Package: PCAtools (via r-universe)

October 29, 2026

Type Package

Title PCAtools: Everything Principal Components Analysis

Version 2.24.0

Description Principal Component Analysis (PCA) is a very powerful technique that has wide applicability in data science, bioinformatics, and further afield. It was initially developed to analyse large volumes of data in order to tease out the differences/relationships between the logical entities being analysed. It extracts the fundamental structure of the data without the need to build any model to represent it. This 'summary' of the data is arrived at through a process of reduction that can transform the large number of variables into a lesser number that are uncorrelated (i.e. the 'principal components'), while at the same time being capable of easy interpretation on the original data. PCAtools provides functions for data exploration via PCA, and allows the user to generate publication-ready figures. PCA is performed via BiocSingular - users can also identify optimal number of principal components via different metrics, such as elbow method and Horn's parallel analysis, which has relevance for data reduction in single-cell RNA-seq (scRNA-seq) and high dimensional mass cytometry data.

License GPL-3

Depends ggplot2, ggrepel

Imports lattice, grDevices, cowplot, methods, reshape2, stats, Matrix, DelayedMatrixStats, DelayedArray, beachmat (>= 2.21.1), BiocSingular, BiocParallel, Rcpp, dqrng

Suggests testthat, scran, BiocGenerics, knitr, Biobase, GEOquery, hgu133a.db, ggplotify, RMTstat, ggforce, concaveman, DESeq2, airway, org.Hs.eg.db, magrittr, rmarkdown, airway

LinkingTo Rcpp, beachmat, assorthead, BH, dqrng

URL https://github.com/kevinblighe/PCAtools

biocViews RNASeq, ATACSeq, GeneExpression, Transcription, SingleCell, PrincipalComponent

VignetteBuilder knitr

SystemRequirements C++17

RoxygenNote 7.3.3

Encoding UTF-8

Config/pak/sysreqs libicu-dev zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 12:49:22 UTC

RemoteUrl https://github.com/bioc/PCAtools

RemoteRef RELEASE_3_23

RemoteSha 6270c94610c8de0e2c81d01a3d708f382dab4be3

Contents

biplot . . . 2
chooseGavishDonoho . . . 9
chooseMarchenkoPastur . . . 11
eigencorplot . . . 12
findElbowPoint . . . 15
getComponents . . . 16
getLoadings . . . 17
getVars . . . 19
pairsplot . . . 20
parallelPCA . . . 24
pca . . . 26
plotloadings . . . 28
screeplot . . . 32

Index **37**

`biplot` *Draw a bi-plot, comparing 2 selected principal components / eigenvectors.*

Description

Draw a bi-plot, comparing 2 selected principal components / eigenvectors.

Usage

```
biplot(
  pcaobj,
  x = "PC1",
  y = "PC2",
  showLoadings = FALSE,
  ntopLoadings = 5,
  LoadingsNames = NULL,
  showLoadingsNames = if (showLoadings) TRUE else FALSE,
  colLoadingsNames = "black",
  sizeLoadingsNames = 3,
  boxedLoadingsNames = TRUE,
  fillBoxedLoadings = alpha("white", 1/4),
  drawConnectorsLoadings = TRUE,
  widthConnectorsLoadings = 0.5,
  colConnectorsLoadings = "grey50",
  lengthLoadingsArrowsFactor = 1.5,
  colLoadingsArrows = "black",
  widthLoadingsArrows = 0.5,
  alphaLoadingsArrow = 1,
  colby = NULL,
  colkey = NULL,
  colLegendTitle = if (!is.null(colby)) colby else NULL,
  singlecol = NULL,
  shape = NULL,
  shapekey = NULL,
  shapeLegendTitle = if (!is.null(shape)) shape else NULL,
  pointSize = 3,
  legendPosition = "none",
  legendLabSize = 12,
  legendTitleSize = 14,
  legendIconSize = 5,
  encircle = FALSE,
  encircleFill = TRUE,
  encircleFillKey = NULL,
  encircleAlpha = 1/4,
  encircleLineSize = 0.25,
  encircleLineCol = NULL,
  ellipse = FALSE,
  ellipseType = "t",
  ellipseLevel = 0.95,
  ellipseSegments = 51,
  ellipseFill = TRUE,
  ellipseFillKey = NULL,
  ellipseAlpha = 1/4,
  ellipseLineSize = 0.25,
  ellipseLineCol = NULL,
  xlim = if (showLoadings || ellipse) c(min(pcaobj$rotated[, x]) -
```

```
  abs((min(pcaobj$rotated[, x])/100) * 35), max(pcaobj$rotated[, x]) +
  abs((min(pcaobj$rotated[, x])/100) * 35)) else c(min(pcaobj$rotated[, x]) -
  abs((min(pcaobj$rotated[, x])/100) * 10), max(pcaobj$rotated[, x]) +
  abs((min(pcaobj$rotated[, x])/100) * 10)),
ylim = if (showLoadings || ellipse) c(min(pcaobj$rotated[, y]) -
  abs((min(pcaobj$rotated[, y])/100) * 35), max(pcaobj$rotated[, y]) +
  abs((min(pcaobj$rotated[, y])/100) * 35)) else c(min(pcaobj$rotated[, y]) -
  abs((min(pcaobj$rotated[, y])/100) * 10), max(pcaobj$rotated[, y]) +
  abs((min(pcaobj$rotated[, y])/100) * 10)),
lab = rownames(pcaobj$metadata),
labSize = 3,
boxedLabels = FALSE,
selectLab = NULL,
drawConnectors = TRUE,
widthConnectors = 0.5,
colConnectors = "grey50",
max.overlaps = 15,
maxoverlapsConnectors = NULL,
min.segment.length = 0,
directionConnectors = "both",
xlab = paste0(x, ", ", round(pcaobj$variance[x], digits = 2), "% variation"),
xlabAngle = 0,
xlabhjust = 0.5,
xlabvjust = 0.5,
ylab = paste0(y, ", ", round(pcaobj$variance[y], digits = 2), "% variation"),
ylabAngle = 0,
ylabhjust = 0.5,
ylabvjust = 0.5,
axisLabSize = 16,
title = "",
subtitle = "",
caption = "",
titleLabSize = 16,
subtitleLabSize = 12,
captionLabSize = 12,
hline = NULL,
hlineType = "longdash",
hlineCol = "black",
hlineWidth = 0.4,
vline = NULL,
vlineType = "longdash",
vlineCol = "black",
vlineWidth = 0.4,
gridlines.major = TRUE,
gridlines.minor = TRUE,
borderWidth = 0.8,
borderColour = "black",
returnPlot = TRUE,
```

```
  flip_axes = FALSE
)
```

Arguments

pcaobj	Object of class 'pca' created by pca().
x	A principal component to plot on x-axis. All principal component names are stored in pcaobj$label.
y	A principal component to plot on y-axis. All principal component names are stored in pcaobj$label.
showLoadings	Logical, indicating whether or not to overlay variable loadings.
ntopLoadings	If showLoadings == TRUE, select this many variables based on absolute ordered variable loading for each PC in the biplot. As a result of looking across 2 PCs, it can occur whereby greater than this number are actually displayed.
LoadingsNames	Names of variable loadings to display. If provided, this overrides 'ntopLoadings'.
showLoadingsNames	Logical, indicating to show variable loadings names or not.
colLoadingsNames	If 'showLoadings == TRUE', colour of text labels.
sizeLoadingsNames	If 'showLoadings == TRUE', size of text labels.
boxedLoadingsNames	Logical, if 'showLoadings == TRUE', draw text labels in boxes.
fillBoxedLoadings	When 'boxedLoadingsNames == TRUE', this controls the background fill of the boxes. To control both the fill and transparency, user can specify a value of the form 'alpha(<colour>, <alpha>)'.
drawConnectorsLoadings	If 'showLoadings == TRUE', draw line connectors to the variable loadings arrows in order to fit more labels in the plot space.
widthConnectorsLoadings	If 'showLoadings == TRUE', width of the line connectors drawn to the variable loadings arrows.
colConnectorsLoadings	If 'showLoadings == TRUE', colour of the line connectors drawn to the variable loadings arrows.
lengthLoadingsArrowsFactor	If 'showLoadings == TRUE', multiply the internally-determined length of the variable loadings arrows by this factor.
colLoadingsArrows	If showLoadings == TRUE, colour of the variable loadings arrows.
widthLoadingsArrows	If showLoadings == TRUE, width of the variable loadings arrows.

alphaLoadingsArrow
: If showLoadings == TRUE, colour transparency of the variable loadings arrows.

colby
: If NULL, all points will be coloured differently. If not NULL, value is assumed to be a column name in pcaobj$metadata relating to some grouping/categorical variable.

colkey
: Vector of name-value pairs relating to value passed to 'col', e.g., c(A='forestgreen', B='gold').

colLegendTitle
: Title of the legend for the variable specified by 'colby'.

singlecol
: If specified, all points will be shaded by this colour. Overrides 'col'.

shape
: If NULL, all points will be have the same shape. If not NULL, value is assumed to be a column name in pcaobj$metadata relating to some grouping/categorical variable.

shapekey
: Vector of name-value pairs relating to value passed to 'shape', e.g., c(A=10, B=21).

shapeLegendTitle
: Title of the legend for the variable specified by 'shape'.

pointSize
: Size of plotted points.

legendPosition
: Position of legend ('top', 'bottom', 'left', 'right', 'none').

legendLabSize
: Size of plot legend text.

legendTitleSize
: Size of plot legend title text.

legendIconSize
: Size of plot legend icons / symbols.

encircle
: Logical, indicating whether to draw a polygon around the groups specified by 'colby'.

encircleFill
: Logical, if 'encircle == TRUE', this determines whether to fill the encircled region or not.

encircleFillKey
: Vector of name-value pairs relating to value passed to 'encircleFill', e.g., c(A='forestgreen', B='gold'). If NULL, the fill is controlled by whatever has already been used for 'colby' / 'colkey'.

encircleAlpha
: Alpha for purposes of controlling colour transparency of the encircled region. Used when 'encircle == TRUE'.

encircleLineSize
: Line width of the encircled line when 'encircle == TRUE'.

encircleLineCol
: Colour of the encircled line when 'encircle == TRUE'.

ellipse
: Logical, indicating whether to draw a data ellipse around the groups specified by 'colby'.

ellipseType
: [paraphrased from https://ggplot2.tidyverse.org/reference/stat_ellipse.html] The type of ellipse. "t" assumes a multivariate t-distribution, while "norm" assumes a multivariate normal distribution. "euclid" draws a circle with the radius equal to level, representing the euclidean distance from the center. This ellipse probably won't appear circular unless coord_fixed() is applied.

`ellipseLevel` [paraphrased from https://ggplot2.tidyverse.org/reference/stat_ellipse.html] The level at which to draw an ellipse, or, if ellipseType="euclid", the radius of the circle to be drawn.

`ellipseSegments` [from https://ggplot2.tidyverse.org/reference/stat_ellipse.html] The number of segments to be used in drawing the ellipse.

`ellipseFill` Logical, if 'ellipse == TRUE', this determines whether to fill the region or not.

`ellipseFillKey` Vector of name-value pairs relating to value passed to 'ellipseFill', e.g., c(A='forestgreen', B='gold'). If NULL, the fill is controlled by whatever has already been used for 'colby' / 'colkey'.

`ellipseAlpha` Alpha for purposes of controlling colour transparency of the ellipse region. Used when 'ellipse == TRUE'.

`ellipseLineSize` Line width of the ellipse line when 'ellipse == TRUE'.

`ellipseLineCol` Colour of the ellipse line when 'ellipse == TRUE'.

`xlim` Limits of the x-axis.

`ylim` Limits of the y-axis.

`lab` A vector containing labels to add to the plot.

`labSize` Size of labels.

`boxedLabels` Logical, draw text labels in boxes.

`selectLab` A vector containing a subset of lab to plot.

`drawConnectors` Logical, indicating whether or not to connect plot labels to their corresponding points by line connectors.

`widthConnectors` Line width of connectors.

`colConnectors` Line colour of connectors.

`max.overlaps` Equivalent of max.overlaps in ggrepel. Set to 'Inf' to always display all labels when drawConnectors = TRUE.

`maxoverlapsConnectors` See max.overlaps.

`min.segment.length` When drawConnectors = TRUE, specifies the minimum length of the connector line segments.

`directionConnectors` direction in which to draw connectors. 'both', 'x', or 'y'.

`xlab` Label for x-axis.

`xlabAngle` Rotation angle of x-axis labels.

`xlabhjust` Horizontal adjustment of x-axis labels.

`xlabvjust` Vertical adjustment of x-axis labels.

`ylab` Label for y-axis.

`ylabAngle` Rotation angle of y-axis labels.

ylabhjust	Horizontal adjustment of y-axis labels.
ylabvjust	Vertical adjustment of y-axis labels.
axisLabSize	Size of x- and y-axis labels.
title	Plot title.
subtitle	Plot subtitle.
caption	Plot caption.
titleLabSize	Size of plot title.
subtitleLabSize	Size of plot subtitle.
captionLabSize	Size of plot caption.
hline	Draw one or more horizontal lines passing through this/these values on y-axis. For single values, only a single numerical value is necessary. For multiple lines, pass these as a vector, e.g., c(60,90).
hlineType	Line type for hline ('blank', 'solid', 'dashed', 'dotted', 'dotdash', 'longdash', 'twodash').
hlineCol	Colour of hline.
hlineWidth	Width of hline.
vline	Draw one or more vertical lines passing through this/these values on x-axis. For single values, only a single numerical value is necessary. For multiple lines, pass these as a vector, e.g., c(60,90).
vlineType	Line type for vline ('blank', 'solid', 'dashed', 'dotted', 'dotdash', 'longdash', 'twodash').
vlineCol	Colour of vline.
vlineWidth	Width of vline.
gridlines.major	Logical, indicating whether or not to draw major gridlines.
gridlines.minor	Logical, indicating whether or not to draw minor gridlines.
borderWidth	Width of the border on the x and y axes.
borderColour	Colour of the border on the x and y axes.
returnPlot	Logical, indicating whether or not to return the plot object.
flip_axes	Logical, indicating whether to swap the x- and y-axes.

Details

Draw a bi-plot, comparing 2 selected principal components / eigenvectors.

Value

A ggplot2 object.

Author(s)

Kevin Blighe <kevin@clinicalbioinformatics.co.uk>

Examples

```
options(scipen=10)
options(digits=6)

col <- 20
row <- 20000
mat1 <- matrix(
  rexp(col*row, rate = 0.1),
  ncol = col)
rownames(mat1) <- paste0('gene', 1:nrow(mat1))
colnames(mat1) <- paste0('sample', 1:ncol(mat1))

mat2 <- matrix(
rexp(col*row, rate = 0.1),
  ncol = col)
rownames(mat2) <- paste0('gene', 1:nrow(mat2))
colnames(mat2) <- paste0('sample', (ncol(mat1)+1):(ncol(mat1)+ncol(mat2)))

mat <- cbind(mat1, mat2)

metadata <- data.frame(row.names = colnames(mat))
metadata$Group <- rep(NA, ncol(mat))
metadata$Group[seq(1,40,2)] <- 'A'
metadata$Group[seq(2,40,2)] <- 'B'
metadata$CRP <- sample.int(100, size=ncol(mat), replace=TRUE)
metadata$ESR <- sample.int(100, size=ncol(mat), replace=TRUE)

p <- pca(mat, metadata = metadata, removeVar = 0.1)

biplot(p)

biplot(p, colby = 'Group', shape = 'Group')

biplot(p, colby = 'Group', colkey = c(A = 'forestgreen', B = 'gold'),
  legendPosition = 'right')

biplot(p, colby = 'Group', colkey = c(A='forestgreen', B='gold'),
  shape = 'Group', shapekey = c(A=10, B=21), legendPosition = 'bottom')
```

---

chooseGavishDonoho *Choosing PCs with the Gavish-Donoho method*

Description

Use the Gavish-Donoho method to determine the optimal number of PCs to retain.

Usage

```
chooseGavishDonoho(x, .dim = dim(x), var.explained, noise)
```

Arguments

x	The data matrix used for the PCA, containing variables in rows and observations in columns. Ignored if dim is supplied.
.dim	An integer vector containing the dimensions of the data matrix used for PCA. The first element should contain the number of variables and the second element should contain the number of observations.
var.explained	A numeric vector containing the variance explained by successive PCs. This should be sorted in decreasing order. Note that this should be the variance explained, NOT the percentage of variance explained!
noise	Numeric scalar specifying the variance of the random noise.

Details

Assuming that x is the sum of some low-rank truth and some i.i.d. random matrix with variance noise, the Gavish-Donoho method defines a threshold on the singular values that minimizes the reconstruction error from the PCs. This provides a mathematical definition of the "optimal" choice of the number of PCs for a given matrix, though it depends on both the i.i.d. assumption and an estimate for noise.

Value

An integer scalar specifying the number of PCs to retain. The effective limit on the variance explained is returned in the attributes.

Author(s)

Aaron Lun

See Also

chooseMarchenkoPastur, parallelPCA and findElbowPoint, for other approaches to choosing the number of PCs.

Examples

```
truth <- matrix(rnorm(1000), nrow=100)
truth <- truth[,sample(ncol(truth), 1000, replace=TRUE)]
obs <- truth + rnorm(length(truth), sd=2)

# Note, we need the variance explained, NOT the percentage
# of variance explained!
pcs <- pca(obs)
chooseGavishDonoho(obs, var.explained=pcs$sdev^2, noise=4)
```

`chooseMarchenkoPastur` *Choosing PCs with the Marchenko-Pastur limit*

Description

Use the Marchenko-Pastur limit to choose the number of top PCs to retain.

Usage

```
chooseMarchenkoPastur(x, .dim = dim(x), var.explained, noise)
```

Arguments

`x`	The data matrix used for the PCA, containing variables in rows and observations in columns. Ignored if `dim` is supplied.
`.dim`	An integer vector containing the dimensions of the data matrix used for PCA. The first element should contain the number of variables and the second element should contain the number of observations.
`var.explained`	A numeric vector containing the variance explained by successive PCs. This should be sorted in decreasing order. Note that this should be the variance explained, NOT the percentage of variance explained!
`noise`	Numeric scalar specifying the variance of the random noise.

Details

For a random matrix with i.i.d. values, the Marchenko-Pastur (MP) limit defines the maximum eigenvalue. Let us assume that `x` is the sum of some low-rank truth and some i.i.d. random matrix with variance `noise`. We can use the MP limit to determine the maximum variance that could be explained by a fully random PC; all PCs that explain more variance are thus likely to contain real structure and should be retained.

Of course, this has some obvious caveats such as the unrealistic i.i.d. assumption and the need to estimate `noise`. Moreover, PCs below the MP limit are not necessarily uninformative or lacking structure; it is just that their variance explained does not match the most extreme case that random noise has to offer.

Value

An integer scalar specifying the number of PCs with variance explained beyond the MP limit. The limit itself is returned in the attributes.

Author(s)

Aaron Lun

See Also

chooseGavishDonoho, parallelPCA and findElbowPoint, for other approaches to choosing the number of PCs.

Examples

```
truth <- matrix(rnorm(1000), nrow=100)
truth <- truth[,sample(ncol(truth), 1000, replace=TRUE)]
obs <- truth + rnorm(length(truth), sd=2)

# Note, we need the variance explained, NOT the percentage
# of variance explained!
pcs <- pca(obs)
chooseMarchenkoPastur(obs, var.explained=pcs$sdev^2, noise=4)
```

`eigencorplot` *Correlate principal components to continuous variable metadata and test significancies of these.*

Description

Correlate principal components to continuous variable metadata and test significancies of these.

Usage

```
eigencorplot(
  pcaobj,
  components = getComponents(pcaobj, seq_len(10)),
  metavars,
  titleX = "",
  cexTitleX = 1,
  rotTitleX = 0,
  colTitleX = "black",
  fontTitleX = 2,
  titleY = "",
  cexTitleY = 1,
  rotTitleY = 0,
  colTitleY = "black",
  fontTitleY = 2,
  cexLabX = 1,
  rotLabX = 0,
  colLabX = "black",
  fontLabX = 2,
  cexLabY = 1,
  rotLabY = 0,
  colLabY = "black",
```

```
  fontLabY = 2,
  posLab = "bottomleft",
 col = c("blue4", "blue3", "blue2", "blue1", "white", "red1", "red2", "red3", "red4"),
  posColKey = "right",
  cexLabColKey = 1,
  cexCorval = 1,
  colCorval = "black",
  fontCorval = 1,
  scale = TRUE,
  main = "",
  cexMain = 2,
  rotMain = 0,
  colMain = "black",
  fontMain = 2,
  corFUN = "pearson",
  corUSE = "pairwise.complete.obs",
  corMultipleTestCorrection = "none",
  signifSymbols = c("***", "**", "*", ""),
  signifCutpoints = c(0, 0.001, 0.01, 0.05, 1),
  colFrame = "white",
  plotRsquared = FALSE,
  returnPlot = TRUE
)
```

Arguments

pcaobj	Object of class 'pca' created by pca().
components	The principal components to be included in the plot.
metavars	A vector of column names in metadata representing continuos variables.
titleX	X-axis title.
cexTitleX	X-axis title cex.
rotTitleX	X-axis title rotation in degrees.
colTitleX	X-axis title colour.
fontTitleX	X-axis title font style. 1, plain; 2, bold; 3, italic; 4, bold-italic.
titleY	Y-axis title.
cexTitleY	Y-axis title cex.
rotTitleY	Y-axis title rotation in degrees.
colTitleY	Y-axis title colour.
fontTitleY	Y-axis title font style. 1, plain; 2, bold; 3, italic; 4, bold-italic.
cexLabX	X-axis labels cex.
rotLabX	X-axis labels rotation in degrees.
colLabX	X-axis labels colour.
fontLabX	X-axis labels font style. 1, plain; 2, bold; 3, italic; 4, bold-italic.
cexLabY	Y-axis labels cex.

rotLabY	Y-axis labels rotation in degrees.
colLabY	Y-axis labels colour.
fontLabY	Y-axis labels font style. 1, plain; 2, bold; 3, italic; 4, bold-italic.
posLab	Positioning of the X- and Y-axis labels. 'bottomleft', bottom and left; 'topright', top and right; 'all', bottom / top and left /right; 'none', no labels.
col	Colour shade gradient for RColorBrewer.
posColKey	Position of colour key. 'bottom', 'left', 'top', 'right'.
cexLabColKey	Colour key labels cex.
cexCorval	Correlation values cex.
colCorval	Correlation values colour.
fontCorval	Correlation values font style. 1, plain; 2, bold; 3, italic; 4, bold-italic.
scale	Logical, indicating whether or not to scale the colour range to max and min cor values.
main	Plot title.
cexMain	Plot title cex.
rotMain	Plot title rotation in degrees.
colMain	Plot title colour.
fontMain	Plot title font style. 1, plain; 2, bold; 3, italic; 4, bold-italic.
corFUN	Correlation method: 'pearson', 'spearman', or 'kendall'.
corUSE	Method for handling missing values (see documentation for cor function via ?cor). 'everything', 'all.obs', 'complete.obs', 'na.or.complete', or 'pairwise.complete.obs'.
corMultipleTestCorrection	Multiple testing p-value adjustment method. Any method from stats::p.adjust() can be used. Activating this function means that signifSymbols and signifCutpoints then relate to adjusted (not nominal) p-values.
signifSymbols	Statistical significance symbols to display beside correlation values.
signifCutpoints	Cut-points for statistical significance.
colFrame	Frame colour.
plotRsquared	Logical, indicating whether or not to plot R-squared values.
returnPlot	Logical, indicating whether or not to return the plot object.

Details

Correlate principal components to continuous variable metadata and test significancies of these.

Value

A lattice object.

Author(s)

Kevin Blighe <kevin@clinicalbioinformatics.co.uk>

Examples

```
options(scipen=10)
options(digits=6)

col <- 20
row <- 20000
mat1 <- matrix(
  rexp(col*row, rate = 0.1),
  ncol = col)
rownames(mat1) <- paste0('gene', 1:nrow(mat1))
colnames(mat1) <- paste0('sample', 1:ncol(mat1))

mat2 <- matrix(
  rexp(col*row, rate = 0.1),
  ncol = col)
rownames(mat2) <- paste0('gene', 1:nrow(mat2))
colnames(mat2) <- paste0('sample', (ncol(mat1)+1):(ncol(mat1)+ncol(mat2)))

mat <- cbind(mat1, mat2)

metadata <- data.frame(row.names = colnames(mat))
metadata$Group <- rep(NA, ncol(mat))
metadata$Group[seq(1,40,2)] <- 'A'
metadata$Group[seq(2,40,2)] <- 'B'
metadata$CRP <- sample.int(100, size=ncol(mat), replace=TRUE)
metadata$ESR <- sample.int(100, size=ncol(mat), replace=TRUE)

p <- pca(mat, metadata = metadata, removeVar = 0.1)

eigencorplot(p, components = getComponents(p, 1:10),
  metavars = c('ESR', 'CRP'))
```

`findElbowPoint` *Find the elbow point in the curve of variance explained by each successive PC. This can be used to determine the number of PCs to retain.*

Description

Find the elbow point in the curve of variance explained by each successive PC. This can be used to determine the number of PCs to retain.

Usage

```
findElbowPoint(variance)
```

Arguments

`variance`	Numeric vector containing the variance explained by each PC. Should be monotonic decreasing.

Details

Find the elbow point in the curve of variance explained by each successive PC. This can be used to determine the number of PCs to retain.

Value

An integer scalar specifying the number of PCs at the elbow point.

Author(s)

Aaron Lun

Examples

```
col <- 20
row <- 1000
mat <- matrix(rexp(col*row, rate = 1), ncol = col)

# Adding some structure to make it more interesting.
mat[1:100,1:3] <- mat[1:100,1:3] + 5
mat[1:100+100,3:6] <- mat[1:100+100,3:6] + 5
mat[1:100+200,7:10] <- mat[1:100+200,7:10] + 5
mat[1:100+300,11:15] <- mat[1:100+300,11:15] + 5

p <- pca(mat)
chosen <- findElbowPoint(p$variance)

plot(p$variance)
abline(v=chosen, col="red")
```

---

getComponents *Return the principal component labels for an object of class 'pca'.*

Description

Return the principal component labels for an object of class 'pca'.

Usage

```
getComponents(pcaobj, components = NULL)
```

Arguments

pcaobj	Object of class 'pca' created by pca().
components	Indices of the principal components whose names will be returned. If NULL, all PC names will be returned.

Details

Return the principal component labels for an object of class 'pca'.

Value

A character object.

Author(s)

Kevin Blighe <kevin@clinicalbioinformatics.co.uk>

Examples

```
options(scipen=10)
options(digits=6)

col <- 20
row <- 20000
mat1 <- matrix(
  rexp(col*row, rate = 0.1),
  ncol = col)
rownames(mat1) <- paste0('gene', 1:nrow(mat1))
colnames(mat1) <- paste0('sample', 1:ncol(mat1))

mat2 <- matrix(
  rexp(col*row, rate = 0.1),
  ncol = col)
rownames(mat2) <- paste0('gene', 1:nrow(mat2))
colnames(mat2) <- paste0('sample', (ncol(mat1)+1):(ncol(mat1)+ncol(mat2)))

mat <- cbind(mat1, mat2)

metadata <- data.frame(row.names = colnames(mat))
metadata$Group <- rep(NA, ncol(mat))
metadata$Group[seq(1,40,2)] <- 'A'
metadata$Group[seq(2,40,2)] <- 'B'
metadata$CRP <- sample.int(100, size=ncol(mat), replace=TRUE)
metadata$ESR <- sample.int(100, size=ncol(mat), replace=TRUE)

p <- pca(mat, metadata = metadata, removeVar = 0.1)

getComponents(p)
```

`getLoadings` *Return component loadings for principal components from an object of class 'pca'.*

Description

Return component loadings for principal components from an object of class 'pca'.

Usage

```
getLoadings(pcaobj, components = NULL)
```

Arguments

pcaobj	Object of class 'pca' created by pca().
components	Indices of the principal components whose component loadings will be returned. If NULL, all PC names will be returned.

Details

Return component loadings for principal components from an object of class 'pca'.

Value

A data.frame object.

Author(s)

Kevin Blighe <kevin@clinicalbioinformatics.co.uk>

Examples

```
options(scipen=10)
options(digits=6)

col <- 20
row <- 20000
mat1 <- matrix(
  rexp(col*row, rate = 0.1),
  ncol = col)
rownames(mat1) <- paste0('gene', 1:nrow(mat1))
colnames(mat1) <- paste0('sample', 1:ncol(mat1))

mat2 <- matrix(
  rexp(col*row, rate = 0.1),
  ncol = col)
rownames(mat2) <- paste0('gene', 1:nrow(mat2))
colnames(mat2) <- paste0('sample', (ncol(mat1)+1):(ncol(mat1)+ncol(mat2)))

mat <- cbind(mat1, mat2)

metadata <- data.frame(row.names = colnames(mat))
metadata$Group <- rep(NA, ncol(mat))
metadata$Group[seq(1,40,2)] <- 'A'
metadata$Group[seq(2,40,2)] <- 'B'
metadata$CRP <- sample.int(100, size=ncol(mat), replace=TRUE)
```

```
metadata$ESR <- sample.int(100, size=ncol(mat), replace=TRUE)

p <- pca(mat, metadata = metadata, removeVar = 0.1)

getLoadings(p)
```

`getVars` *Return the explained variation for each principal component for an object of class 'pca'.*

Description

Return the explained variation for each principal component for an object of class 'pca'.

Usage

```
getVars(pcaobj, components = NULL)
```

Arguments

`pcaobj`	Object of class 'pca' created by pca().
`components`	Indices of the principal components whose explained variances will be returned. If NULL, all values will be returned.

Details

Return the explained variation for each principal component for an object of class 'pca'.

Value

A `numeric` object.

Author(s)

Kevin Blighe <kevin@clinicalbioinformatics.co.uk>

Examples

```
options(scipen=10)
options(digits=6)

col <- 20
row <- 20000
mat1 <- matrix(
  rexp(col*row, rate = 0.1),
  ncol = col)
rownames(mat1) <- paste0('gene', 1:nrow(mat1))
colnames(mat1) <- paste0('sample', 1:ncol(mat1))
```

```
mat2 <- matrix(
  rexp(col*row, rate = 0.1),
  ncol = col)
rownames(mat2) <- paste0('gene', 1:nrow(mat2))
colnames(mat2) <- paste0('sample', (ncol(mat1)+1):(ncol(mat1)+ncol(mat2)))

mat <- cbind(mat1, mat2)

metadata <- data.frame(row.names = colnames(mat))
metadata$Group <- rep(NA, ncol(mat))
metadata$Group[seq(1,40,2)] <- 'A'
metadata$Group[seq(2,40,2)] <- 'B'
metadata$CRP <- sample.int(100, size=ncol(mat), replace=TRUE)
metadata$ESR <- sample.int(100, size=ncol(mat), replace=TRUE)

p <- pca(mat, metadata = metadata, removeVar = 0.1)

getVars(p)
```

---

pairsplot *Draw multiple bi-plots.*

Description

Draw multiple bi-plots.

Usage

```
pairsplot(
  pcaobj,
  components = getComponents(pcaobj, seq_len(5)),
  triangle = TRUE,
  trianglelabSize = 18,
  plotaxes = TRUE,
  margingaps = unit(c(0.1, 0.1, 0.1, 0.1), "cm"),
  ncol = NULL,
  nrow = NULL,
  x = NULL,
  y = NULL,
  colby = NULL,
  colkey = NULL,
  singlecol = NULL,
  shape = NULL,
  shapekey = NULL,
  pointSize = 1,
  legendPosition = "none",
  legendLabSize = 6,
```

```
  legendIconSize = 1.5,
  xlim = NULL,
  ylim = NULL,
  lab = NULL,
  labSize = 1.5,
  selectLab = NULL,
  drawConnectors = FALSE,
  widthConnectors = 0.5,
  colConnectors = "grey50",
  xlab = NULL,
  xlabAngle = 0,
  xlabhjust = 0.5,
  xlabvjust = 0.5,
  ylab = NULL,
  ylabAngle = 0,
  ylabhjust = 0.5,
  ylabvjust = 0.5,
  axisLabSize = 10,
  title = NULL,
  titleLabSize = 32,
  hline = NULL,
  hlineType = "longdash",
  hlineCol = "black",
  hlineWidth = 0.4,
  vline = NULL,
  vlineType = "longdash",
  vlineCol = "black",
  vlineWidth = 0.4,
  gridlines.major = TRUE,
  gridlines.minor = TRUE,
  borderWidth = 0.8,
  borderColour = "black",
  returnPlot = TRUE
)
```

Arguments

pcaobj	Object of class 'pca' created by pca().
components	The principal components to be included in the plot. These will be compared in a pairwise fashion via multiple calls to biplot().
triangle	Logical, indicating whether or not to draw the plots in the upper panel in a triangular arrangement? Principal component names will be labeled along the diagonal.
trianglelabSize	Size of p rincipal component label (when triangle = TRUE).
plotaxes	Logical, indicating whether or not to draw the axis tick, labels, and titles.
margingaps	The margins between plots in the plot space. Takes the form of a 'unit()' variable.

ncol	If triangle = FALSE, the number of columns in the final merged plot.
nrow	If triangle = FALSE, the number of rows in the final merged plot.
x	A principal component to plot on x-axis. All principal component names are stored in pcaobj$label.
y	A principal component to plot on y-axis. All principal component names are stored in pcaobj$label.
colby	If NULL, all points will be coloured differently. If not NULL, value is assumed to be a column name in pcaobj$metadata relating to some grouping/categorical variable.
colkey	Vector of name-value pairs relating to value passed to 'col', e.g., c(A='forestgreen', B='gold').
singlecol	If specified, all points will be shaded by this colour. Overrides 'col'.
shape	If NULL, all points will be have the same shape. If not NULL, value is assumed to be a column name in pcaobj$metadata relating to some grouping/categorical variable.
shapekey	Vector of name-value pairs relating to value passed to 'shape', e.g., c(A=10, B=21).
pointSize	Size of plotted points.
legendPosition	Position of legend ('top', 'bottom', 'left', 'right', 'none').
legendLabSize	Size of plot legend text.
legendIconSize	Size of plot legend icons / symbols.
xlim	Limits of the x-axis.
ylim	Limits of the y-axis.
lab	A vector containing labels to add to the plot.
labSize	Size of labels.
selectLab	A vector containing a subset of lab to plot.
drawConnectors	Logical, indicating whether or not to connect plot labels to their corresponding points by line connectors.
widthConnectors	Line width of connectors.
colConnectors	Line colour of connectors.
xlab	Label for x-axis.
xlabAngle	Rotation angle of x-axis labels.
xlabhjust	Horizontal adjustment of x-axis labels.
xlabvjust	Vertical adjustment of x-axis labels.
ylab	Label for y-axis.
ylabAngle	Rotation angle of y-axis labels.
ylabhjust	Horizontal adjustment of y-axis labels.
ylabvjust	Vertical adjustment of y-axis labels.
axisLabSize	Size of x- and y-axis labels.

title	Plot title.
titleLabSize	Size of plot title.
hline	Draw one or more horizontal lines passing through this/these values on y-axis. For single values, only a single numerical value is necessary. For multiple lines, pass these as a vector, e.g., c(60,90).
hlineType	Line type for hline ('blank', 'solid', 'dashed', 'dotted', 'dotdash', 'longdash', 'twodash').
hlineCol	Colour of hline.
hlineWidth	Width of hline.
vline	Draw one or more vertical lines passing through this/these values on x-axis. For single values, only a single numerical value is necessary. For multiple lines, pass these as a vector, e.g., c(60,90).
vlineType	Line type for vline ('blank', 'solid', 'dashed', 'dotted', 'dotdash', 'longdash', 'twodash').
vlineCol	Colour of vline.
vlineWidth	Width of vline.
gridlines.major	Logical, indicating whether or not to draw major gridlines.
gridlines.minor	Logical, indicating whether or not to draw minor gridlines.
borderWidth	Width of the border on the x and y axes.
borderColour	Colour of the border on the x and y axes.
returnPlot	Logical, indicating whether or not to return the plot object.

Details

Draw multiple bi-plots.

Value

A cowplot object.

Author(s)

Kevin Blighe <kevin@clinicalbioinformatics.co.uk>

Examples

```
options(scipen=10)
options(digits=6)

col <- 20
row <- 20000
mat1 <- matrix(
  rexp(col*row, rate = 0.1),
  ncol = col)
```

```
rownames(mat1) <- paste0('gene', 1:nrow(mat1))
colnames(mat1) <- paste0('sample', 1:ncol(mat1))

mat2 <- matrix(
  rexp(col*row, rate = 0.1),
  ncol = col)
rownames(mat2) <- paste0('gene', 1:nrow(mat2))
colnames(mat2) <- paste0('sample', (ncol(mat1)+1):(ncol(mat1)+ncol(mat2)))

mat <- cbind(mat1, mat2)

metadata <- data.frame(row.names = colnames(mat))
metadata$Group <- rep(NA, ncol(mat))
metadata$Group[seq(1,40,2)] <- 'A'
metadata$Group[seq(2,40,2)] <- 'B'
metadata$CRP <- sample.int(100, size=ncol(mat), replace=TRUE)
metadata$ESR <- sample.int(100, size=ncol(mat), replace=TRUE)

p <- pca(mat, metadata = metadata, removeVar = 0.1)

pairsplot(p, triangle = TRUE)
```

`parallelPCA` *Perform Horn's parallel analysis to choose the number of principal components to retain.*

Description

Perform Horn's parallel analysis to choose the number of principal components to retain.

Usage

```
parallelPCA(
  mat,
  max.rank = 100,
  ...,
  niters = 50,
  threshold = 0.1,
  transposed = FALSE,
  BSPARAM = ExactParam(),
  BPPARAM = SerialParam()
)
```

Arguments

`mat`	A numeric matrix where rows correspond to variables and columns correspond to samples.
`max.rank`	Integer scalar specifying the maximum number of PCs to retain.

...	Further arguments to pass to pca.
niters	Integer scalar specifying the number of iterations to use for the parallel analysis.
threshold	Numeric scalar representing the "p-value" threshold above which PCs are to be ignored.
transposed	Logical scalar indicating whether mat is transposed, i.e., rows are samples and columns are variables.
BSPARAM	A BiocSingularParam object specifying the algorithm to use for PCA.
BPPARAM	A BiocParallelParam object specifying how the iterations should be paralellized.

Details

Horn's parallel analysis involves shuffling observations within each row of x to create a permuted matrix. PCA is performed on the permuted matrix to obtain the percentage of variance explained under a random null hypothesis. This is repeated over several iterations to obtain a distribution of curves on the scree plot.

For each PC, the "p-value" (for want of a better word) is defined as the proportion of iterations where the variance explained at that PC is greater than that observed with the original matrix. The number of PCs to retain is defined as the last PC where the p-value is below threshold. This aims to retain all PCs that explain "significantly" more variance than expected by chance.

This function can be sped up by specifying BSPARAM=IrlbaParam() or similar, to use approximate strategies for performing the PCA. Another option is to set BPPARAM to perform the iterations in parallel.

Value

A list is returned, containing:

- original, the output from running pca on mat with the specified arguments.
- permuted, a matrix of variance explained from randomly permuted matrices. Each column corresponds to a single permutated matrix, while each row corresponds to successive principal components.
- n, the estimated number of principal components to retain.

Author(s)

Aaron Lun

Examples

```
# Mocking up some data.
ngenes <- 1000
means <- 2^runif(ngenes, 6, 10)
dispersions <- 10/means + 0.2
nsamples <- 50
counts <- matrix(rnbinom(ngenes*nsamples, mu=means,
  size=1/dispersions), ncol=nsamples)

# Choosing the number of PCs
```

```
lcounts <- log2(counts + 1)
output <- parallelPCA(lcounts)
output$n
```

---

pca *PCAtools*

Description

PCAtools

Usage

```
pca(
  mat,
  metadata = NULL,
  center = TRUE,
  scale = FALSE,
  rank = NULL,
  removeVar = NULL,
  transposed = FALSE,
  BSPARAM = ExactParam()
)
```

Arguments

mat	A data-matrix or data-frame containing numerical data only. Variables are expected to be in the rows and samples in the columns by default.
metadata	A data-matrix or data-frame containing metadata. This will be stored in the resulting pca object. Strictly enforced that rownames(metadata) == colnames(mat).
center	Center the data before performing PCA? Same as prcomp() 'center' parameter.
scale	Scale the data? Same as prcomp() 'scale' parameter.
rank	An integer scalar specifying the number of PCs to retain. OPTIONAL for an exact SVD, whereby it defaults to all PCs. Otherwise REQUIRED for approximate SVD methods.
removeVar	Remove this % of variables based on low variance.
transposed	Is mat transposed? DEFAULT = FALSE. If set to TRUE, samples are in the rows and variables are in the columns.
BSPARAM	A BiocSingularParam object specifying the algorithm to use for the SVD. Defaults to an exact SVD.

Details

Principal Component Analysis (PCA) is a very powerful technique that has wide applicability in data science, bioinformatics, and further afield. It was initially developed to analyse large volumes of data in order to tease out the differences/relationships between the logical entities being analysed. It extracts the fundamental structure of the data without the need to build any model to represent it. This 'summary' of the data is arrived at through a process of reduction that can transform the large number of variables into a lesser number that are uncorrelated (i.e. the 'principal components'), whilst at the same time being capable of easy interpretation on the original data. PCAtools provides functions for data exploration via PCA, and allows the user to generate publication-ready figures. PCA is performed via BiocSingular - users can also identify optimal number of principal component via different metrics, such as elbow method and Horn's parallel analysis, which has relevance for data reduction in single-cell RNA-seq (scRNA-seq) and high dimensional mass cytometry data.

Value

A pca object, containing:

- `rotated`, a data frame of the rotated data, i.e., the centred and scaled (if either or both are requested) input data multiplied by the variable loadings ('loadings'). This is the same as the 'x' variable returned by prcomp().
- `loadings`, a data frame of variable loadings ('rotation' variable returned by prcomp()).
- `variance`, a numeric vector of the explained variation for each principal component.
- `sdev`, the standard deviations of the principal components.
- `metadata`, the original metadata
- `xvars`, a character vector of rownames from the input data.
- `yvars`, a character vector of colnames from the input data.
- `components`, a character vector of principal component / eigenvector names.

Author(s)

Kevin Blighe <kevin@clinicalbioinformatics.co.uk>

Examples

```
options(scipen=10)
options(digits=6)

col <- 20
row <- 20000
mat1 <- matrix(
  rexp(col*row, rate = 0.1),
  ncol = col)
rownames(mat1) <- paste0('gene', 1:nrow(mat1))
colnames(mat1) <- paste0('sample', 1:ncol(mat1))

mat2 <- matrix(
  rexp(col*row, rate = 0.1),
  ncol = col)
```

```
rownames(mat2) <- paste0('gene', 1:nrow(mat2))
colnames(mat2) <- paste0('sample', (ncol(mat1)+1):(ncol(mat1)+ncol(mat2)))

mat <- cbind(mat1, mat2)

metadata <- data.frame(row.names = colnames(mat))
metadata$Group <- rep(NA, ncol(mat))
metadata$Group[seq(1,40,2)] <- 'A'
metadata$Group[seq(2,40,2)] <- 'B'
metadata$CRP <- sample.int(100, size=ncol(mat), replace=TRUE)
metadata$ESR <- sample.int(100, size=ncol(mat), replace=TRUE)

p <- pca(mat, metadata = metadata, removeVar = 0.1)

getComponents(p)

getVars(p)

getLoadings(p)

screeplot(p)

screeplot(p, hline = 80)

biplot(p)

biplot(p, colby = 'Group', shape = 'Group')

biplot(p, colby = 'Group', colkey = c(A = 'forestgreen', B = 'gold'),
  legendPosition = 'right')

biplot(p, colby = 'Group', colkey = c(A='forestgreen', B='gold'),
  shape = 'Group', shapekey = c(A=10, B=21), legendPosition = 'bottom')

pairsplot(p, triangle = TRUE)

plotloadings(p, drawConnectors=TRUE)

eigencorplot(p, components = getComponents(p, 1:10),
  metavars = c('ESR', 'CRP'))
```

`plotloadings` *Plot the component loadings for selected principal components / eigenvectors and label variables driving variation along these.*

Description

Plot the component loadings for selected principal components / eigenvectors and label variables driving variation along these.

Usage

```
plotloadings(
  pcaobj,
  components = getComponents(pcaobj, seq_len(5)),
  rangeRetain = 0.05,
  absolute = FALSE,
  col = c("gold", "white", "royalblue"),
  colMidpoint = 0,
  shape = 21,
  shapeSizeRange = c(10, 10),
  legendPosition = "top",
  legendLabSize = 10,
  legendIconSize = 3,
  xlim = NULL,
  ylim = NULL,
  labSize = 2,
  labhjust = 1.5,
  labvjust = 0,
  drawConnectors = TRUE,
  positionConnectors = "right",
  widthConnectors = 0.5,
  typeConnectors = "closed",
  endsConnectors = "first",
  lengthConnectors = unit(0.01, "npc"),
  colConnectors = "grey50",
  xlab = "Principal component",
  xlabAngle = 0,
  xlabhjust = 0.5,
  xlabvjust = 0.5,
  ylab = "Component loading",
  ylabAngle = 0,
  ylabhjust = 0.5,
  ylabvjust = 0.5,
  axisLabSize = 16,
  title = "",
  subtitle = "",
  caption = "",
  titleLabSize = 16,
  subtitleLabSize = 12,
  captionLabSize = 12,
  hline = c(0),
  hlineType = "longdash",
  hlineCol = "black",
  hlineWidth = 0.4,
  vline = NULL,
  vlineType = "longdash",
  vlineCol = "black",
  vlineWidth = 0.4,
```

```
  gridlines.major = TRUE,
  gridlines.minor = TRUE,
  borderWidth = 0.8,
  borderColour = "black",
  returnPlot = TRUE
)
```

Arguments

pcaobj	Object of class 'pca' created by pca().
components	The principal components to be included in the plot.
rangeRetain	Cut-off value for retaining variables. The function will look across each specified principal component and retain the variables that fall within this top/bottom fraction of the loadings range.
absolute	Logical, indicating whether or not to plot absolute loadings.
col	Colours used for generation of fill gradient according to loadings values. Can be 2 or 3 colours.
colMidpoint	Mid-point (loading) for the colour range.
shape	Shape of the plotted points.
shapeSizeRange	Size range for the plotted points (min, max).
legendPosition	Position of legend ('top', 'bottom', 'left', 'right', 'none').
legendLabSize	Size of plot legend text.
legendIconSize	Size of plot legend icons / symbols.
xlim	Limits of the x-axis.
ylim	Limits of the y-axis.
labSize	Size of labels.
labhjust	Horizontal adjustment of label.
labvjust	Vertical adjustment of label.
drawConnectors	Logical, indicating whether or not to connect plot labels to their corresponding points by line connectors.
positionConnectors	Position of the connectors and their labels with respect to the plotted points ('left', 'right').
widthConnectors	Line width of connectors.
typeConnectors	Have the arrow head open or filled ('closed')? ('open', 'closed').
endsConnectors	Which end of connectors to draw arrow head? ('last', 'first', 'both').
lengthConnectors	Length of the connectors.
colConnectors	Line colour of connectors.
xlab	Label for x-axis.
xlabAngle	Rotation angle of x-axis labels.

xlabhjust	Horizontal adjustment of x-axis labels.
xlabvjust	Vertical adjustment of x-axis labels.
ylab	Label for y-axis.
ylabAngle	Rotation angle of y-axis labels.
ylabhjust	Horizontal adjustment of y-axis labels.
ylabvjust	Vertical adjustment of y-axis labels.
axisLabSize	Size of x- and y-axis labels.
title	Plot title.
subtitle	Plot subtitle.
caption	Plot caption.
titleLabSize	Size of plot title.
subtitleLabSize	Size of plot subtitle.
captionLabSize	Size of plot caption.
hline	Draw one or more horizontal lines passing through this/these values on y-axis. For single values, only a single numerical value is necessary. For multiple lines, pass these as a vector, e.g., c(60,90).
hlineType	Line type for hline ('blank', 'solid', 'dashed', 'dotted', 'dotdash', 'longdash', 'twodash').
hlineCol	Colour of hline.
hlineWidth	Width of hline.
vline	Draw one or more vertical lines passing through this/these values on x-axis. For single values, only a single numerical value is necessary. For multiple lines, pass these as a vector, e.g., c(60,90).
vlineType	Line type for vline ('blank', 'solid', 'dashed', 'dotted', 'dotdash', 'longdash', 'twodash').
vlineCol	Colour of vline.
vlineWidth	Width of vline.
gridlines.major	Logical, indicating whether or not to draw major gridlines.
gridlines.minor	Logical, indicating whether or not to draw minor gridlines.
borderWidth	Width of the border on the x and y axes.
borderColour	Colour of the border on the x and y axes.
returnPlot	Logical, indicating whether or not to return the plot object.

Details

Plot the component loadings for selected principal components / eigenvectors and label variables driving variation along these.

Value

A ggplot2 object.

Author(s)

Kevin Blighe <kevin@clinicalbioinformatics.co.uk>

Examples

```
options(scipen=10)
options(digits=6)

col <- 20
row <- 20000
mat1 <- matrix(
  rexp(col*row, rate = 0.1),
  ncol = col)
rownames(mat1) <- paste0('gene', 1:nrow(mat1))
colnames(mat1) <- paste0('sample', 1:ncol(mat1))

mat2 <- matrix(
  rexp(col*row, rate = 0.1),
  ncol = col)
rownames(mat2) <- paste0('gene', 1:nrow(mat2))
colnames(mat2) <- paste0('sample', (ncol(mat1)+1):(ncol(mat1)+ncol(mat2)))

mat <- cbind(mat1, mat2)

metadata <- data.frame(row.names = colnames(mat))
metadata$Group <- rep(NA, ncol(mat))
metadata$Group[seq(1,40,2)] <- 'A'
metadata$Group[seq(2,40,2)] <- 'B'
metadata$CRP <- sample.int(100, size=ncol(mat), replace=TRUE)
metadata$ESR <- sample.int(100, size=ncol(mat), replace=TRUE)

p <- pca(mat, metadata = metadata, removeVar = 0.1)

plotloadings(p, drawConnectors = TRUE)
```

`screeplot` *Draw a SCREE plot, showing the distribution of explained variance across all or select principal components / eigenvectors.*

Description

Draw a SCREE plot, showing the distribution of explained variance across all or select principal components / eigenvectors.

Usage

```
screeplot(
  pcaobj,
  components = getComponents(pcaobj),
  xlim = NULL,
  ylim = c(0, 100),
  xlab = "Principal component",
  xlabAngle = 90,
  xlabhjust = 0.5,
  xlabvjust = 0.5,
  ylab = "Explained variation (%)",
  ylabAngle = 0,
  ylabhjust = 0.5,
  ylabvjust = 0.5,
  axisLabSize = 16,
  title = "SCREE plot",
  subtitle = "",
  caption = "",
  titleLabSize = 16,
  subtitleLabSize = 12,
  captionLabSize = 12,
  colBar = "dodgerblue",
  drawCumulativeSumLine = TRUE,
  colCumulativeSumLine = "red2",
  sizeCumulativeSumLine = 1.5,
  drawCumulativeSumPoints = TRUE,
  colCumulativeSumPoints = "red2",
  sizeCumulativeSumPoints = 2,
  hline = NULL,
  hlineType = "longdash",
  hlineCol = "black",
  hlineWidth = 0.4,
  vline = NULL,
  vlineType = "longdash",
  vlineCol = "black",
  vlineWidth = 0.4,
  gridlines.major = TRUE,
  gridlines.minor = TRUE,
  borderWidth = 0.8,
  borderColour = "black",
  returnPlot = TRUE
)
```

Arguments

pcaobj	Object of class 'pca' created by pca().
components	The principal components to be included in the plot.
xlim	Limits of the x-axis.

`ylim` Limits of the y-axis.

`xlab` Label for x-axis.

`xlabAngle` Rotation angle of x-axis labels.

`xlabhjust` Horizontal adjustment of x-axis labels.

`xlabvjust` Vertical adjustment of x-axis labels.

`ylab` Label for y-axis.

`ylabAngle` Rotation angle of y-axis labels.

`ylabhjust` Horizontal adjustment of y-axis labels.

`ylabvjust` Vertical adjustment of y-axis labels.

`axisLabSize` Size of x- and y-axis labels.

`title` Plot title.

`subtitle` Plot subtitle.

`caption` Plot caption.

`titleLabSize` Size of plot title.

`subtitleLabSize`
Size of plot subtitle.

`captionLabSize` Size of plot caption.

`colBar` Colour of the vertical bars.

`drawCumulativeSumLine`
Logical, indicating whether or not to overlay plot with a cumulative explained variance line.

`colCumulativeSumLine`
Colour of cumulative explained variance line.

`sizeCumulativeSumLine`
Size of cumulative explained variance line.

`drawCumulativeSumPoints`
Logical, indicating whether or not to draw the cumulative explained variance points.

`colCumulativeSumPoints`
Colour of cumulative explained variance points.

`sizeCumulativeSumPoints`
Size of cumulative explained variance points.

`hline` Draw one or more horizontal lines passing through this/these values on y-axis. For single values, only a single numerical value is necessary. For multiple lines, pass these as a vector, e.g., c(60,90).

`hlineType` Line type for hline ('blank', 'solid', 'dashed', 'dotted', 'dotdash', 'longdash', 'twodash').

`hlineCol` Colour of hline.

`hlineWidth` Width of hline.

`vline` Draw one or more vertical lines passing through this/these values on x-axis. For single values, only a single numerical value is necessary. For multiple lines, pass these as a vector, e.g., c(60,90).

vlineType	Line type for vline ('blank', 'solid', 'dashed', 'dotted', 'dotdash', 'longdash', 'twodash').
vlineCol	Colour of vline.
vlineWidth	Width of vline.
gridlines.major	Logical, indicating whether or not to draw major gridlines.
gridlines.minor	Logical, indicating whether or not to draw minor gridlines.
borderWidth	Width of the border on the x and y axes.
borderColour	Colour of the border on the x and y axes.
returnPlot	Logical, indicating whether or not to return the plot object.

Details

Draw a SCREE plot, showing the distribution of explained variance across all or select principal components / eigenvectors.

Value

A ggplot2 object.

Author(s)

Kevin Blighe <kevin@clinicalbioinformatics.co.uk>

Examples

```
options(scipen=10)
options(digits=6)

col <- 20
row <- 20000
mat1 <- matrix(
  rexp(col*row, rate = 0.1),
  ncol = col)
rownames(mat1) <- paste0('gene', 1:nrow(mat1))
colnames(mat1) <- paste0('sample', 1:ncol(mat1))

mat2 <- matrix(
  rexp(col*row, rate = 0.1),
  ncol = col)
rownames(mat2) <- paste0('gene', 1:nrow(mat2))
colnames(mat2) <- paste0('sample', (ncol(mat1)+1):(ncol(mat1)+ncol(mat2)))

mat <- cbind(mat1, mat2)

metadata <- data.frame(row.names = colnames(mat))
metadata$Group <- rep(NA, ncol(mat))
metadata$Group[seq(1,40,2)] <- 'A'
```

```
metadata$Group[seq(2,40,2)] <- 'B'
metadata$CRP <- sample.int(100, size=ncol(mat), replace=TRUE)
metadata$ESR <- sample.int(100, size=ncol(mat), replace=TRUE)

p <- pca(mat, metadata = metadata, removeVar = 0.1)

screeplot(p)

screeplot(p, hline = 80)
```

Index

BiocParallelParam, *25*
BiocSingularParam, *25*, *26*
biplot, 2

character, *17*
chooseGavishDonoho, 9, *12*
chooseMarchenkoPastur, *10*, 11
cowplot, *23*

data.frame, *18*

eigencorplot, 12

findElbowPoint, *10*, *12*, 15

getComponents, 16
getLoadings, 17
getVars, 19
ggplot2, *8*, *32*, *35*

lattice, *14*

numeric, *19*

pairsplot, 20
parallelPCA, *10*, *12*, 24
pca, *25*, 26, *27*
plotloadings, 28

screeplot, 32