# Package 'rsvd'

November 2, 2026

**Type** Package

**Title** Randomized Singular Value Decomposition

**Version** 2.0.0

**Author** N. Benjamin Erichson [aut, cre]

**Maintainer** N. Benjamin Erichson <erichson@icsi.berkeley.edu>

**Description** Computes low-rank matrix approximations for dimension reduction, data analysis, and compression. Randomized algorithms use random weighted combinations of the input to find a small subspace, then factorize the projected matrix. This can reduce the work when the requested rank is small relative to the matrix dimensions and the input has a useful low-rank approximation. Methods include singular value decomposition, principal component analysis, QB factorization, and interpolative and CUR decompositions. Robust principal component analysis separates a matrix into low-rank and entrywise sparse components. Selected routines support sparse matrices, including principal component analysis with implicit centering and scaling. Plotting functions display scores, loadings, and explained variance. See Erichson et al. (2019) <doi:10.18637/jss.v089.i11> for the algorithms and examples of their use.

**Depends** R (>= 4.4.0)

**Imports** Matrix (>= 1.7-0), methods

**License** GPL (>= 3)

**LazyData** TRUE

**LazyDataCompression** xz

**URL** https://github.com/erichson/rSVD

**BugReports** https://github.com/erichson/rSVD/issues

**Suggests** ggplot2, testthat (>= 3.0.0)

**NeedsCompilation** no

**Encoding** UTF-8

**Config/roxygen2/version** 8.1.1

**Repository** CRAN

**Date/Publication** 2026-11-02 09:30:02 UTC

# Contents

digits . . . 2
ggbiplot . . . 3
ggcorplot . . . 5
ggindplot . . . 7
ggscreeplot . . . 9
plot.rpca . . . 10
rcur . . . 12
rid . . . 14
rpca . . . 17
rqb . . . 21
rrpca . . . 24
rsvd . . . 27
tiger . . . 30

**Index** **32**

---

`digits` *Handwritten digit images*

---

## Description

A subset of MNIST containing 3,000 handwritten examples of each of the digits 0, 1, 2 and 3. Each observation consists of a class label and a flattened 28 by 28 grayscale image.

## Usage

```
data("digits")
```

## Format

An integer matrix with 12,000 rows and 785 columns. The first column, `label`, contains the digit class (0, 1, 2 or 3). The remaining 784 columns, named `pixel0` through `pixel783`, contain grayscale pixel values from 0 to 255. There are 3,000 rows per class.

## Details

Use only the pixel columns when fitting a decomposition or another unsupervised method. Keeping class labels separate allows the fitted representation to be compared with the known digits without treating the labels as image features. The stored intensities are not centered, standardized or divided by 255. Some background pixels may be constant across a chosen subset, so scaling every pixel to unit variance can be inappropriate.

The example reshapes the 784 pixel values from one observation into a 28 by 28 matrix. Reversing its second axis gives the display orientation used by `image`. The reversed grayscale palette shows dark handwriting on a light background.

## References

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

## Examples

```
data("digits")
dim(digits)
table(digits[, "label"])

# Keep the label separate from the pixels.
label <- digits[1L, "label"]
digit <- matrix(digits[1L, -1L], nrow = 28, ncol = 28)
image(digit[, 28:1], col = gray((255:0) / 255), asp = 1,
      axes = FALSE, useRaster = TRUE, main = paste("Digit", label))
```

---

`ggbiplot` *PCA scores and variable directions in one plot*

---

## Description

Display observations in a plane defined by two retained principal components from `rpca`, with arrows showing the corresponding variable directions. The combined view relates variation among the observations to the variables used in the fit. The optional **ggplot2** package is required.

## Usage

```
ggbiplot(
  rpcaObj,
  pcs = c(1, 2),
  loadings = TRUE,
  groups = NULL,
  alpha = 0.6,
  ellipse = TRUE,
  alpha.ellipse = 0.2,
  var_labels = TRUE,
  var_labels.names = NULL,
  ind_labels = TRUE,
  ind_labels.names = NULL
)
```

## Arguments

| | |
|---|---|
| `rpcaObj` | An object returned by `rpca`, with real-valued scores and directions. Complex factor maps are not supported. Fit with `retx = TRUE` so that `rpcaObj$x` is available. |

pcs — Integer vector of length two selecting the retained principal components to plot, in horizontal and vertical order. The default is `c(1, 2)`. Both indices must be finite integers within the retained component range.

loadings — Logical; when `TRUE` (the default), multiply each selected direction by its component standard deviation, so the coordinates are `rotation[, pcs]` scaled columnwise by `sdev[pcs]`. When `FALSE`, plot the unscaled entries of `rotation[, pcs]`.

groups — Optional factor with one entry for each row of `rpcaObj$x`, in the same order. The length must match the number of score rows; shorter inputs are not recycled. Groups control point colours and, when requested, ellipses. The default, `NULL`, uses one colour and draws no ellipse. If `rpca` omitted incomplete observations, supply groups for the retained rows only.

alpha — Numeric value between zero and one controlling point transparency, with zero fully transparent and one opaque. The default is `0.6`.

ellipse — Logical; draw a filled data ellipse for each supplied group when `TRUE` (the default). The defaults of `stat_ellipse` give a multivariate-t ellipse at level `0.95`. See Details for its interpretation.

alpha.ellipse — Numeric value between zero and one controlling ellipse transparency. The default is `0.2`; ignored when no ellipse is drawn.

var_labels — Logical; label the variable arrows when `TRUE` (the default).

var_labels.names — Optional character vector with one label per row of `rpcaObj$rotation`, in the original variable order. By default, the rotation row names are used, or sequential variable numbers when names are absent. Labels are displayed only when `var_labels = TRUE`.

ind_labels — Logical; show a text label for each observation when `TRUE` (the default). Set to `FALSE` to reduce clutter.

ind_labels.names — Optional character vector with one label per row of `rpcaObj$x`, in the same order. By default, the score row names are used, or sequential row numbers when names are absent. Labels are displayed only when `ind_labels = TRUE`.

## Details

Each point represents one observation, with coordinates taken directly from `rpcaObj$x`. These fitted component scores reflect the centering and scaling requested in `rpca`. The plotting function leaves their normalization unchanged. Axis labels give each selected component's share of the total processed variance in `rpcaObj$var`, which includes variance outside the retained components.

Each arrow represents one variable. The `loadings` argument changes its coordinates by including or omitting the component standard deviations. Arrows and observations retain their respective loading and score scales; the function does not rescale them to match visually.

For centered variables with unit sample variance and nonzero component variance, the default coordinates equal the correlations with fitted scores for a deterministic SVD. A randomized fit can introduce approximation error in this interpretation. Other preprocessing choices give scaled directions that need not be correlations. Use `ggcorplot` to examine the directions separately with a unit circle.

When groups are supplied, data ellipses summarize the spread of fitted scores within each group under the multivariate-t model used by `stat_ellipse`. They do not estimate confidence regions for group means or test group separation. The multivariate-t computation also requires the recommended package **MASS**. Small groups or degenerate score distributions may not support ellipse estimation; use `ellipse = FALSE` in that case.

## Value

A **ggplot2** plot object. Print the object to draw the plot, or add layers, labels, and themes with **ggplot2**.

## Author(s)

N. Benjamin Erichson

## See Also

`rpca`, `ggindplot`, `ggcorplot`, `ggscreeplot`, `ggplot`, `stat_ellipse`

## Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  set.seed(42)
  fit <- rpca(iris[, 1:4], k = 2)
  p <- ggbiplot(fit, groups = iris$Species, ellipse = FALSE,
                ind_labels = FALSE)
  print(p + ggplot2::labs(title = "Iris scores and variable directions"))
}
```

---

# ggcorplot *PCA variable directions and correlation circle*

---

## Description

Display how variables are represented in a plane defined by two retained principal components. Arrows show their directions, optionally scaled by the component standard deviations, and a unit circle provides a reference for interpreting standardized fits. The optional **ggplot2** package is required.

## Usage

```
ggcorplot(
  rpcaObj,
  pcs = c(1, 2),
  loadings = TRUE,
  var_labels = FALSE,
  var_labels.names = NULL,
  alpha = 1,
  top.n = NULL
)
```

## Arguments

| | |
|---|---|
| rpcaObj | An object returned by rpca, with real-valued directions. Complex factor maps are not supported. Scores are not needed, so fits made with retx = FALSE are supported. |
| pcs | Integer vector of length two selecting the retained principal components to plot, in horizontal and vertical order. The default is c(1, 2). Both indices must be finite integers within the retained component range. |
| loadings | Logical; when TRUE (the default), multiply each selected direction by its component standard deviation, so the coordinates are rotation[, pcs] scaled columnwise by sdev[pcs]. When FALSE, plot the unscaled entries of rotation[, pcs]. |
| var_labels | Logical; show text labels for the variables selected by top.n when TRUE. The default is FALSE. |
| var_labels.names | Optional character vector with one label per row of rpcaObj$rotation, in the original variable order. By default, the rotation row names are used, or sequential variable numbers when names are absent. These names identify the endpoint markers and, when requested, the text labels. |
| alpha | Numeric value between zero and one controlling arrow transparency. The default is 1 (opaque). |
| top.n | Positive integer limiting the number of variables selected for endpoint markers and text labels. Selection is by decreasing arrow length in the plotted two-component plane. The default, NULL, selects all variables; values larger than the number of variables are clamped to that number. See Details for the marker display rule. |

## Details

The direction coefficients describe how each original variable enters a component. Multiplying a coefficient by the component standard deviation also accounts for the variation captured by that component. With loadings = TRUE, variable $j$ therefore has coordinate $V_{ji}\sqrt{\lambda_i}$ on component $i$, where $V$ is rotation and $\lambda_i$ is the corresponding eigenvalue. This is the direction entry multiplied by sdev[i]. With loadings = FALSE, the coordinates are the unscaled direction coefficients, which changes relative arrow lengths and may change which variables top.n selects.

For centered variables with unit sample variance and nonzero component variance, the scaled coordinates equal the correlations with fitted scores for a deterministic SVD. A randomized fit can introduce approximation error in this interpretation. In this setting, the unit circle helps show how well the selected component plane represents each variable. With unscaled or uncentered input, the coordinates need not be correlations and the circle serves as a visual reference. Equal horizontal and vertical scales preserve the circle's shape.

Every variable is shown by an arrow. To make a crowded plot easier to read, top.n limits the variables selected for endpoint markers and, when var_labels = TRUE, text labels. Selection uses arrow length within the displayed plane, so the ranking applies only to these two components. Endpoint markers are drawn only when the effective number selected is less than 50.

Axis labels give each selected component's share of rpcaObj$var, the total variance of the processed input. The denominator includes variance outside the retained components.

## Value

A **ggplot2** plot object. Print the object to draw the plot, or add layers, labels, and themes with **ggplot2**.

## Author(s)

N. Benjamin Erichson

## See Also

rpca, ggbiplot, ggindplot, ggscreeplot, ggplot

## Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  set.seed(42)
  fit <- rpca(iris[, 1:4], k = 2, retx = FALSE)
  # Label the three longest arrows in the selected component plane.
  print(ggcorplot(fit, var_labels = TRUE, top.n = 3))
}
```

---

# ggindplot *PCA observation scores*

---

## Description

Display observations in a plane defined by two retained principal components from rpca. Each point shows an observation's fitted scores, allowing patterns in the selected components to be compared across observations or supplied groups. Labels and group ellipses can be added. The optional **ggplot2** package is required.

## Usage

```
ggindplot(
  rpcaObj,
  pcs = c(1, 2),
  groups = NULL,
  alpha = 0.6,
  ellipse = TRUE,
  alpha.ellipse = 0.2,
  ind_labels = TRUE,
  ind_labels.names = NULL
)
```

## Arguments

| | |
|---|---|
| rpcaObj | An object returned by rpca, with real-valued scores. Complex factor maps are not supported. Fit with retx = TRUE so that rpcaObj$x is available. |
| pcs | Integer vector of length two selecting the retained principal components to plot, in horizontal and vertical order. The default is c(1, 2). Both indices must be finite integers within the retained component range. |
| groups | Optional factor with one entry for each row of rpcaObj$x, in the same order. The length must match the number of score rows; shorter inputs are not recycled. Groups control point colours and, when requested, ellipses. The default, NULL, uses one colour and draws no ellipse. If rpca omitted incomplete observations, supply groups for the retained rows only. |
| alpha | Numeric value between zero and one controlling point transparency, with zero fully transparent and one opaque. The default is 0.6. |
| ellipse | Logical; draw a filled data ellipse for each supplied group when TRUE (the default). The defaults of stat_ellipse give a multivariate-t ellipse at level 0.95. See Details for its interpretation. |
| alpha.ellipse | Numeric value between zero and one controlling ellipse transparency. The default is 0.2; ignored when no ellipse is drawn. |
| ind_labels | Logical; show a text label for each observation when TRUE (the default). Set to FALSE to reduce clutter. |
| ind_labels.names | Optional character vector with one label per row of rpcaObj$x, in the same order. By default, the score row names are used, or sequential row numbers when names are absent. Labels are displayed only when ind_labels = TRUE. |

## Details

The horizontal and vertical coordinates are the columns of rpcaObj$x selected by pcs. These fitted scores retain the centering and scaling used in rpca; the plotting function does not standardize or recompute them. For randomized fits, they are the stored projected-SVD scores described in rpca.

Each axis label gives its component's share of rpcaObj$var, the total variance of the processed input. Variance in omitted components remains in the denominator, so a view of two components need not account for all of the variation.

When groups are supplied, point colours identify group membership and data ellipses summarize the spread of fitted scores within each group. The ellipses use the multivariate-t model in stat_ellipse, whose computation also requires the recommended package **MASS**. They do not estimate confidence regions for group means or test group separation. Small groups or degenerate score distributions may not support ellipse estimation; use ellipse = FALSE in that case.

## Value

A **ggplot2** plot object. Print the object to draw the plot, or add layers, labels, and themes with **ggplot2**.

## Author(s)

N. Benjamin Erichson

## See Also

rpca, ggbiplot, ggcorplot, ggscreeplot, ggplot, stat_ellipse

## Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  set.seed(42)
  fit <- rpca(iris[, 1:4], k = 3)
  print(ggindplot(fit, pcs = c(1, 3), groups = iris$Species,
                  ellipse = FALSE, ind_labels = FALSE))
}
```

---

# ggscreeplot *PCA scree plot and explained variance*

---

## Description

Show how much variation each retained principal component from rpca captures in the processed data. Plot individual component variances, their proportions of total variance, or the cumulative proportion to assess how much variation the retained representation captures. The optional **ggplot2** package is required.

## Usage

```
ggscreeplot(rpcaObj, type = c("var", "ratio", "cum", "eigenvals"))
```

## Arguments

| | |
|---|---|
| rpcaObj | An object returned by rpca. Scores are not needed, so fits made with retx = FALSE are supported. The variance quantities are real-valued for both real and complex PCA. |
| type | Character string selecting the vertical-axis quantity. "var" (the default) plots sdev^2; "ratio" plots sdev^2 / var; "cum" plots the cumulative sum of those ratios; and "eigenvals" plots eigvals. For an rpca object, "var" and "eigenvals" give the same values with different axis labels. Unambiguous partial matches are accepted. |

## Details

The horizontal axis follows the retained components in their fitted order. The variance and eigenvalue views show how much variation each component captures, in squared units of the processed input. The ratio view expresses each amount as a proportion of total variance; the cumulative view adds those proportions through each displayed component.

Both proportion views use rpcaObj$var as their denominator. This total includes energy outside the retained components, so the cumulative curve can end below one when the fit retains only a

low-rank approximation. The proportions lie between zero and one when total processed variance is positive. They are undefined for an input with zero total processed variance.

Interpretation follows the preprocessing used in rpca. With `center = FALSE`, the quantities use uncentered sums of squares, which include the contribution of any nonzero column means. They need not equal variances computed after subtracting the means of the fitted score columns.

## Value

A **ggplot2** plot object. Print the object to draw the plot, or add layers, labels, and themes with **ggplot2**.

## Author(s)

N. Benjamin Erichson

## See Also

rpca, ggbiplot, ggcorplot, ggindplot, ggplot

## Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  set.seed(42)
  fit <- rpca(iris[, 1:4], k = 2, retx = FALSE)
  # The curve uses all processed variance, including omitted components.
  print(ggscreeplot(fit, type = "cum"))
}
```

---

`plot.rpca` *PCA summary plots*

---

## Description

Summarize an object returned by rpca with plots of retained variance and of variables and observations in a component plane. A scree plot shows how much variation the fit captures; factor maps display the corresponding directions and fitted scores.

## Usage

```
## S3 method for class 'rpca'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| x | An object returned by rpca. Factor maps are intended for real-valued loadings and scores. |
| ... | Named arguments for ggscreeplot, ggcorplot, or ggindplot. Each argument is passed only to helpers that accept it. For example, type selects the scree-plot quantity, pcs selects components for both factor maps, and groups is used only for the individual factor map. Names must be exact and unique; unknown or unnamed arguments are rejected. An explicit pcs value must contain two finite integer component indices within the retained range, even when a factor map is unavailable. |

## Details

The function draws the available plots in sequence on the active graphics device. The scree plot shows the variance captured by the retained components. With at least two components, a variable factor map then shows their directions in the selected plane. An individual factor map follows when fitted observation scores are available from retx = TRUE. A fit with one retained component therefore produces only a scree plot; a fit with at least two components and retx = FALSE produces the scree and variable factor plots.

Factor maps use components one and two by default. Supply pcs to inspect another pair of retained components. Any observation groups or labels must correspond to the fitted score rows, after incomplete training observations have been omitted. The **ggplot2** package must be installed. Call the plotting helpers directly when a particular plot is needed for further customization or saving.

## Value

The last **ggplot2** plot drawn, returned invisibly. This is the individual factor map when available, otherwise the variable factor map or the scree plot.

## Author(s)

N. Benjamin Erichson

## See Also

rpca, ggscreeplot, ggcorplot, ggindplot, ggbiplot.

## Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  set.seed(42)
  fit <- rpca(iris[, 1:4], k = 2)
  last_plot <- plot(fit, type = "ratio", pcs = c(1, 2),
                    groups = iris$Species, ind_labels = FALSE,
                    ellipse = FALSE)

  # Request a single plot directly for further customization or saving.
  scree <- ggscreeplot(fit, type = "cum")
}
```

rcur *Randomized CUR matrix decomposition*

## Description

Approximate a matrix using selected columns and rows of the original data, joined by a small matrix of coefficients. The selected data retain their correspondence to measured variables and observations. Randomized column selection is useful when the input has low-rank structure and the requested rank is small.

## Usage

```
rcur(A, k = NULL, p = 10, q = 0, idx_only = FALSE, rand = TRUE)
```

## Arguments

| | |
|---|---|
| A | A real or complex numeric matrix, a numeric data frame, or a matrix from the **Matrix** package. Rows containing missing values are omitted with a warning; the remaining entries must be finite. |
| k | A positive integer giving the number of columns and rows to select. The default, NULL, uses min(dim(A)) after missing rows are omitted. Larger requests are capped at this value. A fractional value of at least one is truncated toward zero. |
| p | A non-negative integer oversampling parameter for randomized column selection. The default is 10. Ignored when rand = FALSE. |
| q | A non-negative integer number of power iterations for randomized column selection. The default is zero. Ignored when rand = FALSE. |
| idx_only | Logical. If TRUE, leave C and R as NULL in the returned object. Their indices and the connector U are still computed and returned. The default is FALSE. |
| rand | Logical. If TRUE (the default), use randomized column selection; otherwise use a deterministic column-pivoted QR decomposition. Row selection from the chosen column matrix is deterministic in both cases. |

## Details

A column interpolative decomposition (ID) represents a matrix using selected columns and coefficients for combining them. CUR also selects rows, so both sides of the approximation contain subsets of the original data. For an $m \times n$ matrix, these factors are combined as

$$A \approx CUR.$$

The m by k matrix C contains actual columns of the input, and the k by n matrix R contains actual rows. These subsets can be examined as measured variables and observations in their original units. The k by k matrix U contains the coefficients that connect the two subsets. Its role differs from that of the orthonormal left singular-vector matrix in a singular value decomposition (SVD).

The algorithm first computes a column interpolative decomposition with rid. It then chooses rows using column-pivoted QR of the conjugate transpose of C. To form the connector, it uses these rows to approximate the interpolation coefficients from the column ID,

$$U = ZR^{+},$$

where Z is the interpolation matrix from the column ID. The numerical pseudoinverse excludes negligible singular values using a threshold that depends on matrix dimensions and floating-point precision. This allows the calculation to proceed when the selected subsets are rank deficient, as can happen when k exceeds the input's numerical rank.

Oversampling and power iterations can improve randomized column selection at the cost of more computation. The supplied settings remain fixed; the algorithm does not adjust them to satisfy an error tolerance. Row selection from the chosen column matrix is deterministic in both modes.

The approximation error depends on the input's spectrum and on which subsets are selected, including their conditioning. Column and row selection does not guarantee the smallest possible rank-k error, so inspect the reconstruction residual when choosing a rank. The QR scores describe the selection procedure and have no interpretation as statistical importance scores.

Centering or rescaling changes the matrix used for selection and can therefore change which variables and observations are chosen. The function uses the supplied data without automatic centering or scaling.

Sparse **Matrix** input is supported. Randomized column selection uses sparse matrix products, but the selected column and row matrices and the connector are dense. With rand = FALSE, column selection also requires a dense input matrix. The option idx_only = TRUE reduces the returned object's size by omitting the selected data. Those data are still used internally to compute U, so this option does not remove their temporary storage. Explicit reconstruction creates a dense matrix.

Set the random seed before a randomized call for a repeatable result in the same computational environment.

## Value

A list of class "rcur" with the following components. Dimensions refer to the input after missing-row omission and the effective k after clamping.

**C** The dense m by k matrix containing columns C.idx of A. It is NULL when idx_only = TRUE.

**R** The dense k by n matrix containing rows R.idx of A. It is NULL when idx_only = TRUE.

**U** The dense k by k connector matrix.

**C.idx** An integer vector of k selected column indices, in selection order.

**R.idx** An integer vector of k selected row indices, in selection order. These index the matrix after rows with missing values have been omitted.

**C.scores** Column QR scores in input column order, as returned by rid. Entries without a QR diagonal are NA.

**R.scores** The k selected-row QR scores, in the same order as R.idx. Scores for unselected input rows are not returned.

**rand** The randomization setting.

## Author(s)

N. Benjamin Erichson

## References

N. B. Erichson, S. Voronin, S. L. Brunton, and J. N. Kutz (2019). Randomized Matrix Decompositions Using R. *Journal of Statistical Software*, 89(11), 1–48. doi:10.18637/jss.v089.i11.

N. Halko, P. G. Martinsson, and J. A. Tropp (2009). Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. https://arxiv.org/abs/0909.4061.

## See Also

rid, rsvd

## Examples

```
set.seed(42)
A <- matrix(rnorm(60 * 5), 60, 5) %*% matrix(rnorm(5 * 35), 5, 35)
A <- A + 0.01 * matrix(rnorm(length(A)), nrow(A), ncol(A))

fit <- rcur(A, k = 5)
A_cur <- fit$C %*% fit$U %*% fit$R
norm(A - A_cur, "F") / norm(A, "F")
fit$C.idx
fit$R.idx

# idx_only still returns the connector, so reconstruction remains possible.
small <- rcur(A, k = 5, idx_only = TRUE, rand = FALSE)
A_small <- A[, small$C.idx, drop = FALSE] %*% small$U %*%
  A[small$R.idx, , drop = FALSE]
norm(A - A_small, "F") / norm(A, "F")
```

---

`rid` *Randomized interpolative decomposition (ID)*

---

## Description

Approximate a matrix by expressing its columns or rows as linear combinations of a selected subset of the original data. The selected data and the coefficients of these combinations form an interpolative decomposition. A randomized sketch can reduce the work needed to choose the subset when the target rank is small.

## Usage

```
rid(A, k = NULL, mode = "column", p = 10, q = 0,
    idx_only = FALSE, rand = TRUE)
```

## Arguments

| | |
|---|---|
| A | A real or complex numeric matrix, a numeric data frame, or a matrix from the **Matrix** package. Rows containing missing values are omitted with a warning; the remaining entries must be finite. |
| k | A positive integer giving the number of columns or rows to select. The default, NULL, uses min(dim(A)) after missing rows are omitted. Larger requests are capped at this value. Use a substantially smaller rank to obtain a compressed approximation. A fractional value of at least one is truncated toward zero. |
| mode | Either "column" (the default) or "row", specifying which type of subset to select. |
| p | A non-negative integer oversampling parameter, used only when rand = TRUE. The default is 10. The sketch width is bounded by the matrix dimensions. |
| q | A non-negative integer number of power iterations, used only when rand = TRUE. The default is zero. |
| idx_only | Logical. If TRUE, leave both selected-data components C and R as NULL. The interpolation matrix Z, indices and scores are still computed and returned. The default is FALSE. |
| rand | Logical. If TRUE (the default), select the subset using a randomized QB sketch. If FALSE, use column-pivoted QR on the full input, or its conjugate transpose for a row ID. |

## Details

An interpolative decomposition (ID) retains actual columns or rows, so the selected subset can be examined in the original units of the data. This is useful when the factors should correspond to measured variables or observations. An orthonormal representation of the data is available through rsvd.

A column ID combines the selected columns with coefficients that approximate each input column. For an $m \times n$ matrix, this gives

$$A \approx CZ.$$

Here C = A[, idx, drop = FALSE] contains k actual columns, and each column of Z gives the coefficients for one input column. Its selected columns satisfy Z[, idx, drop = FALSE] = diag(k), so the selected data columns are reproduced exactly. A row ID applies the same idea to rows and has the form

$$A \approx ZR,$$

R = A[idx, , drop = FALSE] contains the selected rows, and Z[idx, , drop = FALSE] = diag(k). Dimensions and indices in these expressions refer to the input after missing-row omission.

Selection depends on how the data are represented. Centering or rescaling changes the matrix used to choose the subset and can therefore change the selected variables or observations. The function uses the supplied data without automatic centering or scaling.

To reduce the work of subset selection, the randomized algorithm first compresses the input with rqb. It applies column-pivoted QR to the projected factor and selects the first k columns in the resulting pivot order. A row ID applies this procedure to the conjugate transpose, so the pivots identify input rows. Oversampling and power iterations can improve the subspace used for selection, at

additional time and memory cost. The supplied parameters remain fixed throughout the calculation; the algorithm does not adjust them to meet an error tolerance.

The interpolation coefficients are computed using a numerical pseudoinverse of the selected QR block. This permits requests for more columns than the numerical rank, including cases where the selected columns are dependent. Neither column-pivoted QR nor this pseudoinverse guarantees an optimal rank-k approximation or uniformly small coefficients. Assess the reconstruction error to determine whether the selected subset meets the accuracy needs of the application. The QR scores describe the selection procedure and have no interpretation as statistical importance scores or probabilities.

With sparse **Matrix** input, the randomized algorithm forms its dense sketch using sparse matrix products. Returned factors are ordinary dense R matrices. With `rand = FALSE`, the input is converted to a dense matrix before QR. Forming `C %*% Z` or `Z %*% R` also produces a dense reconstruction.

Use `set.seed()` before a randomized call for a repeatable result in the same computational environment. Pivot choices can differ across numerical libraries when columns have similar scores.

## Value

A list of class `"rid"` with the following components. Here `m` is the number of complete input rows and `k` is the effective rank after clamping.

**C** The selected `m` by `k` column matrix for a column ID, or `NULL` for a row ID or when `idx_only = TRUE`.

**R** The selected `k` by `n` row matrix for a row ID, or `NULL` for a column ID or when `idx_only = TRUE`.

**Z** The dense interpolation matrix, with dimensions `k` by `n` for a column ID or `m` by `k` for a row ID.

**idx** An integer vector of length `k` containing the selected column or row indices, in selection order. Row indices refer to the matrix after rows with missing values have been omitted.

**pivot** The full QR column-pivot permutation, or the corresponding row permutation for a row ID. Its first `k` entries are `idx`.

**scores** Absolute diagonal entries of the QR factor, mapped back to input column or row order. Entries without a corresponding QR diagonal are `NA`, which can occur for rectangular inputs or reduced sketches.

**scores.idx** The QR scores of the selected subset, in the same order as `idx`.

**mode, rand** The selection mode and randomization setting.

## Author(s)

N. Benjamin Erichson

## References

N. Halko, P. G. Martinsson, and J. A. Tropp (2009). Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. https://arxiv.org/abs/0909.4061.

N. B. Erichson, S. Voronin, S. L. Brunton, and J. N. Kutz (2019). Randomized Matrix Decompositions Using R. *Journal of Statistical Software*, 89(11), 1–48. doi:10.18637/jss.v089.i11.

## See Also

rcur, rqb, rsvd

## Examples

```
set.seed(42)
A <- matrix(rnorm(60 * 5), 60, 5) %*% matrix(rnorm(5 * 35), 5, 35)
A <- A + 0.01 * matrix(rnorm(length(A)), nrow(A), ncol(A))

# Column ID: all columns are approximated using five original columns.
fit <- rid(A, k = 5)
A_id <- fit$C %*% fit$Z
norm(A - A_id, "F") / norm(A, "F")
fit$idx
fit$Z[, fit$idx, drop = FALSE]

# Row ID has the opposite factor order.
row_fit <- rid(A, k = 5, mode = "row")
norm(A - row_fit$Z %*% row_fit$R, "F") / norm(A, "F")

# Retain indices and coefficients without returning the selected data.
small <- rid(A, k = 5, idx_only = TRUE, rand = FALSE)
is.null(small$C)
dim(small$Z)
```

---

# rpca *Randomized principal component analysis*

---

## Description

Compute principal components of an optionally centered and scaled matrix using a randomized or deterministic singular value decomposition (SVD). Loadings describe the components, scores locate observations in those coordinates, and variances measure the contribution of each component.

## Usage

```
rpca(
  A,
  k = NULL,
  center = TRUE,
  scale = TRUE,
  retx = TRUE,
  p = 10,
  q = 2,
  rand = TRUE
)

## S3 method for class 'rpca'
```

```
print(x, ...)

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

## S3 method for class 'rpca'
predict(object, newdata, ...)
```

## Arguments

| | |
|---|---|
| A | A numeric matrix or data frame with observations in rows and variables in columns. Dense real or complex data and real sparse matrices from **Matrix** are supported. Rows containing missing values are omitted with a warning; all remaining values must be finite. At least two rows must remain. In the dimensions below, $m$ is the number of retained observations and $n$ is the number of variables. |
| k | A positive integer giving the number of principal components to return. A fractional value of at least one is truncated toward zero. Values greater than `min(m, n)` are reduced to that limit. The default, `NULL`, returns `min(m, n)` components using the deterministic SVD. The randomized method is most useful when `k` is much smaller than both matrix dimensions. |
| center | Logical; subtract the training column means before computing the SVD (default `TRUE`). `FALSE` leaves the column means unchanged. |
| scale | Logical; divide each processed column by its scale factor (default `TRUE`). This gives unit sample variance for centered columns whose scale factors are at least `1e-8`; see Details. |
| retx | Logical; return the fitted component scores in `x` (default `TRUE`). With `FALSE`, `x` is `NULL`; loadings and new-data prediction remain available. |
| p | A nonnegative integer oversampling parameter for `rsvd` (default `10`). Ignored for the deterministic SVD. |
| q | A nonnegative integer number of power iterations for `rsvd` (default `2`). Ignored for the deterministic SVD. |
| rand | Logical; use the randomized SVD when `TRUE` and `k` is supplied. With `FALSE`, or `k = NULL`, use the deterministic SVD. Requesting fewer components does not make the deterministic backend a partial SVD algorithm. |
| ... | Additional arguments; currently ignored by the `print`, `summary`, and `predict` methods. |
| object, x | An object returned by `rpca`. |
| newdata | A numeric matrix, data frame, or real sparse **Matrix** containing new observations. Supply the same variables in the same column order as the training data, preferably with no missing values; see Details. |

## Details

Principal components describe data along orthogonal directions in the space of the variables. Each loading vector weights the original variables to specify one direction. Retaining `k` components gives

a low-rank approximation to the processed data. For centered data, the leading components describe directions of greatest sample variance.

With `rand = TRUE` and a supplied `k`, `rpca` uses `rsvd` to approximate the dominant singular subspace. Finding a small subspace first reduces the size of the subsequent SVD, with the greatest savings for large matrices and small requested ranks. Increasing `p` enlarges the sampled subspace, and increasing `q` spends more work refining it to improve accuracy. Use `set.seed` before fitting for reproducible random draws.

**Centering and scaling:** Centering makes the components describe deviations from the training column means. Scaling adjusts the relative influence of variables with different units or magnitudes. Both operations are enabled by default. As in `prcomp`, `center = FALSE` leaves the column means unchanged and accepts either already-centered or uncentered data. Scaling remains enabled unless `scale = FALSE`; `prcomp` disables scaling by default.

With `scale = TRUE`, each column is divided by the square root of its sum of squared magnitudes divided by $m - 1$, after any requested centering. For centered data this is the sample standard deviation. Scale factors below `1e-8` are replaced by one for compatibility, so constant and nearly constant columns are retained. Complex columns use squared magnitudes and therefore have real scale factors.

**Interpreting variance:** The total variance `var` includes every direction in the processed data, including those omitted from the approximation. It is computed as the squared Frobenius norm of the processed matrix divided by $m-1$. For uncentered data this quantity measures squared energy divided by $m-1$ and includes the contribution of the column means. The proportion attributed to a retained component is its `eigvals` entry divided by `var`, so the denominator stays the same as the retained rank changes. At least two complete observations are required; variance proportions are undefined when the processed matrix has zero total variance.

**Using a fit:** Fitted scores are constructed from the SVD as $U_k D_k$. Multiplying them by the conjugate transpose of the loadings reconstructs the approximation in processed variable coordinates. Undo any scaling and centering to express it in the original input coordinates.

`predict(object, newdata)` applies the stored training centering and scaling, then multiplies the processed observations by `object$rotation`. Reusing the training means and scale factors places new observations in the same coordinate system. Supply complete rows in the training column order; the method does not reorder columns by name or omit incomplete new rows automatically. Missing values generally propagate to the scores. For compatibility, `NaN` entries produced or retained during scaling are replaced by zero. Prediction also works when the fit was created with `retx = FALSE`. For a randomized fit, projecting the training data with `predict` can differ from the fitted scores in `object$x` because the randomized subspace omits part of the data, represented by the range-finder residual.

The `print` and `summary` methods describe a fit. `plot.rpca` draws the available scree and factor plots; the individual plotting helpers can also be called directly.

**Sparse input:** Subtracting nonzero column means would fill in many entries of a sparse matrix. With `rand = TRUE` and a supplied `k`, `rpca` instead incorporates centering and scaling into its matrix products for real sparse inputs. This avoids forming the entire centered matrix. Storage includes the sparse entries and dense working factors of width up to `k + p`, capped by the matrix dimensions. Loadings, fitted scores, and predicted scores are dense. The computation estimates ordinary PCA loadings and imposes no sparsity constraint on them.

With `rand = FALSE` or `k = NULL`, a sparse input is converted to a dense matrix for the deterministic SVD.

## Value

`rpca` returns a list of class `"rpca"` with these components:

**rotation** A dense $n$ by $k$ matrix of orthonormal loadings (exact or approximate right singular vectors of the processed data).

**eigvals** A numeric vector of length $k$ containing the retained squared singular values divided by $m - 1$. These estimate the leading eigenvalues of the processed cross-product matrix divided by $m - 1$, which is the sample covariance matrix for centered data.

**sdev** A numeric vector of length $k$ containing the square roots of `eigvals`.

**var** The squared Frobenius norm of the processed matrix divided by $m - 1$, including directions not returned. For centered data this is the total sample variance.

**x** A dense $m$ by $k$ matrix of fitted scores, or `NULL` when `retx = FALSE`. Rows correspond to complete observations retained from the input.

**center, scale** Numeric vectors containing the training column means and scale factors, respectively, or `FALSE` when the corresponding operation was disabled. Centers can be complex; scale factors are real.

The `predict` method returns a dense matrix with one row per new observation and one column per retained component. The `summary` method returns a numeric matrix with rows for explained variance, standard deviation, proportion of variance, and cumulative proportion. Its entries are rounded to three decimal places. The `print` method displays standard deviations, eigenvalues, and loadings, and invisibly returns the displayed loading matrix.

## Note

Singular vectors are defined only up to sign (or a constant of modulus one for complex data). With repeated singular values, any orthonormal basis of the corresponding subspace is valid. Individual loading vectors can therefore differ between runs or implementations without changing the represented subspace.

## Author(s)

N. Benjamin Erichson

## References


- N. B. Erichson, S. Voronin, S. L. Brunton and J. N. Kutz. 2019. Randomized Matrix Decompositions Using R. Journal of Statistical Software, 89(11), 1–48. doi:10.18637/jss.v089.i11.
- N. Halko, P. G. Martinsson and J. A. Tropp. 2011. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. SIAM Review, 53(2), 217–288. doi:10.1137/090771806.


## See Also

`rsvd`, `prcomp`, `plot.rpca`, `ggscreeplot`, `ggindplot`, `ggcorplot`, `rrpca` for a low-rank plus sparse decomposition.

## Examples

```
# Fit two components and project observations held out from the fit.
training <- log(as.matrix(iris[1:145, 1:4]))
new_data <- log(as.matrix(iris[146:150, 1:4]))
set.seed(42)
fit <- rpca(training, k = 2)
summary(fit)
predict(fit, new_data)

# Reconstruct in input coordinates (here, log measurements).
reconstructed <- fit$x %*% Conj(t(fit$rotation))
reconstructed <- sweep(reconstructed, 2, fit$scale, "*")
reconstructed <- sweep(reconstructed, 2, fit$center, "+")

# k = NULL uses the deterministic SVD and returns all components.
full <- rpca(training)
length(full$sdev)

# Sparse input: implicit preprocessing, with training scores omitted.
A <- Matrix::bdiag(outer(1:12, 1:6), outer(1:12, 6:1))
set.seed(42)
sparse_fit <- rpca(A, k = 2, retx = FALSE)
predict(sparse_fit, A[1:2, , drop = FALSE])

if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot(fit, type = "ratio", groups = iris$Species[1:145],
       ind_labels = FALSE, ellipse = FALSE)
}
```

---

# rqb *Randomized QB decomposition*

---

## Description

Project the columns of a matrix onto an orthonormal basis and return the basis and projected coordinates. A basis with few columns gives a smaller matrix for subsequent computations. The basis is found using either randomized range finding or a deterministic QR factorization.

## Usage

```
rqb(A, k = NULL, p = 10, q = 2, sdist = "normal", rand = TRUE)
```

## Arguments

| | |
|---|---|
| A | A real or complex numeric matrix or a numeric data frame. Sparse and dense matrices from the **Matrix** package are supported; see Details. Other matrix-like inputs are converted with `as.matrix`. The input must have positive dimensions and finite entries. |

k — A positive integer giving the target rank before randomized oversampling. Requests larger than `min(dim(A))` are capped at that value. The default, `NULL`, requests the full available basis width. Use a rank much smaller than the matrix dimensions for the main computational savings.

p — A non-negative integer oversampling parameter for the randomized algorithm. The default is 10. The oversampled basis is retained in the result; see Details for its relationship to `rsvd`. This does not change the basis width when `rand = FALSE`.

q — A non-negative integer giving the number of randomized power (subspace) iterations. The default is 2; zero skips these iterations. Unused when `rand = FALSE`.

sdist — A character string selecting the sampling distribution. `"normal"` (the default) uses standard normal entries, `"unif"` uses uniform entries on $[0, 1]$, and `"rademacher"` uses equally likely values of -1 and 1. For complex input, the real and imaginary parts are sampled independently. Unused when `rand = FALSE`.

rand — A logical value. If `TRUE` (the default), use randomized range finding; otherwise use the deterministic QR basis described in Details.

## Details

The QB decomposition uses the same set of orthonormal vectors to approximate every column of `A`. These vectors form `Q`, and each column of `B` gives the coordinates of an input column projected onto their span. Their product gives the approximation

$$A \approx QB, \qquad B = Q^H A.$$

The superscript $H$ denotes the conjugate transpose, which is the ordinary transpose for real input. Thus `Q %*% B` retains the part of each column of `A` that lies in the column space of `Q`. This is an orthogonal projection. The input is used without centering or scaling.

When `rand = TRUE`, random matrix products identify a candidate subspace. The algorithm refines it with `q` power iterations and intermediate orthogonalization before projecting the input. The basis width is $\ell = \min(k + p, m, n)$ for integer `k` and `p`, with `m` rows and `n` columns in the input. The oversampled columns are retained, so the result generally has `k + p` basis vectors. A singular value decomposition (SVD) of the projected matrix `B` can then identify its leading directions and truncate the approximation further. The function `rsvd` follows this approach and returns `k` singular triplets with its default vector counts.

Oversampling gives the sampled subspace additional directions, and power iterations can improve how well it captures the input. Both can improve approximation quality, especially when singular values decay slowly, at the cost of more computation. The parameters remain fixed throughout the calculation; there is no adaptive error tolerance or automatic rank selection. The useful rank depends on how much information the application needs to retain. Even a matrix with full algebraic rank may be well represented in a much smaller subspace.

When `rand = FALSE`, the basis width is $\ell = \min(k, m, n)$. The result uses the first $\ell$ columns of `qr.Q(qr(as.matrix(A)))`, with base R's default QR conventions. For real input whose leading block is numerically independent, only that block needs dense factorization. Complex input, dependent leading columns and other cases use the full dense QR fallback. This preserves base QR's ordering, including its pivoting behavior. The resulting subspace follows that ordering and need not minimize the rank-`k` approximation error.

For randomized decompositions, real-valued sparse **Matrix** inputs are multiplied by thin dense factors without converting the entire input to a dense matrix. Logical and pattern matrices are treated as zero/one data. Complex input is supported for ordinary R matrices. Samples, orthonormal bases and projected matrices are dense. The deterministic fallback can convert the entire input to dense storage, which limits its usefulness for large sparse matrices. It does not provide a general sparse QR implementation. Constructing `Q %*% B` also forms a dense matrix.

Setting `k = NULL` uses the full available basis width and gives up the storage advantage of a small basis. The function does not reduce that width based on an estimated numerical rank.

## Value

A list of class `"rqb"` containing two ordinary R matrices:

**Q** An $m \times \ell$ matrix with orthonormal columns.

**B** An $\ell \times n$ matrix containing the projection $Q^H A$.

The input dimensions are $m \times n$. The basis width $\ell$ is defined in Details and is not automatically reduced to the numerical rank.

## Note

Different bases can describe the same subspace. Signs, complex phases or rotations within that subspace can change `Q` and `B` while preserving their product. Assess agreement through the reconstruction or subspace, since individual basis entries can differ between valid results.

Use `set.seed` for reproducible random sampling within a fixed numerical environment. The deterministic branch does not draw random numbers.

## Author(s)

N. Benjamin Erichson

## References


- N. B. Erichson, S. Voronin, S. L. Brunton and J. N. Kutz. 2019. Randomized Matrix Decompositions Using R. Journal of Statistical Software, 89(11), 1-48. doi:10.18637/jss.v089.i11.
- N. Halko, P. Martinsson and J. Tropp. 2009. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. https://arxiv.org/abs/0909.4061.


## See Also

`rsvd`, `svd`, `qr`

## Examples

```
# An exact rank-three signal.
set.seed(42)
A <- matrix(rnorm(60 * 3), 60, 3) %*% matrix(rnorm(3 * 40), 3, 40)
qb <- rqb(A, k = 3, p = 2, q = 1)
dim(qb$Q)  # 60 by 5: randomized QB retains the two oversampling columns.
```

```
norm(A - qb$Q %*% qb$B, "F") / norm(A, "F")

# Further truncation is performed in the projected matrix.
small <- svd(qb$B, nu = 3, nv = 3)
U <- qb$Q %*% small$u
A_approx <- sweep(U, 2, small$d[1:3], "*") %*% Conj(t(small$v))
norm(A - A_approx, "F") / norm(A, "F")

# The deterministic branch returns k columns, without oversampling.
qb_deterministic <- rqb(A, k = 3, rand = FALSE)
dim(qb_deterministic$Q)
norm(A - qb_deterministic$Q %*% qb_deterministic$B, "F") / norm(A, "F")

# Sparse randomized QB still returns ordinary dense factors.
S <- Matrix::Diagonal(x = c(9, 4, 1, rep(0, 17)))
qb_sparse <- rqb(S, k = 3, p = 2, q = 1)
dim(qb_sparse$B)
```

---

rrpca *Randomized robust principal component analysis*

---

## Description

Estimate a low-rank component and an entrywise sparse component whose sum reconstructs a real matrix. Principal component pursuit uses penalties to encourage this separation, with randomized or deterministic singular value decompositions for the low-rank updates.

## Usage

```
rrpca(
  A,
  lambda = NULL,
  maxiter = 50,
  tol = 1e-05,
  p = 10,
  q = 2,
  trace = FALSE,
  rand = TRUE
)
```

## Arguments

| | |
|---|---|
| A | A real numeric matrix or data frame to decompose. A sparse matrix from **Matrix** is accepted and converted to a dense matrix. Rows containing missing values are omitted with a warning; all remaining entries must be finite. Complex input is not supported. In the dimensions below, $m$ and $n$ are the numbers of rows and columns after omission; both must be positive. |

| | |
|---|---|
| `lambda` | A positive numeric penalty on the entrywise sparse component. The default, `NULL`, uses `1 / sqrt(max(m, n))`. Larger values penalize the sparse component more strongly. |
| `maxiter` | A positive integer maximum number of iterations (default `50`). Reaching this limit does not imply convergence. |
| `tol` | A positive numeric tolerance for both the relative reconstruction residual and stationarity diagnostic (default `1e-5`). |
| `p` | A nonnegative integer oversampling parameter for `rsvd` (default `10`); used only in randomized SVD steps. |
| `q` | A nonnegative integer initial number of power iterations for `rsvd` (default `2`); increased adaptively if randomized accuracy stalls. Ignored when `rand = FALSE`. |
| `trace` | Logical; print iteration progress, including both convergence diagnostics (default `FALSE`). |
| `rand` | Logical; start with randomized SVD steps when `TRUE` (default), with deterministic SVD used for large target ranks or stalled accuracy. With `FALSE`, use the deterministic SVD throughout. |

## Details

A matrix can contain patterns shared across many entries together with corruption confined to a few entries. For example, frames of a nearly static video can share a low-rank background while a small moving object affects only a subset of the pixels in each frame. Principal component pursuit seeks a low-rank term $L$ and a sparse term $S$ satisfying $A = L + S$.

Among such splits, the objective minimizes the sum of singular values of $L$ plus `lambda` times the sum of absolute entries of $S$. These penalties are called the nuclear norm and the entrywise one-norm, respectively. They encourage low rank in $L$ and few nonzero entries in $S$. Successful denoising depends on the data and the assumptions that make low-rank structure distinguishable from sparse corruption.

**Computation:** The inexact augmented Lagrange multiplier method (IALM) first updates the sparse component by shrinking entries toward zero and setting small entries to zero. It then applies the same idea to singular values to update the low-rank component. These steps are entrywise soft thresholding and singular-value thresholding. The surviving singular values determine the rank, so the function selects it during iteration.

Randomized singular-value thresholding increases its target rank if all returned singular values exceed the shrinkage threshold, allowing it to examine further directions that might survive thresholding. Large target ranks use deterministic SVD. If accuracy stalls, the algorithm makes bounded increases in the number of power iterations before falling back to deterministic SVD. These refinements can increase runtime.

Input and working arrays are dense, including for sparse input. Apply any desired centering or scaling before calling `rrpca`; the function uses the supplied values without this preprocessing. Use `set.seed` before fitting for reproducible random draws when `rand = TRUE`.

**Stopping diagnostics:** Many different splits can reconstruct the input, so reconstruction accuracy alone is insufficient to establish convergence. The algorithm also monitors the change in the low-rank component.

The relative reconstruction residual combines errors across all entries and compares their Frobenius norm with that of the input,

$$\frac{\|A - L - S\|_F}{\|A\|_F}.$$

A small value means that $L + S$ closely reproduces $A$ relative to its magnitude. With the sparse component updated first, the stationarity diagnostic measures the change in $L$ scaled by the current penalty,

$$\frac{\mu\|L_{new} - L_{old}\|_F}{\max(1, \|Z\|_F)}.$$

Here $Z$ is the updated Lagrange multiplier and $\mu$ is the penalty used for that iteration. Both diagnostics must be at most `tol` for the stopping test to be satisfied. The penalty increases only when the stationarity diagnostic is at most `tol`. These normalized tests are invariant to rescaling the input.

The iteration limit `maxiter` may be reached before both diagnostics satisfy `tol`. The stopping diagnostics and randomized approximations provide no general certificate of optimality. Use `trace = TRUE` to display both diagnostics during fitting. The returned `err` records reconstruction residuals only; a small final value cannot establish convergence on its own.

**Using the decomposition:** `rrpca` fits all entries of the supplied matrix jointly. Its `L` and `S` describe those observations, and a fitted PCA rotation or a `predict.rrpca` method is unavailable. Projecting new rows onto a basis extracted from `L` gives an ordinary low-rank projection with a generally dense residual. Estimating a robust sparse component for those new rows with a fixed learned basis requires a separate optimization problem. For ordinary PCA with new-data scores, use `rpca` and its `predict` method.

## Value

A list of class `"rrpca"` with the following components.

**L** A dense $m$ by $n$ matrix containing the estimated low-rank component.

**S** A dense $m$ by $n$ matrix containing the estimated sparse component. It is stored densely even when most entries are zero.

**err** A numeric vector of relative Frobenius reconstruction residuals, one per completed iteration. For a zero input, the function returns zero components and `err = 0` immediately.

Rows of `L` and `S` correspond to retained input rows after missing rows have been omitted. No fitted loading basis or new-data prediction method is returned.

## Author(s)

N. Benjamin Erichson

## References


- N. B. Erichson, S. Voronin, S. L. Brunton and J. N. Kutz. 2019. Randomized Matrix Decompositions Using R. Journal of Statistical Software, 89(11), 1–48. doi:10.18637/jss.v089.i11.
- Z. Lin, M. Chen and Y. Ma. 2010. The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices. https://arxiv.org/abs/1009.5055.

- S. Boyd, N. Parikh, E. Chu, B. Peleato and J. Eckstein. 2011. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. Foundations and Trends in Machine Learning, 3(1), 1–122. doi:10.1561/2200000016.

## See Also

rpca, rsvd.

## Examples

```
# A small video: each column is one frame, with pixels stored in rows.
background <- outer(1:32, 1:32, function(x, y) 0.1 + 0.001 * (x + y))
video <- matrix(rep(as.vector(background), 30), nrow = 32 * 32)
for (i in 1:30) {
  foreground <- matrix(0, 32, 32)
  foreground[i:(i + 2), 15:17] <- 0.8
  video[, i] <- video[, i] + as.vector(foreground)
}

set.seed(42)
out <- rrpca(video, maxiter = 100, tol = 1e-6)
tail(out$err, 1) # Reconstruction residual; see Details for convergence.

# Compare one frame and its estimated background and foreground.
# Each panel uses its own intensity range.
old_par <- par(mfrow = c(1, 4), mar = c(0, 0, 2, 0))
image(background, col = gray.colors(64), axes = FALSE,
      asp = 1, useRaster = TRUE, main = "Background")
image(matrix(video[, 10], 32, 32), col = gray.colors(64),
      axes = FALSE, asp = 1, useRaster = TRUE, main = "Frame")
image(matrix(out$L[, 10], 32, 32), col = gray.colors(64),
      axes = FALSE, asp = 1, useRaster = TRUE, main = "Low-rank part")
image(matrix(out$S[, 10], 32, 32), col = gray.colors(64),
      axes = FALSE, asp = 1, useRaster = TRUE, main = "Sparse part")
par(old_par)
```

---

# rsvd *Randomized singular value decomposition*

---

## Description

Approximate the leading singular values and singular vectors of a matrix. Random matrix products identify a subspace in which the singular value decomposition (SVD) is computed. This can reduce the factorization work when the requested rank is small relative to the matrix dimensions.

## Usage

```
rsvd(A, k = NULL, nu = NULL, nv = NULL, p = 10, q = 2, sdist = "normal")
```

## Arguments

| | |
|---|---|
| A | A real or complex numeric matrix or a numeric data frame. Sparse and dense matrices from the **Matrix** package are supported; see Details. The input must have positive dimensions and finite entries. |
| k | A positive integer giving the target rank. The default, NULL, uses the smaller matrix dimension; larger requests are capped at that value. Use a rank much smaller than both dimensions for the main computational savings. |
| nu | An integer giving the number of left singular vectors to return. The default, NULL, returns k vectors. Integer requests are clamped to the range from zero to the effective target rank; zero omits u. |
| nv | An integer giving the number of right singular vectors to return. The default, NULL, returns k vectors. Integer requests are clamped to the range from zero to the effective target rank; zero omits v. |
| p | A non-negative integer oversampling parameter. The default is 10. The sketch uses up to k + p columns, bounded by the matrix dimensions. |
| q | A non-negative integer giving the number of power (subspace) iterations. The default is 2; zero skips the power iterations. |
| sdist | A character string selecting the sampling distribution. "normal" (the default) uses standard normal entries, "unif" uses uniform entries on $[0, 1]$, and "rademacher" uses equally likely values of -1 and 1. For complex input, the real and imaginary parts are sampled independently. |

## Details

A low-rank approximation represents the columns of a matrix using a small set of shared directions. When these directions capture the structure needed for an analysis, storing and working with the factors can save both memory and computation. For an $m \times n$ input matrix $A$, the rank-$k$ reconstruction has the form

$$A \approx U_k D_k V_k^H.$$

The columns of $U_k$ and $V_k$ are orthonormal and the diagonal of $D_k$ contains the returned singular values. Each pair of left and right vectors contributes one matrix component, weighted by its singular value. The superscript $H$ denotes the conjugate transpose, which is the ordinary transpose for real data. Reconstruction requires both sets of vectors and uses Conj(t(v)) for complex input.

To find these directions, the algorithm samples a subspace of width $\ell = \min(k+p, m, n)$. It refines the sample with q power iterations and intermediate orthogonalization, then computes an SVD of the matrix projected onto that subspace. The large input is accessed through matrix products. When the subspace is small, the final SVD acts on a smaller dense matrix.

Oversampling gives the algorithm extra directions from which to identify the dominant rank-k subspace; only k singular values are returned. Increasing p or q can improve accuracy, at the cost of more computation and, for p, more intermediate storage. The function uses the supplied parameters throughout. It does not adapt them to an error tolerance or estimate a numerical rank.

The accuracy achievable at a given rank depends on the singular values of the input. A full-rank matrix can have a useful low-rank approximation when these values decay rapidly. With a flat

spectrum and a small target rank, even an optimal approximation leaves a large residual. If $\sigma_j$ are the singular values of the original matrix in decreasing order, its optimal rank-$k$ Frobenius error is

$$\left(\sum_{j>k} \sigma_j^2\right)^{1/2}.$$

This expression measures the contribution of the discarded components. It is zero when k reaches the full available rank. Use this truncation error as a reference for the randomized reconstruction residual, with the same preprocessing and target rank for both approximations. An arbitrary full-rank matrix cannot in general be recovered exactly at a smaller rank.

The decomposition acts directly on the supplied matrix, without centering or scaling. See rpca for principal component analysis with those preprocessing options.

Real-valued sparse **Matrix** inputs are multiplied by thin dense factors without converting the entire input to a dense matrix. Logical and pattern matrices are treated as zero/one data. Complex input is supported for ordinary R matrices. Dense storage is still needed for the random samples, intermediate bases, projected matrices and returned singular vectors. Explicitly reconstructing the approximation also creates a dense matrix.

Omitting k requests all available singular values, even if nu or nv requests only a few vectors. The resulting subspace no longer provides the storage savings of a small-rank approximation. When the target rank is a substantial fraction of the matrix dimensions, a deterministic method may be faster. Use svd as a reference for a full SVD.

## Value

A list of class "rsvd" with the following components. Here k denotes the target rank after clamping to the matrix dimensions.

**d** A numeric vector of length k containing the approximate singular values in decreasing order. The values are non-negative.

**u** An ordinary R matrix with nrow(A) rows and nu orthonormal columns, after applying the default and clamping rules above. Omitted when no left singular vectors are requested.

**v** An ordinary R matrix with ncol(A) rows and nv orthonormal columns, after applying the default and clamping rules above. Omitted when no right singular vectors are requested.

## Note

Singular vectors are defined only up to sign, or a constant of modulus one for complex data; multiplying a corresponding left and right vector by the same factor leaves the reconstruction unchanged. Repeated singular values also allow rotations within their singular subspace, and closely spaced values can make individual directions sensitive. Compare reconstruction residuals or subspaces, since different valid singular vectors can represent the same approximation.

Use set.seed for reproducibility within a fixed numerical environment; identical vectors are not guaranteed across BLAS or package versions.

## Author(s)

N. Benjamin Erichson

## References


- N. B. Erichson, S. Voronin, S. L. Brunton and J. N. Kutz. 2019. Randomized Matrix Decompositions Using R. Journal of Statistical Software, 89(11), 1-48. doi:10.18637/jss.v089.i11.
- N. Halko, P. Martinsson and J. Tropp. 2009. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. https://arxiv.org/abs/0909.4061.


## See Also

svd, rqb, rpca

## Examples

```
# A low-rank signal with small dense noise.
set.seed(42)
A <- matrix(rnorm(60 * 5), 60, 5) %*% matrix(rnorm(5 * 40), 5, 40)
A <- A + 0.01 * matrix(rnorm(60 * 40), 60, 40)
k <- 5L
fit <- rsvd(A, k = k, p = 5, q = 2)
A_approx <- sweep(fit$u, 2, fit$d, "*") %*% Conj(t(fit$v))

# Compare residuals at the same target rank on this small matrix.
d <- svd(A, nu = 0, nv = 0)$d
best <- sqrt(sum(d[-seq_len(k)]^2))
c(randomized = norm(A - A_approx, "F"), optimal = best)

# Complex data require a conjugate transpose for reconstruction.
Z <- A + 1i * A[, rev(seq_len(ncol(A)))]
fit_complex <- rsvd(Z, k = k, p = 5, q = 2)
Z_approx <- sweep(fit_complex$u, 2, fit_complex$d, "*") %*%
  Conj(t(fit_complex$v))
sqrt(sum(Mod(Z - Z_approx)^2) / sum(Mod(Z)^2))

# Sparse input stays sparse; the returned factors are dense.
S <- Matrix::Diagonal(x = c(9, 4, 1, rep(0, 17)))
fit_sparse <- rsvd(S, k = 3, p = 2, q = 1)
fit_sparse$d
```

---

# tiger *Tiger image*

---

## Description

A grayscale tiger photograph for examining how low-rank approximation changes image detail. The image is stored as a matrix, so a factorization and its reconstruction can be compared directly with the original pixels. Intensities have been divided by 255 to lie between zero and one.

## Usage

```
data("tiger")
```

## Format

A numeric matrix with 1600 rows and 1200 columns. There are no labels or color channels; each entry is a grayscale intensity.

## Details

The stored orientation is suitable for `image` as shown below. Matrix rows and columns correspond to the horizontal and vertical axes used by that function, respectively. For a decomposition, use `tiger` directly and reconstruct the matrix in the same orientation.

The photograph is not exactly low rank. A low-rank factorization retains part of its structure, and increasing the approximation rank preserves more texture and fine detail. Storing only the factors can require much less space than storing all pixels. See `rsvd`, `rid` and `rcur` for decomposition examples.

## Source

Wikimedia image and attribution

## References

S. Taheri (2006), edited by Fir0002. Panthera tigris altaica. Online photograph, distributed under Creative Commons Attribution-ShareAlike 2.5. The package supplies a grayscale conversion.

## Examples

```
data("tiger")
dim(tiger)
range(tiger)
image(seq_len(nrow(tiger)), seq_len(ncol(tiger)), tiger,
      col = gray((0:255) / 255), asp = 1,
      axes = FALSE, useRaster = TRUE)
```

# Index

∗ **datasets**
digits, 2
tiger, 30

digits, 2

ggbiplot, 3, *7*, *9–11*
ggcorplot, *4, 5*, 5, *9–11*, *20*
ggindplot, *5*, *7*, 7, *10, 11*, *20*
ggplot, *5*, *7*, *9, 10*
ggscreeplot, *5*, *7*, *9*, 9, *11*, *20*

image, *2*, *31*

plot.rpca, 10, *19, 20*
prcomp, *19, 20*
predict.rpca *(*rpca*)*, 17
print.rpca *(*rpca*)*, 17

qr, *23*

rcur, 12, *17*, *31*
rid, *13, 14*, 14, *31*
rpca, *3*, *5–11*, 17, *26, 27*, *29, 30*
rqb, *15*, *17*, 21, *30*
rrpca, *20*, 24
rsvd, *14, 15*, *17–20*, *22, 23*, *25*, *27*, 27, *31*

stat_ellipse, *4, 5*, *8, 9*
summary.rpca *(*rpca*)*, 17
svd, *23*, *29, 30*

tiger, 30