, `pca_scores_bigmatrix()`, `pca_variable_loadings()`, `pca_variable_correlations()`, `pca_variable_contributions()`, and the streaming variants `pca_stream_bigmatrix()` and companions.

bigpca

bigpca

bigpca

Examples

```
set.seed(123)
mat <- bigmemory::as.big.matrix(matrix(rnorm(40), nrow = 10))
pca <- pca_bigmatrix(mat, center = TRUE, scale = TRUE, ncomp = 3)
scores <- pca_scores_bigmatrix(mat, pca$rotation, pca$center, pca$scale, ncomp = 3)
loadings <- pca_variable_loadings(pca$rotation, pca$sdev)
correlations <- pca_variable_correlations(pca$rotation, pca$sdev, pca$column_sd, pca$scale)
contributions <- pca_variable_contributions(loadings)
list(scores = scores, loadings = loadings, correlations = correlations,
     contributions = contributions)
```

`pca_plot_biplot` *PCA biplot helper*

Description

Combines principal component scores and variable loadings in a single scatter plot. The helper accepts both standard matrices and `bigmemory::big.matrix` inputs, extracting only the requested component columns. When `draw = TRUE`, the function scales the loadings to match the score ranges, draws optional axes, overlays loading arrows, and labels observations when requested.

Usage

```
pca_plot_biplot(
  scores,
  loadings,
  components = c(1L, 2L),
  draw = TRUE,
  draw_axes = TRUE,
  draw_arrows = TRUE,
  label_points = FALSE,
  ...
)
```

Arguments

`scores`	Matrix or `bigmemory::big.matrix` containing principal component scores with observations in rows and components in columns.
`loadings`	Matrix or `bigmemory::big.matrix` of variable loadings whose columns correspond to principal components.
`components`	Integer vector of length two selecting the components to display.
`draw`	Logical; set to `FALSE` to return the prepared data without plotting.
`draw_axes`	Logical; when `TRUE`, horizontal and vertical axes are drawn through the origin.
`draw_arrows`	Logical; when `TRUE`, loading arrows are rendered.
`label_points`	Logical; when `TRUE`, point labels derived from row names are drawn next to the scores.
`...`	Additional graphical parameters passed to `graphics::plot()`.

Value

A list containing the selected `components`, extracted `scores`, original `loadings`, scaled loadings (`loadings_scaled`), and the applied `scale_factor`. The list is returned invisibly. When `draw = TRUE`, a biplot is produced using base graphics.

`pca_plot_contributions`

Plot variable contributions

Description

Highlights the variables that contribute most to a selected principal component. The helper works with dense matrices returned by `pca_variable_contributions()` as well as with `bigmemory::big.matrix` objects via sampling.

Usage

```
pca_plot_contributions(
  contributions,
  component = 1L,
  top_n = 20L,
  draw = TRUE,
  ...
)
```

Arguments

`contributions`	Contribution matrix where rows correspond to variables and columns to components.
`component`	Integer index of the component to visualise.
`top_n`	Number of variables with the largest absolute contribution to include in the bar plot.
`draw`	Logical; set to `FALSE` to skip plotting.
`...`	Additional arguments passed to `barplot()`.

Value

A data frame with the variables and their contributions is returned invisibly. When `draw = TRUE`, a bar plot of the top variables is produced.

`pca_plot_correlation_circle`

Plot a PCA correlation circle

Description

Visualises the correlation between each variable and a pair of principal components. The variables are projected onto the unit circle, where points near the perimeter indicate strong correlation with the selected components.

Usage

```
pca_plot_correlation_circle(
  correlations,
  components = c(1L, 2L),
  labels = NULL,
  draw = TRUE,
  ...
)
```

Arguments

`correlations`	Matrix or `bigmemory::big.matrix` containing variable correlations, typically produced by `pca_variable_correlations()`.
`components`	Length-two integer vector specifying the principal components to display.
`labels`	Optional character vector specifying the labels to display for each variable. When `NULL`, the row names of `correlations` are used when available.
`draw`	Logical; set to `FALSE` to return the prepared coordinates without plotting.
`...`	Additional graphical parameters passed to `graphics::plot()`.

Value

A data frame with `variable`, `PCx`, and `PCy` columns representing the projected correlations, where `PCx`/`PCy` correspond to the requested component indices. The data frame is returned invisibly.

`pca_plot_scores` *Plot sampled PCA scores*

Description

Streams a subset of observations through the PCA rotation and plots their scores on the requested components. Sampling keeps the drawn subset small so graphics remain interpretable even when the source big matrix contains millions of rows.

Usage

```
pca_plot_scores(
  x,
  rotation,
  center = numeric(),
  scale = numeric(),
  components = c(1L, 2L),
  max_points = 5000L,
  sample = c("uniform", "head"),
  seed = NULL,
  draw = TRUE,
  ...
)
```

Arguments

`x`	Either a `bigmemory::big.matrix`, a standard matrix, or a data frame.
`rotation`	A rotation matrix such as `pca_result$rotation`.
`center`	Optional centering vector. Use `numeric()` when no centering was applied.
`scale`	Optional scaling vector. Use `numeric()` when no scaling was applied.
`components`	Length-two integer vector selecting the principal components to display.
`max_points`	Maximum number of observations to sample for the plot.
`sample`	Strategy for selecting rows. `"uniform"` draws a random sample without replacement, whereas `"head"` takes the first `max_points` rows.
`seed`	Optional seed to make the sampling reproducible.
`draw`	Logical; set to `FALSE` to skip plotting and only return the sampled scores.
`...`	Additional graphical parameters forwarded to `plot()`.

Value

A list containing `indices` (the sampled row indices) and `scores` (the corresponding score matrix) is returned invisibly. When `draw = TRUE` a scatter plot is produced.

`pca_plot_scree` *Scree plot for principal component importance*

Description

Displays the proportion of variance explained by the leading principal components. The function caps the number of displayed components to keep the visualization legible on very high-dimensional problems.

Usage

```
pca_plot_scree(
  pca_result,
  max_components = 25L,
  cumulative = TRUE,
  draw = TRUE,
  ...
)
```

Arguments

`pca_result`	A list created by `pca_bigmatrix()` or `pca_stream_bigmatrix()` containing standard deviation and explained variance elements.
`max_components`	Maximum number of components to display. Defaults to 25 or the available number of components, whichever is smaller.
`cumulative`	Logical flag indicating whether to overlay the cumulative explained variance line.
`draw`	Logical; set to `FALSE` to return the prepared data without drawing a plot (useful for testing).
`...`	Additional parameters passed to `plot()`.

Value

A list with `component`, `explained`, and `cumulative` vectors is returned invisibly. When `draw = TRUE`, the function produces a scree plot using base graphics.

`pca_plots` *Plot PCA diagnostics for big data workflows*

Description

These helpers visualise the results returned by `pca_bigmatrix()` and its companions without requiring users to materialise dense intermediate structures. Each plotting function optionally samples the inputs so the default output remains responsive even when the underlying big matrix spans millions of observations.

See Also

`pca_bigmatrix()`, `pca_variable_loadings()`, `pca_variable_contributions()`

`pca_robust` *Robust principal component analysis*

Description

Compute principal component analysis (PCA) using robust measures of location and scale so that extreme observations have a reduced influence on the resulting components. The implementation centres each variable by its median and, when requested, scales by the median absolute deviation (MAD) before performing an iteratively reweighted singular value decomposition that down-weights observations with unusually large reconstruction errors.

Usage

```
pca_robust(x, center = TRUE, scale = FALSE, ncomp = NULL)
```

Arguments

`x`	A numeric matrix, data frame, or an object coercible to a numeric matrix. Missing values are not supported.
`center`	Logical; should variables be centred by their median before applying PCA?
`scale`	Logical; when `TRUE`, variables are scaled by the MAD after centring. Scaling requires `center = TRUE`.
`ncomp`	Number of components to retain. Use `NULL` or a non-positive value to keep all components returned by the decomposition.

Value

A `bigpca` object mirroring the structure of `pca_bigmatrix()` with robust estimates of location, scale, and variance metrics.

Examples

```
set.seed(42)
x <- matrix(rnorm(50), nrow = 10)
x[1, 1] <- 25  # outlier
robust <- pca_robust(x, ncomp = 2)
robust$sdev
```

`pca_spca` *Scalable principal component analysis via streaming power iterations*

Description

Implements the scalable PCA (sPCA) procedure of Elgamal et al. (2015), which uses block power iterations to approximate the leading principal components while streaming the data in manageable chunks. The algorithm only requires matrix-vector products, allowing large matrices to be processed without materialising the full cross-product in memory.

Implements the scalable PCA (sPCA) procedure of Elgamal et al. (2015), which uses block power iterations to approximate the leading principal components while streaming the data in manageable chunks. The algorithm only requires matrix-vector products, allowing large matrices to be processed without materialising the full cross-product in memory.

Implements the scalable PCA (sPCA) procedure of Elgamal et al. (2015), which uses block power iterations to approximate the leading principal components while streaming the data in manageable chunks. The algorithm only requires matrix-vector products, allowing large matrices to be processed without materialising the full cross-product in memory.

Usage

```
pca_spca(
  x,
  ncomp = NULL,
  center = TRUE,
  scale = FALSE,
  block_size = 2048L,
  max_iter = 50L,
  tol = 1e-04,
  seed = NULL,
  return_scores = FALSE,
  verbose = FALSE
)
```

```
pca_spca(
  x,
  ncomp = NULL,
  center = TRUE,
  scale = FALSE,
  block_size = 2048L,
  max_iter = 50L,
  tol = 1e-04,
  seed = NULL,
  return_scores = FALSE,
  verbose = FALSE
)
```

```
pca_spca_R(
  x,
  ncomp = NULL,
  center = TRUE,
  scale = FALSE,
  block_size = 2048L,
  max_iter = 50L,
  tol = 1e-04,
  seed = NULL,
  return_scores = FALSE,
  verbose = FALSE
)

pca_spca(
  x,
  ncomp = NULL,
  center = TRUE,
  scale = FALSE,
  block_size = 2048L,
  max_iter = 50L,
  tol = 1e-04,
  seed = NULL,
  return_scores = FALSE,
  verbose = FALSE
)

pca_spca_R(
  x,
  ncomp = NULL,
  center = TRUE,
  scale = FALSE,
  block_size = 2048L,
  max_iter = 50L,
  tol = 1e-04,
  seed = NULL,
  return_scores = FALSE,
  verbose = FALSE
)
```

Arguments

`x`	A numeric matrix, data frame, `bigmemory::big.matrix`, or an external pointer referencing a big.matrix. The input is processed in row-wise blocks so that large matrices can be analysed without creating dense copies in R memory.
`ncomp`	Number of principal components to retain. Use `NULL` or a non-positive value to keep `min(nrow(x), ncol(x))` components.
`center`	Logical; should column means be subtracted before performing PCA?

`scale`	Logical; when `TRUE`, columns are scaled to unit variance after centring. Scaling requires `center = TRUE`.
`block_size`	Number of rows to stream per block when computing column statistics and matrix-vector products.
`max_iter`	Maximum number of block power iterations.
`tol`	Convergence tolerance applied to the Frobenius norm of the difference between successive subspace projectors.
`seed`	Optional integer seed used to initialise the random starting basis.
`return_scores`	Logical; when `TRUE`, principal component scores are computed in a final streaming pass over the data.
`verbose`	Logical; when `TRUE`, diagnostic messages describing the iteration progress are emitted.

Value

A `bigpca` object containing the approximate PCA solution with the same structure as `pca_bigmatrix()`. The result includes component standard deviations, rotation/loadings, optional scores, column statistics, and variance summaries. Additional metadata is stored in `attr(result, "iterations")` (number of iterations performed), `attr(result, "tolerance")` (requested tolerance), and `attr(result, "converged")` (logical convergence flag).

A `bigpca` object containing the approximate PCA solution with the same structure as `pca_bigmatrix()`. The result includes component standard deviations, rotation/loadings, optional scores, column statistics, and variance summaries. Additional metadata is stored in `attr(result, "iterations")` (number of iterations performed), `attr(result, "tolerance")` (requested tolerance), and `attr(result, "converged")` (logical convergence flag).

A `bigpca` object containing the approximate PCA solution with the same structure as `pca_bigmatrix()`. The result includes component standard deviations, rotation/loadings, optional scores, column statistics, and variance summaries. Additional metadata is stored in `attr(result, "iterations")` (number of iterations performed), `attr(result, "tolerance")` (requested tolerance), and `attr(result, "converged")` (logical convergence flag).

References

Tarek Elgamal, Maysam Yabandeh, Ashraf Aboulnaga, Waleed Mustafa, and Mohamed Hefeeda (2015). *sPCA: Scalable Principal Component Analysis for Big Data on Distributed Platforms*. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. doi: 10.1145/2723372.2751520.

Tarek Elgamal, Maysam Yabandeh, Ashraf Aboulnaga, Waleed Mustafa, and Mohamed Hefeeda (2015). *sPCA: Scalable Principal Component Analysis for Big Data on Distributed Platforms*. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. doi: 10.1145/2723372.2751520.

Tarek Elgamal, Maysam Yabandeh, Ashraf Aboulnaga, Waleed Mustafa, and Mohamed Hefeeda (2015). *sPCA: Scalable Principal Component Analysis for Big Data on Distributed Platforms*. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. doi: 10.1145/2723372.2751520.

`pca_stream_bigmatrix` *Streaming big.matrix PCA helpers*

Description

Variants of the PCA helpers that stream results directly into `bigmemory::big.matrix` objects, enabling file-backed workflows without materialising dense R matrices.

Usage

```
pca_spca_stream_bigmatrix(
  xpMat,
  xpRotation = NULL,
  center = TRUE,
  scale = FALSE,
  ncomp = -1L,
  block_size = 2048L,
  max_iter = 50L,
  tol = 1e-04,
  seed = NULL,
  return_scores = FALSE,
  verbose = FALSE
)

pca_scores_stream_bigmatrix(
  xpMat,
  xpDest,
  rotation,
  center,
  scale,
  ncomp = -1L,
  block_size = 1024L
)

pca_variable_loadings_stream_bigmatrix(xpRotation, sdev, xpDest)

pca_variable_correlations_stream_bigmatrix(
  xpRotation,
  sdev,
  column_sd,
  scale = NULL,
  xpDest
)

pca_variable_contributions_stream_bigmatrix(xpLoadings, xpDest)
```

Arguments

`xpMat`	Either a `bigmemory::big.matrix` or an external pointer such as `mat@address` that references the source `big.matrix`.
`xpRotation`	For `pca_variable_correlations_stream_bigmatrix()`, a `bigmemory::big.matrix` or external pointer containing the rotation matrix to stream from.
`center`	For `pca_scores_bigmatrix()`, a numeric vector of column means (optional).
`scale`	Optional numeric vector of scaling factors returned by `pca_stream_bigmatrix()` or `pca_bigmatrix()`. When supplied, correlations are reported on the scaled data without dividing by `column_sd`.
`ncomp`	Number of components to retain. Use a non-positive value to keep all components returned by the decomposition.
`block_size`	Number of rows to process per block when streaming data through BLAS kernels. Larger values improve throughput at the cost of additional memory.
`max_iter`	Maximum number of block power iterations.
`tol`	Convergence tolerance applied to the Frobenius norm of the difference between successive subspace projectors.
`seed`	Optional integer seed used to initialise the random starting basis.
`return_scores`	Logical; when `TRUE`, principal component scores are computed in a final streaming pass over the data.
`verbose`	Logical; when `TRUE`, diagnostic messages describing the iteration progress are emitted.
`xpDest`	Either a `big.matrix` or external pointer referencing the destination `big.matrix` that stores the computed quantity.
`rotation`	A rotation matrix such as the `rotation` element returned by `pca_bigmatrix()`.
`sdev`	A numeric vector of component standard deviations, typically the `sdev` element from `pca_bigmatrix()`.
`column_sd`	A numeric vector of variable standard deviations used to scale the correlations when the PCA was performed on unscaled data.
`xpLoadings`	For `pca_variable_contributions_stream_bigmatrix()`, the loadings matrix supplied as a `big.matrix` or external pointer.

Value

For `pca_stream_bigmatrix()`, the same bigpca object as `pca_bigmatrix()` with the addition of a `rotation_stream_bigmatrix` element referencing the populated `big.matrix` when `xpRotation` is supplied. For `pca_spca_stream_bigmatrix()`, the same scalable PCA structure as `pca_spca()` with the optional pointer populated when provided.

The external pointer supplied in `xpDest`, invisibly.

Functions

- `pca_scores_stream_bigmatrix()`: Stream PCA scores into a destination big.matrix.
- `pca_variable_loadings_stream_bigmatrix()`: Populate big.matrix objects with derived variable diagnostics.

- `pca_variable_correlations_stream_bigmatrix()`: Stream variable correlations into a destination big.matrix.
- `pca_variable_contributions_stream_bigmatrix()`: Stream variable contributions into a destination big.matrix.

Examples

```
set.seed(456)
mat <- bigmemory::as.big.matrix(matrix(rnorm(30), nrow = 6))
ncomp <- 2
rotation_store <- bigmemory::big.matrix(ncol(mat), ncomp, type = "double")
pca_stream <- pca_stream_bigmatrix(mat, xpRotation = rotation_store, ncomp = ncomp)
score_store <- bigmemory::big.matrix(nrow(mat), ncomp, type = "double")
pca_scores_stream_bigmatrix(
    mat,
    score_store,
    pca_stream$rotation,
    pca_stream$center,
    pca_stream$scale,
    ncomp = ncomp
)
loadings_store <- bigmemory::big.matrix(ncol(mat), ncomp, type = "double")
pca_variable_loadings_stream_bigmatrix(
    pca_stream$rotation_stream_bigmatrix,
    pca_stream$sdev,
    loadings_store
)
correlation_store <- bigmemory::big.matrix(ncol(mat), ncomp, type = "double")
pca_variable_correlations_stream_bigmatrix(
    pca_stream$rotation_stream_bigmatrix,
    pca_stream$sdev,
    pca_stream$column_sd,
    pca_stream$scale,
    correlation_store
)
contribution_store <- bigmemory::big.matrix(ncol(mat), ncomp, type = "double")
pca_variable_contributions_stream_bigmatrix(
    loadings_store,
    contribution_store
)
```

`pca_supplementary_individuals`

Supplementary individual diagnostics

Description

Compute principal component scores and quality metrics for supplementary individuals (rows) projected into an existing PCA solution.

Usage

```
pca_supplementary_individuals(
  data,
  rotation,
  sdev,
  center = NULL,
  scale = NULL,
  total_weight = NA_real_
)
```

Arguments

`data`	Matrix-like object whose rows correspond to supplementary individuals and columns to the original variables.
`rotation`	Rotation matrix from the PCA model (e.g. the `rotation` element of a `bigpca` result).
`sdev`	Numeric vector of component standard deviations associated with `rotation`.
`center`	Optional numeric vector giving the centring applied to each variable when fitting the PCA. Defaults to zero centring.
`scale`	Optional numeric vector describing the scaling applied to each variable when fitting the PCA. When `NULL`, no scaling is applied.
`total_weight`	Optional positive scalar passed to `pca_individual_contributions()` when computing contributions. When left as `NA` (the default), the resulting contributions for each component are normalised to sum to one across supplementary individuals. Supplying a value bypasses this normalisation and delegates the scaling to `pca_individual_contributions()`.

Value

A list with elements `scores`, `contributions`, and `cos2`.

`pca_supplementary_variables` *Supplementary variable diagnostics*

Description

Compute loadings, correlations, contributions, and cos^2 values for supplementary variables (columns) given component scores for the active individuals.

Usage

```
pca_supplementary_variables(data, scores, sdev, center = NULL)
```

Arguments

`data`	Matrix-like object whose columns correspond to supplementary variables measured on the active individuals.
`scores`	Numeric matrix of component scores for the active individuals.
`sdev`	Numeric vector of component standard deviations associated with `scores`.
`center`	Optional numeric vector specifying the centring to apply to each supplementary variable. When `NULL`, column means of `data` are used.

Value

A list with elements `loadings`, `correlations`, `contributions`, and `cos2`.

`svd_bigmatrix` *Singular value decomposition for* `bigmemory::big.matrix` *inputs*

Description

Compute the singular value decomposition (SVD) of a `bigmemory::big.matrix` without materialising it as a base R matrix. Blocks of rows are streamed through BLAS before LAPACK is invoked so that even moderately large matrices can be decomposed efficiently.

Usage

```
svd_bigmatrix(
  xpMat,
  nu = -1L,
  nv = -1L,
  block_size = 1024L,
  method = c("dgesdd", "dgesvd")
)
```

Arguments

`xpMat`	Either a `bigmemory::big.matrix` or an external pointer such as `mat@address` that references the source `big.matrix`.
`nu`	Number of left singular vectors to return. Use a negative value to request the default of `min(nrow, ncol)` vectors and zero to skip returning `u` entirely.
`nv`	Number of right singular vectors to return. Use a negative value to request the default of `min(nrow, ncol)` vectors and zero to skip returning `v` entirely.
`block_size`	Number of rows to process per block when streaming data into BLAS kernels. Larger values can improve throughput at the cost of additional temporary memory.
`method`	LAPACK backend used to compute the decomposition. The default uses the divide-and-conquer routine `dgesdd` and falls back to `dgesvd` when required.

Value

A list with components `u`, `d`, and `v` analogous to base R's `svd()` output. When `nu` or `nv` are zero the corresponding matrix has zero columns.

Examples

```
set.seed(42)
mat <- bigmemory::as.big.matrix(matrix(rnorm(20), nrow = 5))
svd_res <- svd_bigmatrix(mat, nu = 2, nv = 2)
svd_res$d
```

`svd_robust` *Robust singular value decomposition (C++ backend)*

Description

Compute the iteratively reweighted SVD using the high-performance C++ implementation. The interface mirrors `svd_robust_R()` while delegating the heavy lifting to compiled code.

Usage

```
svd_robust(
  x,
  ncomp,
  max_iter = 25L,
  tol = sqrt(.Machine$double.eps),
  huber_k = 1.345
)
```

Arguments

`x`	Numeric matrix for which the decomposition should be computed.
`ncomp`	Number of leading components to retain.
`max_iter`	Maximum number of reweighting iterations.
`tol`	Convergence tolerance applied to successive changes in the row weights and singular values.
`huber_k`	Tuning constant controlling the aggressiveness of the Huber weight function. Larger values down-weight fewer observations.

Value

A list containing the left and right singular vectors (`u` and `v`), the singular values (`d`), the final row weights (`weights`), and the number of iterations required for convergence (`iterations`).

`svd_robust_R` *Iteratively reweighted singular value decomposition*

Description

Internal helper used by `pca_robust()` to compute a singular value decomposition that is less sensitive to individual rows with extreme values. The routine alternates between computing the SVD of a row-weighted matrix and updating the weights via a Huber-type scheme based on the reconstruction residuals.

Usage

```
svd_robust_R(
  x,
  ncomp,
  max_iter = 25L,
  tol = sqrt(.Machine$double.eps),
  huber_k = 1.345
)
```

Arguments

`x`	Numeric matrix for which the decomposition should be computed.
`ncomp`	Number of leading components to retain.
`max_iter`	Maximum number of reweighting iterations.
`tol`	Convergence tolerance applied to successive changes in the row weights and singular values.
`huber_k`	Tuning constant controlling the aggressiveness of the Huber weight function. Larger values down-weight fewer observations.

Value

A list containing the left and right singular vectors (`u` and `v`), the singular values (`d`), the final row weights (`weights`), and the number of iterations required for convergence (`iterations`). The structure mirrors base R's `base::svd()` output with additional metadata.

Index