Principal component analysis (PCA) is very useful for doing some basic quality control (e.g. looking for batch effects) and assessment of how the data is distributed (e.g. finding outliers). A straightforward way is to make your own wrapper function for prcomp and ggplot2, another way is to use the one that comes with M3C) or another package. M3C is a consensus clustering tool that makes some major modifications to the Monti et al. (2003) algorithm so that it behaves better, it also provides functions for data visualisation.

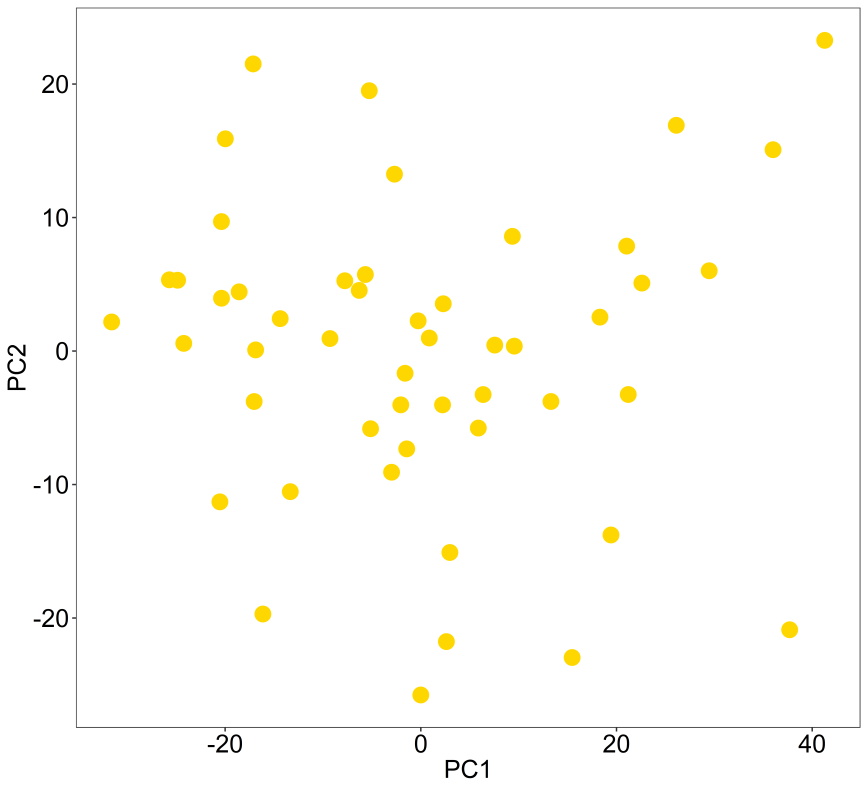
Let’s have a go on an example cancer microarray dataset.

# M3C loads an example dataset mydata with metadata desx

library(M3C)

# do PCA

pca(mydata,colvec=c('gold'),printres=TRUE)

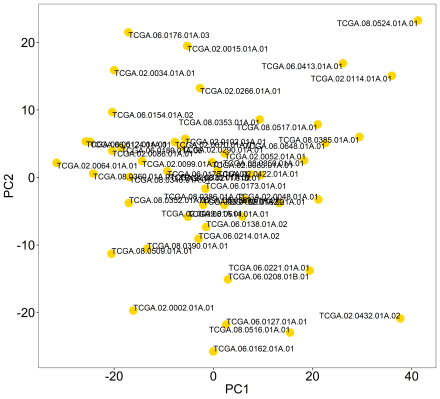


So, now what prcomp has done is extracted the eigenvectors of the data’s covariance matrix, then projected the original data samples onto them using linear combination. This yields PC scores which are plotted on PC1 and PC2 here (eigenvectors 1 and 2). The eigenvectors are sorted and these first two contain the highest variation in the data, but it might be a good idea to look at additional PCs, which is beyond this simple blog post and function.

You can see above there are no obvious outliers for removal here, which is good for cluster analysis. However, were there outliers, we can label all samples with their names using the ‘text’ parameter.

library(M3C)

pca(mydata,colvec=c('gold'),printres=TRUE,text=colnames(mydata))



Now other objectives would be comparing samples with batch to make sure we do not have batch effects driving the variation in our data, and comparing with other variables such as gender and age. Since the metadata only contains tumour class we are going to use that next to see how it is related to these PC scores.

This is a categorical variable, so the ‘scale’ parameter should be set to 3, ‘controlscale’ is set to TRUE because I want to choose the colours, and ‘labels’ parameter passes the metadata tumour class into the function. I am increasing the ‘printwidth’ from its default value because the legend is quite wide.

For more information see the function documentation using ?pca.

# first reformat meta to replace NA with Unknown

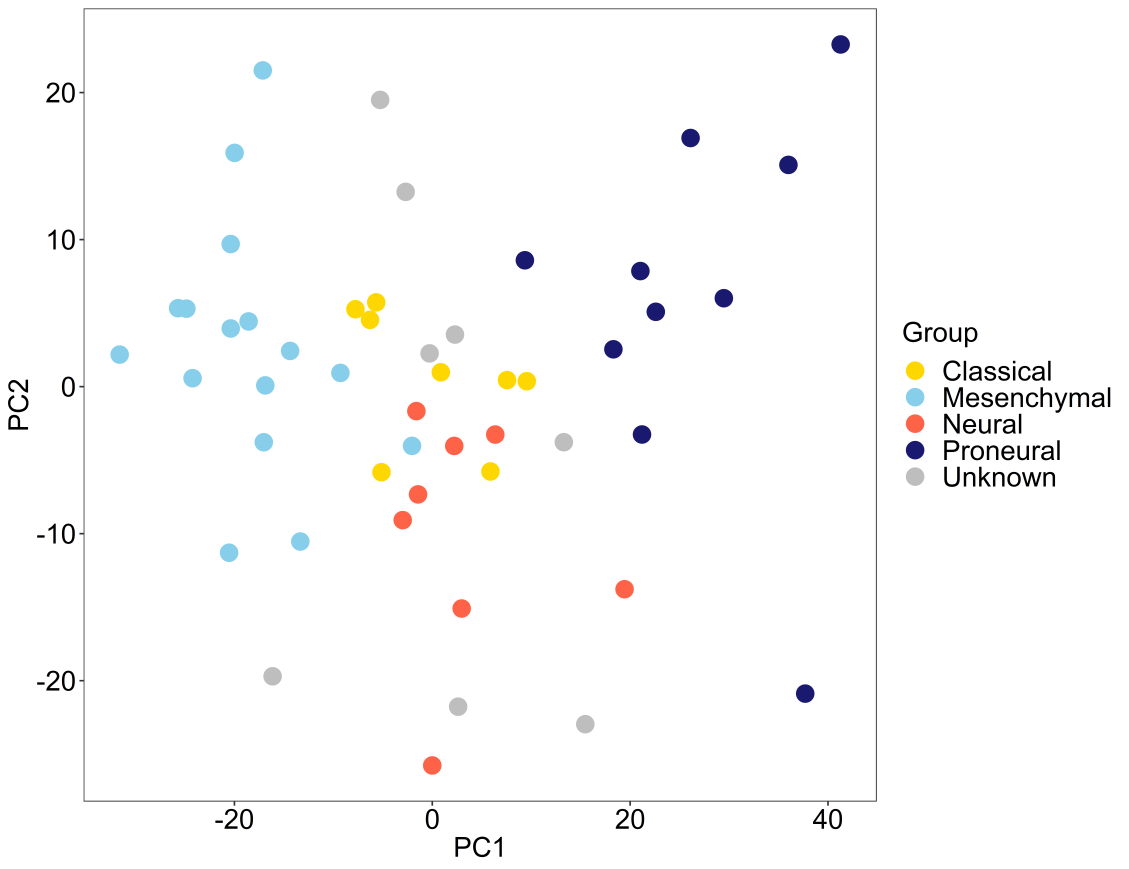
desx$class <- as.character(desx$class)

desx$class[is.na(desx$class)] <- as.factor('Unknown')

# next do the plot

pca(mydata,colvec=c('gold','skyblue','tomato','midnightblue','grey'),

labels=desx$class,controlscale=TRUE,scale=3,printres=TRUE,printwidth=25)

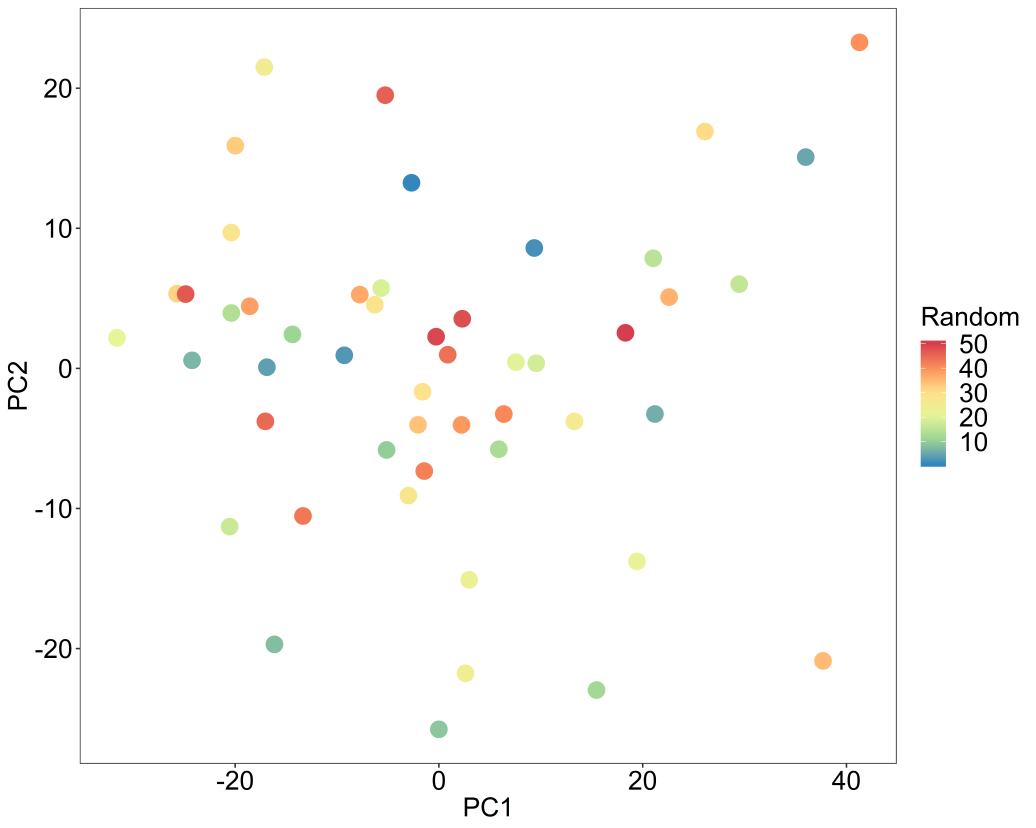


So, now it appears that the variation that governs these two PCs is indeed related to tumour class which is good. But, what if the variable is continuous, and we wanted to compare read mapping rate, or read duplication percentage, or age with our data? So, a simple change of the parameters can allow this too. Let’s make up a continuous variable, then add this to the plot. In this case we change the ‘scale’ parameter to reflect we are using continuous data, and the spectral palette is used for the colours by default.

randomvec <- seq(1,50)

pca(mydata,labels=randomvec,controlscale=TRUE,scale=1,legendtitle='Random',

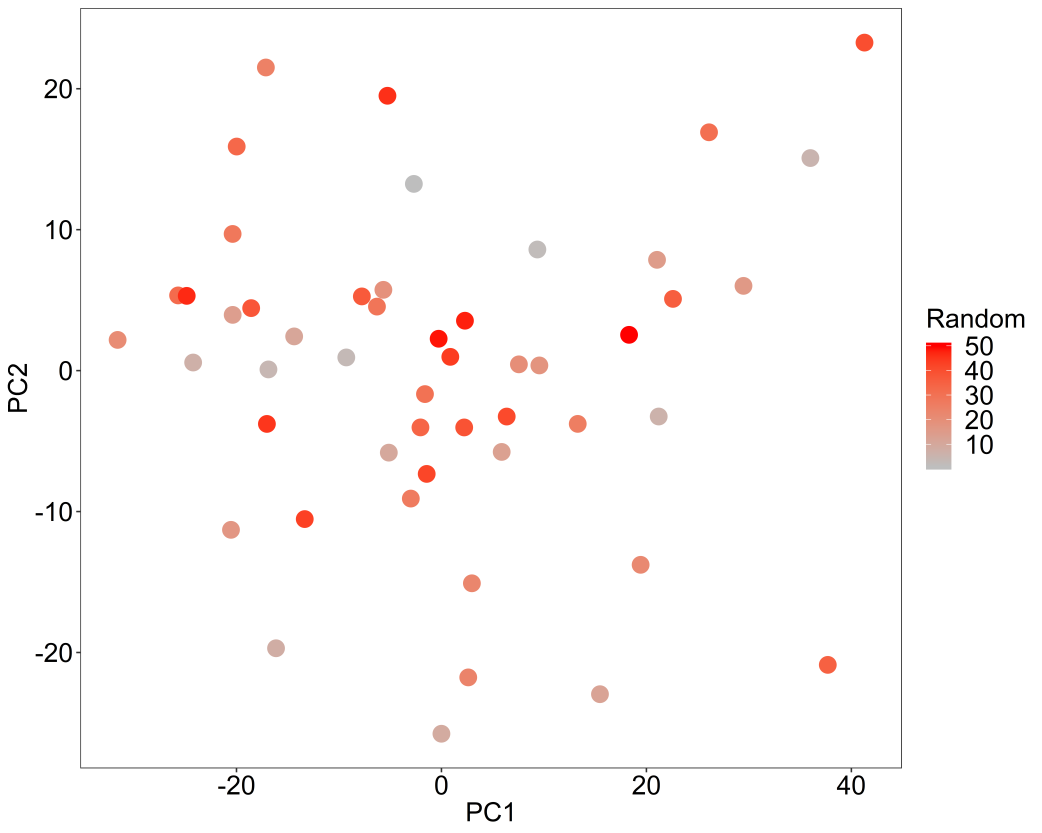
printres=TRUE,printwidth=25)



So, since this is a random variable, we can see it has no relationship with our data. Let’s just define a custom scale now. So we change ‘scale’ to 2, then use the ‘low’ and ‘high’ parameters to define the scale colour range.

pca(mydata,labels=randomvec,controlscale=TRUE,scale=2,legendtitle='Random',

printres=TRUE,,printwidth=25,low='grey',high='red')



Super easy, yet pretty cool. The idea here is just to minimise the amount of code hanging around for doing basic analyses like these. You can rip the code from here: Remember if your features are on different scales, the data needs transforming to be comparable beforehand.

PCA.R

|  |
| --- |
| #' pca: A principal component analysis function |
|  | #' |
|  | #' This is a flexible PCA function that can be run on a standard data frame. |
|  | #' It is a wrapper for prcomp/ggplot2 code and can be customised with different colours and font sizes |
|  | #' and more. |
|  | #' |
|  | #' @param mydata Data frame or matrix: if dataframe/matrix should have samples as columns and rows as features |
|  | #' @param printres Logical flag: whether to print the PCA into current directory |
|  | #' @param labels Character vector: if we want to just label with gender for example |
|  | #' @param text Character vector: if we wanted to label the samples with text IDs to look for outliers |
|  | #' @param axistextsize Numerical value: axis text size |
|  | #' @param legendtextsize Numerical value: legend text size |
|  | #' @param dotsize Numerical value: dot size |
|  | #' @param textlabelsize Numerical value: text inside plot label size |
|  | #' @param legendtitle Character vector: text legend title |
|  | #' @param controlscale Logical flag: whether to control the colour scale |
|  | #' @param scale Numerical value: 1=spectral palette, 2=manual low and high palette, 3=categorical labels |
|  | #' @param low Character vector: continuous scale low colour |
|  | #' @param high Character vector: continuous scale high colour |
|  | #' @param colvec Character vector: a series of colours in vector for categorical labels, e.g. c("sky blue", "gold") |
|  | #' @param printheight Numerical value: png height (default=20) |
|  | #' @param printwidth Numerical value: png width (default=22) |
|  | #' @param pcx Numerical value: which PC to plot on X axis (default=1) |
|  | #' @param pcy Numerical value: which PC to plot on Y axis (default=2) |
|  | #' @param scaler Logical flag: whether to scale the features of the input data (rows) (default=FALSE) |
|  | #' |
|  | #' @return A PCA plot object |
|  | #' @export |
|  | #' |
|  | #' @examples |
|  | #' PCA <- pca(mydata) |
|  |  |
|  | pca <- function(mydata, printres = FALSE, labels = FALSE, text = FALSE, axistextsize = 18, |
|  | legendtextsize = 18, dotsize = 5, textlabelsize = 4, legendtitle = 'Group', |
|  | controlscale = FALSE, scale = 1, low = 'grey', high = 'red', |
|  | colvec = c("skyblue", "gold", "violet", "darkorchid", "slateblue", "forestgreen", |
|  | "violetred", "orange", "midnightblue", "grey31", "black"), |
|  | printheight = 20, printwidth = 22, pcx=1, pcy=2, scaler=FALSE){ |
|  |  |
|  | ## basic error handling |
|  | if ( controlscale == TRUE && class(labels) %in% c( "character", "factor") && scale %in% c(1,2) ) { |
|  | stop("when categorical labels, use scale=3") |
|  | } |
|  | if ( controlscale == TRUE && class(labels) %in% c( "numeric") && scale %in% c(3) ) { |
|  | stop("when continuous labels, use scale=1 or scale=2") |
|  | } |
|  | if ( controlscale == FALSE && scale %in% c(2,3) ) { |
|  | warning("if your trying to control the scale, please set controlscale=TRUE") |
|  | } |
|  | if (sum(is.na(labels)) > 0 && class(labels) %in% c('character','factor')){ |
|  | warning("there is NA values in the labels vector, setting to unknown") |
|  | labels <- as.character(labels) |
|  | labels[is.na(labels)] <- 'Unknown' |
|  | } |
|  | if (sum(is.na(text)) > 0 && class(text) %in% c('character','factor')){ |
|  | warning("there is NA values in the text vector, setting to unknown") |
|  | text <- as.character(text) |
|  | text[is.na(text)] <- 'Unknown' |
|  | } |
|  |  |
|  | ## |
|  | message('\*\*\*PCA wrapper function\*\*\*') |
|  | message('running...') |
|  |  |
|  | ## |
|  | if (scaler){ |
|  | mydata <- data.frame(t(scale(t(mydata)))) |
|  | } |
|  | PC1 <- pcx |
|  | PC2 <- pcy |
|  |  |
|  | ## |
|  | if (labels[1] == FALSE && text[1] == FALSE){ |
|  |  |
|  | pca1 = prcomp(t(mydata)) |
|  | scores <- data.frame(pca1$x) # PC score matrix |
|  |  |
|  | eigs <- pca1$sdev^2 |
|  | variance\_percentage <- (eigs / sum(eigs))\*100 |
|  | pc1var <- round(variance\_percentage[PC1],digits=0) |
|  | pc2var <- round(variance\_percentage[PC2],digits=0) |
|  |  |
|  | p <- ggplot(data = scores, aes(x = scores[,pcx], y = scores[,pcy]) ) + geom\_point(colour='skyblue', size = dotsize) + |
|  | theme\_bw() + |
|  | theme(legend.position = "none", panel.grid.major = element\_blank(), panel.grid.minor = element\_blank(), |
|  | axis.text.y = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.text.x = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.title.x = element\_text(size = axistextsize), |
|  | axis.title.y = element\_text(size = axistextsize)) + |
|  | scale\_colour\_manual(values = colvec) + |
|  | xlab(paste('PC',PC1,' (',pc1var,'% variance)',sep='')) + |
|  | ylab(paste('PC',PC2,' (',pc2var,'% variance)',sep='')) |
|  |  |
|  | if (printres == TRUE){ |
|  | message('printing PCA to current directory...') |
|  | png('PCA.png', height = printheight, width = printwidth, units = 'cm', |
|  | res = 900, type = 'cairo') |
|  | print(p) # print ggplot CDF in main plotting window |
|  | dev.off() |
|  | } |
|  |  |
|  | }else if (labels[1] != FALSE && text[1] == FALSE){ #### KEY |
|  |  |
|  | pca1 = prcomp(t(mydata)) |
|  | scores <- data.frame(pca1$x) # PC score matrix |
|  |  |
|  | eigs <- pca1$sdev^2 |
|  | variance\_percentage <- (eigs / sum(eigs))\*100 |
|  |  |
|  | pc1var <- round(variance\_percentage[PC1],digits=0) |
|  | pc2var <- round(variance\_percentage[PC2],digits=0) |
|  |  |
|  | if (controlscale == TRUE){ |
|  | if (scale == 1){ |
|  | p <- ggplot(data = scores, aes(x = scores[,pcx], y = scores[,pcy]) ) + geom\_point(aes(colour = labels), size = dotsize) + |
|  | theme\_bw() + |
|  | theme(panel.grid.major = element\_blank(), panel.grid.minor = element\_blank(), |
|  | axis.text.y = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.text.x = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.title.x = element\_text(size = axistextsize), |
|  | axis.title.y = element\_text(size = axistextsize), |
|  | legend.title = element\_text(size = legendtextsize), |
|  | legend.text = element\_text(size = legendtextsize)) + |
|  | #guides(colour=guide\_legend(title=legendtitle)) + |
|  | labs(colour = legendtitle) + scale\_colour\_distiller(palette = "Spectral") + |
|  | xlab(paste('PC',PC1,' (',pc1var,'% variance)',sep='')) + |
|  | ylab(paste('PC',PC2,' (',pc2var,'% variance)',sep='')) |
|  | #scale\_colour\_gradient(low="red", high="white") |
|  |  |
|  | }else if (scale == 2){ |
|  | p <- ggplot(data = scores, aes(x = scores[,pcx], y = scores[,pcy]) ) + geom\_point(aes(colour = labels), size = dotsize) + |
|  | theme\_bw() + |
|  | theme(panel.grid.major = element\_blank(), panel.grid.minor = element\_blank(), |
|  | axis.text.y = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.text.x = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.title.x = element\_text(size = axistextsize), |
|  | axis.title.y = element\_text(size = axistextsize), |
|  | legend.title = element\_text(size = legendtextsize), |
|  | legend.text = element\_text(size = legendtextsize)) + |
|  | #guides(colour=guide\_legend(title=legendtitle)) + |
|  | labs(colour = legendtitle) + #scale\_colour\_distiller(palette = "Spectral") |
|  | scale\_colour\_gradient(low=low, high=high) + |
|  | xlab(paste('PC',PC1,' (',pc1var,'% variance)',sep='')) + |
|  | ylab(paste('PC',PC2,' (',pc2var,'% variance)',sep='')) |
|  |  |
|  | }else if (scale == 3){ |
|  | p <- ggplot(data = scores, aes(x = scores[,pcx], y = scores[,pcy]) ) + geom\_point(aes(colour = labels), size = dotsize) + |
|  | theme\_bw() + |
|  | theme(panel.grid.major = element\_blank(), panel.grid.minor = element\_blank(), |
|  | axis.text.y = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.text.x = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.title.x = element\_text(size = axistextsize), |
|  | axis.title.y = element\_text(size = axistextsize), |
|  | legend.title = element\_text(size = legendtextsize), |
|  | legend.text = element\_text(size = legendtextsize)) + |
|  | #guides(colour=guide\_legend(title=legendtitle)) + |
|  | labs(colour = legendtitle) + |
|  | scale\_colour\_manual(values = colvec) + |
|  | xlab(paste('PC',PC1,' (',pc1var,'% variance)',sep='')) + |
|  | ylab(paste('PC',PC2,' (',pc2var,'% variance)',sep='')) |
|  |  |
|  | } |
|  | }else{ |
|  | p <- ggplot(data = scores, aes(x = scores[,pcx], y = scores[,pcy]) ) + geom\_point(aes(colour = labels), size = dotsize) + |
|  | theme\_bw() + |
|  | theme(panel.grid.major = element\_blank(), panel.grid.minor = element\_blank(), |
|  | axis.text.y = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.text.x = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.title.x = element\_text(size = axistextsize), |
|  | axis.title.y = element\_text(size = axistextsize), |
|  | legend.title = element\_text(size = legendtextsize), |
|  | legend.text = element\_text(size = legendtextsize)) + |
|  | #guides(colour=guide\_legend(title=legendtitle)) + |
|  | labs(colour = legendtitle) + |
|  | xlab(paste('PC',PC1,' (',pc1var,'% variance)',sep='')) + |
|  | ylab(paste('PC',PC2,' (',pc2var,'% variance)',sep='')) |
|  |  |
|  | } |
|  |  |
|  | if (printres == TRUE){ |
|  | message('printing PCA to current directory...') |
|  | png('PCAlabeled.png', height = printheight, width = printwidth, units = 'cm', |
|  | res = 900, type = 'cairo') |
|  | print(p) # print ggplot CDF in main plotting window |
|  | dev.off() |
|  | } |
|  |  |
|  | }else if (labels[1] != FALSE && text[1] != FALSE){ ##### KEY |
|  |  |
|  | pca1 = prcomp(t(mydata)) |
|  | scores <- data.frame(pca1$x) # PC score matrix |
|  | scores$label <- text |
|  |  |
|  | eigs <- pca1$sdev^2 |
|  | variance\_percentage <- (eigs / sum(eigs))\*100 |
|  |  |
|  | pc1var <- round(variance\_percentage[PC1],digits=0) |
|  | pc2var <- round(variance\_percentage[PC2],digits=0) |
|  |  |
|  | if (controlscale == TRUE){ |
|  | if (scale == 1){ |
|  | p <- ggplot(data = scores, aes(x = scores[,pcx], y = scores[,pcy], label = label) ) + geom\_point(aes(colour = labels), size = dotsize) + |
|  | theme\_bw() + |
|  | theme(panel.grid.major = element\_blank(), panel.grid.minor = element\_blank(), |
|  | axis.text.y = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.text.x = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.title.x = element\_text(size = axistextsize), |
|  | axis.title.y = element\_text(size = axistextsize), |
|  | legend.title = element\_text(size = legendtextsize), |
|  | legend.text = element\_text(size = legendtextsize)) + |
|  | #guides(colour=guide\_legend(title=legendtitle)) + |
|  | labs(colour = legendtitle) + scale\_colour\_distiller(palette = "Spectral")+ |
|  | geom\_text(vjust="inward",hjust="inward",size=textlabelsize) + |
|  | xlab(paste('PC',PC1,' (',pc1var,'% variance)',sep='')) + |
|  | ylab(paste('PC',PC2,' (',pc2var,'% variance)',sep='')) |
|  |  |
|  | #scale\_colour\_gradient(low="red", high="white") |
|  | }else if (scale == 2){ |
|  | p <- ggplot(data = scores, aes(x = scores[,pcx], y = scores[,pcy], label = label) ) + geom\_point(aes(colour = labels), size = dotsize) + |
|  | theme\_bw() + |
|  | theme(panel.grid.major = element\_blank(), panel.grid.minor = element\_blank(), |
|  | axis.text.y = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.text.x = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.title.x = element\_text(size = axistextsize), |
|  | axis.title.y = element\_text(size = axistextsize), |
|  | legend.title = element\_text(size = legendtextsize), |
|  | legend.text = element\_text(size = legendtextsize)) + |
|  | #guides(colour=guide\_legend(title=legendtitle)) + |
|  | labs(colour = legendtitle) + #scale\_colour\_distiller(palette = "Spectral") |
|  | scale\_colour\_gradient(low=low, high=high)+ |
|  | geom\_text(vjust="inward",hjust="inward",size=textlabelsize) + |
|  | xlab(paste('PC',PC1,' (',pc1var,'% variance)',sep='')) + |
|  | ylab(paste('PC',PC2,' (',pc2var,'% variance)',sep='')) |
|  |  |
|  | }else if (scale == 3){ |
|  | p <- ggplot(data = scores, aes(x = scores[,pcx], y = scores[,pcy], label = label) ) + geom\_point(aes(colour = labels), size = dotsize) + |
|  | theme\_bw() + |
|  | theme(panel.grid.major = element\_blank(), panel.grid.minor = element\_blank(), |
|  | axis.text.y = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.text.x = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.title.x = element\_text(size = axistextsize), |
|  | axis.title.y = element\_text(size = axistextsize), |
|  | legend.title = element\_text(size = legendtextsize), |
|  | legend.text = element\_text(size = legendtextsize)) + |
|  | #guides(colour=guide\_legend(title=legendtitle)) + |
|  | labs(colour = legendtitle) + |
|  | scale\_colour\_manual(values = colvec)+ |
|  | geom\_text(vjust="inward",hjust="inward",size=textlabelsize) + |
|  | xlab(paste('PC',PC1,' (',pc1var,'% variance)',sep='')) + |
|  | ylab(paste('PC',PC2,' (',pc2var,'% variance)',sep='')) |
|  |  |
|  | } |
|  | }else{ |
|  | p <- ggplot(data = scores, aes(x = scores[,pcx], y = scores[,pcy], label = label) ) + geom\_point(aes(colour = labels), size = dotsize) + |
|  | theme\_bw() + |
|  | theme(panel.grid.major = element\_blank(), panel.grid.minor = element\_blank(), |
|  | axis.text.y = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.text.x = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.title.x = element\_text(size = axistextsize), |
|  | axis.title.y = element\_text(size = axistextsize), |
|  | legend.title = element\_text(size = legendtextsize), |
|  | legend.text = element\_text(size = legendtextsize)) + |
|  | #guides(colour=guide\_legend(title=legendtitle)) + |
|  | labs(colour = legendtitle) + |
|  | geom\_text(vjust="inward",hjust="inward",size=textlabelsize) + |
|  | xlab(paste('PC',PC1,' (',pc1var,'% variance)',sep='')) + |
|  | ylab(paste('PC',PC2,' (',pc2var,'% variance)',sep='')) |
|  |  |
|  | } |
|  |  |
|  | if (printres == TRUE){ |
|  | message('printing PCA to current directory...') |
|  | png('PCAlabeled.png', height = printheight, width = printwidth, units = 'cm', |
|  | res = 900, type = 'cairo') |
|  | print(p) # print ggplot CDF in main plotting window |
|  | dev.off() |
|  | } |
|  |  |
|  | }else if (labels[1] == FALSE && text[1] != FALSE){ |
|  |  |
|  | pca1 = prcomp(t(mydata)) |
|  | scores <- data.frame(pca1$x) # PC score matrix |
|  | scores$label <- text |
|  |  |
|  | eigs <- pca1$sdev^2 |
|  | variance\_percentage <- (eigs / sum(eigs))\*100 |
|  |  |
|  | pc1var <- round(variance\_percentage[PC1],digits=0) |
|  | pc2var <- round(variance\_percentage[PC2],digits=0) |
|  |  |
|  | p <- ggplot(data = scores, aes(x = scores[,pcx], y = scores[,pcy], label = label) ) + |
|  | geom\_point(aes(colour = factor(rep(1, ncol(mydata)))), size = dotsize) + |
|  | theme\_bw() + |
|  | theme(legend.position = "none", panel.grid.major = element\_blank(), panel.grid.minor = element\_blank(), |
|  | axis.text.y = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.text.x = element\_text(size = axistextsize, colour = 'black'), |
|  | axis.title.x = element\_text(size = axistextsize), |
|  | axis.title.y = element\_text(size = axistextsize)) + |
|  | scale\_colour\_manual(values = colvec) + |
|  | geom\_text(vjust="inward",hjust="inward",size=textlabelsize) + |
|  | xlab(paste('PC',PC1,' (',pc1var,'% variance)',sep='')) + |
|  | ylab(paste('PC',PC2,' (',pc2var,'% variance)',sep='')) |
|  |  |
|  | if (printres == TRUE){ |
|  | message('printing PCA to current directory...') |
|  | png('PCA.png', height = printheight, width = printwidth, units = 'cm', |
|  | res = 900, type = 'cairo') |
|  | print(p) # print ggplot CDF in main plotting window |
|  | dev.off() |
|  | } |
|  |  |
|  | }else{ |
|  | message('no valid options detected') |
|  | } |
|  |  |
|  | message('done.') |
|  |  |
|  | return(p) |
|  | } |

M3C Base Script

## ----setup, include = FALSE---------------------------------------------------

knitr::opts\_chunk$set(

collapse = TRUE,

comment = "#>"

)

## ---- message=FALSE-----------------------------------------------------------

library(M3C)

# now we have loaded the mydata and desx objects (with the package automatically)

# mydata is the expression data for GBM

# desx is the annotation for this data

## ----fig.width=4,fig.height=4-------------------------------------------------

pca(mydata,legendtextsize = 10,axistextsize = 10,dotsize=2)

## ---- results='hide'----------------------------------------------------------

# for vignette

res <- M3C(mydata, des = desx, removeplots = TRUE, iters=25,

objective='PAC', fsize=8, lthick=1, dotsize=1.25)

## -----------------------------------------------------------------------------

res$scores

## -----------------------------------------------------------------------------

for (k in seq(2,10)){

myresults <- res$realdataresults[[k]]$ordered\_annotation

chifit <- suppressWarnings(chisq.test(table(myresults[c('consensuscluster','class')])))

print(chifit$p.value)

}

## ----fig.width=5,fig.height=3.5-----------------------------------------------

res$plots[[1]]

## ----fig.width=4,fig.height=3-------------------------------------------------

res$plots[[2]]

## ----fig.width=4,fig.height=3-------------------------------------------------

res$plots[[4]]

## ----fig.width=4,fig.height=3-------------------------------------------------

res$plots[[3]]

## ----fig.show = 'hide'--------------------------------------------------------

data <- res$realdataresults[[4]]$ordered\_data

annon <- res$realdataresults[[4]]$ordered\_annotation

ccmatrix <- res$realdataresults[[4]]$consensus\_matrix

head(annon)

## -----------------------------------------------------------------------------

# library(ComplexHeatmap)

# ccl <- list()

# x <- c("skyblue", "gold", "violet", "darkorchid", "slateblue", "forestgreen",

# "violetred", "orange", "midnightblue", "grey31", "black")

# names(x) <- as.character(seq(1,11,by=1))

# for (i in seq(2,10)){

# # get cc matrix and labels

# ccmatrix <- res$realdataresults[[i]]$consensus\_matrix

# annon <- res$realdataresults[[i]]$ordered\_annotation

# # do heatmap

# n <- 10

# seq <- rev(seq(0,255,by=255/(n)))

# palRGB <- cbind(seq,seq,255)

# mypal <- rgb(palRGB,maxColorValue=255)

# ha = HeatmapAnnotation(

# df= data.frame(Cluster=as.character(annon[,1])), col = list(Cluster=x))

# ccl[[i]] <- Heatmap(ccmatrix, name = "Consensus\_index", top\_annotation = ha,

# col=mypal, show\_row\_dend = FALSE,

# show\_column\_dend = FALSE, cluster\_rows = FALSE, cluster\_columns = FALSE,

# show\_column\_names = FALSE)

# }

## ----fig.width=5,fig.height=4-------------------------------------------------

pca(data,labels=annon$consensuscluster,legendtextsize = 10,axistextsize = 10,dotsize=2)

## ----fig.show = 'hide',results='hide'-----------------------------------------

res <- M3C(mydata, method = 2, objective='PAC',fsize=8, lthick=1, dotsize=1.25)

## ----fig.width=4,fig.height=3-------------------------------------------------

res$plots[[3]]

## ----fig.show = 'hide',results='hide'-----------------------------------------

res <- M3C(mydata, method = 2,fsize=8, lthick=1, dotsize=1.25)

## ----fig.width=4,fig.height=3-------------------------------------------------

res$plots[[2]]

## ----fig.width=4,fig.height=3-------------------------------------------------

res$plots[[3]]

## -----------------------------------------------------------------------------

filtered\_results <- featurefilter(mydata, percentile=10, method='MAD', topN=5)