Barseq analysis example code

Simon Schmidt

13/02/2019

```r
library(data.table)
library(tidyr)
library(RColorBrewer)
library(reshape2)
library(ggplot2)
library(statmod)
library(plyr)
library(cowplot)
library(ggrepel)
library(data.table)
library(openxlsx)
library(edgeR)

mypal1 <- brewer.pal(8, "Set1")
mypal2 <- brewer.pal(4, "Set1")
names(mypal1) <- c(-1, 0, 1)
mylabels <- (c("reduced", "no change", "increased"))
names(mylabels) <- c(-1, 0, 1)
pointlabels <- c("strain")
```

This script requires the following tab delimited text components. 1) strain names list ("strainnames.txt") 2) targets.txt 3) contrasts.txt 4) folder of counts

Import the targets file that describes the relationship between sequence files and treatments. Designate the treatment and time columns as factors and create a new column that is a concatenation of time and treatments that will act as a grouping variable. Use that to create a DGE objects file that compiles all the information. Filter the data to keep only those that meet some minimum criteria and remove those that contain no data.

```r
targets <- readTargets("targets.txt")

targets$treatment <- factor(targets$treatment, levels = c("DM",
    "HS", "HC", "LpH", "LYAN", "HK", "LpH_HK", "HS_LYAN"))
targets$time <- factor(targets$time, levels = c("1", "2", "3",
    "4"))
targets$group <- targets$group <- factor(paste(targets$treatment,
    targets$time, sep = "."), levels = c("DM.1", "HS.1", "HC.1",
    "LpH.1", "LYAN.1", "HK.1", "LpH_HK.1", "HS_LYAN.1", "DM.2",
    "HS.2", "HC.2", "LpH.2", "LYAN.2", "HK.2", "LpH_HK.2", "HS_LYAN.2",
    "DM.3", "HS.3", "HC.3", "LpH.3", "LYAN.3", "HK.3", "LpH_HK.3",
    "HS_LYAN.3", "DM.4", "HS.4", "HC.4", "LpH.4", "LYAN.4", "HK.4",
    "LpH_HK.4", "HS_LYAN.4"))

data.DGE <- readDGE(targets, comment.char = "!")

keep <- rowSums(cpm(data.DGE) > 15) >= 3
data.DGE <- data.DGE[keep, , keep.lib.sizes = FALSE]
summary(keep)
```

Create an experimental design. In this case without an intercept. Use the "group" column created in targets above to name the columns of the design. This will allow the contrasts table to be built and pairwise analysis to be coded.

After the design is defined calculate normalization factors (RLE is used here) and estimate dipersions.

```r
design <- model.matrix(~0 + treatment + treatment:time, data = data.DGE$samples)

colnames(design) <- levels(targets$group)

data.DGE <- calcNormFactors(data.DGE, method = "RLE")
data.DGE <- estimateGLMCommonDisp(data.DGE, design)
data.DGE <- estimateGLMTrendedDisp(data.DGE, design)
data.DGE <- estimateGLMTagwiseDisp(data.DGE, design)
```

Check the data with a multidimensional scaling plot

```r
mypalette <- brewer.pal(8, "Dark2")
points <- c(0, 1, 2, 3, 15, 16, 17, 18)
pMDS <- plotMDS(data.DGE, col = mypalette[data.DGE$samples$time],
    pch = points[data.DGE$samples$treatment])
legend("topleft", legend = levels(data.DGE$samples$treatment),
    pch = points, col = "#1B9E77", ncol = 2, title = "treatments")
inset <- c(0, 0)
legend("topright", legend = levels(data.DGE$samples$time), pch = 15,
    col = mypalette, horiz = TRUE, title = "time points", inset = inset)
```

Fit the model and plot the dispersions. Conduct a likelyhood ratio test and build the contrasts based on the design. A quasilikelyhood test can also be performed using the following code. Quasilikelyhood tests are appropriate when variance from a poisson distribution are observed.

```r
fit <- glmQLFit(data.DGE, design, robust = TRUE)
plotQLDisp(fit)
```

Build a table of contrasts using the column names in the design table. Because the design in this case does not have an intercept the treatments are all relative to themselves at time point one. The contrast takes the form HS.3 = HS.3-DM.3

This form of contrast is tabulated for every pairwise comparison that is required. The contrasts list (saved as a tab delimited file) is imported using read.delim() and made into an matrix using makeContrasts(). Putting make contrasts in the "as.call" wrapper allows it to work through the list of text elements in the contrasts file.

```r
write.table(targets, file = "contrasts.txt", sep = "\t", row.names = FALSE,
    col.names = TRUE)

contrasts <- read.delim("150317_contrasts.txt", colClasses = "character")
my.contrasts <- eval(as.call(c(as.symbol("makeContrasts"), as.list(noquote(contrasts[,
    2])), levels = list(design))))
colnames(my.contrasts) <- contrasts[[1]]
```

Use a for loop that will run through the items in "my.contrasts" (which should be made up of components that match the headings in the design table), perform a glmQLFTest analysis on that contrast using the data in the object "fit", and extract the information using the user defined formula (datextract) to pull the details into a data frame, save them to a named list called "analysis". We can then use an llply function to run a user defined graphing function ("graph1") over the analysis list to output the graphs to another list called "myplots".

Use write.xls to save all analyses in the "analysis" list elements to an xcel workbook.

Use a for loop to export all the graphs saved in “myplots” list as png files to an image folder. Copy the location of the folder into the ggsave command at the “copy path” location.

```r
analysis <- vector(mode = "list", length = length(contrasts[[1]]))
names(analysis) <- contrasts[[1]]

datextract <- function(dat) {
    detags1 <- decideTestsDGE(dat, adjust.method = "BH", p.value = 0.05)
    top.x <- topTags(dat, n = Inf)
    df <- top.x$table
    df <- cbind(geneID = rownames(df), df)
    rownames(df) <- NULL
    df$geneID <- as.character(df$geneID)
    df <- separate(df, geneID, c("tag", "ID"), sep = "\\I")
    df$tag <- NULL
    df$ID <- as.numeric(df$ID)
    df <- data.table(df, key = "ID")
    df2 <- data.frame(detags1)
    df2 <- cbind(geneID = rownames(df2), df2)
    df2 <- separate(df2, geneID, c("tag", "ID"), sep = "\\I")
    df2$tag <- NULL
    df2$ID <- as.numeric(df2$ID)
    df2 <- data.table(df2, key = "ID")
    colnames(df2) <- c("ID", "sig")
    df3 <- df2[df]
    df3$sig <- as.character(df3$sig)
    df3$sig <- as.factor(df3$sig)
    return(df3)
}

for (i in seq_along(contrasts[[1]])) {
    exper <- contrasts[[1]][i]
    results <- glmQLFTest(fit, contrast = my.contrasts[, exper])
    analysis[[exper]] <- datextract(results)
}

write.xlsx(analysis, file = "150317_analysis.xlsx")

graph1 <- function(dat) {
    # create the named labelling vectors
    mypal1 <- brewer.pal(3, "Set1")
    names(mypal1) <- c(-1, 0, 1)
    mylabels <- (c("increased", "no change", "reduced"))
    names(mylabels) <- c(1, 0, -1)
    # evaluate if the significance column is made of all zeros so
    # that data sets with no significant values can be plotted
    # appropriately
    if (nrow(subset(analysis[[dat]], sig != "0")) != 0)
        dat1 <- subset(analysis[[dat]], sig != "0") else dat1 <- analysis[[dat]]
    ggplot(data = analysis[[dat]], aes(x = logCPM, y = logFC,
        group = sig, colour = sig)) + geom_point(size = 2) +
        scale_colour_manual(values = mypal1, name = "Change\nin\nPopulation\nrepresentation",
            labels = mylabels) + scale_size(range = c(0.5, 7)) +
        scale_y_continuous(breaks = c(-12, -10, -8, -6, -4, -2,
            0, 2, 4, 6, 8, 10, 12)) + geom_hline(yintercept = -1,
        colour = "blue") + geom_hline(yintercept = 1, colour = "blue") +
        xlab(expression(paste("Average log"["2"], " ", "counts per million reads"))) +
        ylab(expression(paste("log"["2"], " fold change"))) +
        theme_cowplot() + background_grid() + geom_text_repel(aes(label = ID),
        colour = "dark grey", size = 1, segment.size = 0.25,
        segment.alpha = 0.8) + theme(axis.title = element_text(size = 8),
        axis.text = element_text(size = 8), legend.text = element_text(size = 6),
        legend.title = element_text(size = 6), legend.key.size = unit(0.25,
            "cm"))
}

myplots <- vector(mode = "list", length = length(contrasts[[1]]))
names(myplots) <- contrasts[[1]]
myplots <- llply(names(analysis), graph1)
# write all the plots to a folder in the working directory
# called images
for (i in seq_along(myplots)) {
    plotname <- names(myplots[i])
    ggsave(filename = paste0("Copy_PATH", plotname, ".png"),
        plot = myplots[[plotname]], device = "png", width = 10,
        height = 6, units = "cm", dpi = 300)
}
```

The final output should be a set of analysis in excel format and a folder of ggplot2 based graphs that are saved to a separate folder

```
devtools::session_info()
```

```
## - Session info ---------------------------------------------------------------
##  setting  value
##  version  R version 3.5.3 (2019-03-11)
##  os       macOS Mojave 10.14.6
##  system   x86_64, darwin15.6.0
##  ui       X11
##  language (EN)
##  collate  en_AU.UTF-8
##  ctype    en_AU.UTF-8
##  tz       Australia/Adelaide
##  date     2019-11-21
##
## - Packages -------------------------------------------------------------------
##  package     * version date       lib source
##  assertthat    0.2.1   2019-03-21 [1] CRAN (R 3.5.1)
##  backports     1.1.5   2019-10-02 [1] CRAN (R 3.5.2)
##  callr         3.3.2   2019-09-22 [1] CRAN (R 3.5.2)
##  cli           1.1.0   2019-03-19 [1] CRAN (R 3.5.2)
##  crayon        1.3.4   2017-09-16 [1] CRAN (R 3.5.0)
##  desc          1.2.0   2018-05-01 [1] CRAN (R 3.5.0)
##  devtools      2.2.1   2019-09-24 [1] CRAN (R 3.5.2)
##  digest        0.6.22  2019-10-21 [1] CRAN (R 3.5.2)
##  ellipsis      0.3.0   2019-09-20 [1] CRAN (R 3.5.2)
##  evaluate      0.14    2019-05-28 [1] CRAN (R 3.5.2)
##  formatR       1.7     2019-06-11 [1] CRAN (R 3.5.2)
##  fs            1.3.1   2019-05-06 [1] CRAN (R 3.5.2)
## glue          1.3.1   2019-03-12 [1] CRAN (R 3.5.2)
## htmltools     0.4.0   2019-10-04 [1] CRAN (R 3.5.2)
## knitr         1.26    2019-11-12 [1] CRAN (R 3.5.2)
## magrittr      1.5     2014-11-22 [1] CRAN (R 3.5.0)
## memoise       1.1.0   2017-04-21 [1] CRAN (R 3.5.0)
## pkgbuild      1.0.6   2019-10-09 [1] CRAN (R 3.5.2)
## pkgload       1.0.2   2018-10-29 [1] CRAN (R 3.5.0)
## prettyunits   1.0.2   2015-07-13 [1] CRAN (R 3.5.0)
## processx      3.4.1   2019-07-18 [1] CRAN (R 3.5.2)
## ps            1.3.0   2018-12-21 [1] CRAN (R 3.5.0)
## R6            2.4.1   2019-11-12 [1] CRAN (R 3.5.2)
## Rcpp          1.0.3   2019-11-08 [1] CRAN (R 3.5.2)
## remotes       2.1.0   2019-06-24 [1] CRAN (R 3.5.2)
## rlang         0.4.1   2019-10-24 [1] CRAN (R 3.5.2)
## rmarkdown     1.17    2019-11-13 [1] CRAN (R 3.5.2)
## rprojroot     1.3-2   2018-01-03 [1] CRAN (R 3.5.0)
## sessioninfo   1.1.1   2018-11-05 [1] CRAN (R 3.5.0)
## stringi       1.4.3   2019-03-12 [1] CRAN (R 3.5.2)
## stringr       1.4.0   2019-02-10 [1] CRAN (R 3.5.2)
## testthat      2.3.0   2019-11-05 [1] CRAN (R 3.5.2)
## usethis       1.5.1   2019-07-04 [1] CRAN (R 3.5.2)
## withr         2.1.2   2018-03-15 [1] CRAN (R 3.5.0)
## xfun          0.11    2019-11-12 [1] CRAN (R 3.5.2)
## yaml          2.2.0   2018-07-25 [1] CRAN (R 3.5.0)
##
## [1] /Users/simon/Rlibs
## [2] /Library/Frameworks/R.framework/Versions/3.5/Resources/library
```