DGE.Tools Data Exploration Plotting Tools

John Thompson

December 28, 2015

Contents

1 DGE.Tools: Deluxe Plotting tools

DGE.Tools has been updated to include an assortment of standard data exploration plotting tools. The intention here is to make common types of plots dead easy to produce and provide a consistent look and feel.

The Plotting functions are:

- JRT_heatmap: convenience wrapper for pheatmap
- profilePlot: LogInt vs. LogRatio (aka MA plot)
- volcanoPlot: LogRatio vs. NegLogP
- cdfPlot: Rank(pvalue) vs. NegLogP
- comparePlot: Compare LogRatios for two contrasts
- obsPlot: Gene boxplots (faceted)
- plotPvalHist: faceted Pvalue histogram

Several reusable themes have been created to support these plots.

With these additions, DGE.Tools now supports the following aspects of DGE analysis.

- DGE workflow (from counts through present/absent filtering (thanks to Ron), normalization, limma modeling and contrasts)
- DGE QC: plotDispersion, voom mean-variance plot, plotPvalHist, cdfPlot
- Exploratory Analysis: Contrast Table, profilePlot, volcanoPlot, comparPlot, obsPlot, JRT_heatmap
- Scalable themes for general use: greyTheme, bwTheme, baseFont, printAndSave
- Some rudimentry ID mapping functions: Entrez2GeneSym, EnsemblGene2Entrez, GeneSym2Entrez (But note these are a simple hack that don't attempt to deal with 1toMany issues. I think the annotables package looks like a better solution for ID mapping tasks.)

Code Block: Load some test data (IPF Fibroblast data) and set global heatmap options.

```
rm(list=ls()) #Clear the workspace
invisible(gc()) #garbage collection to masimize available memory
startTime = Sys.time()

setwd("~/Fibrosis/IPF Cedar Sinai/RNA-Seq/RefGene2015/RData")
library(ggplot2)
library(magrittr)
library(dplyr)
library(gridExtra)
library(DGE.Tools)
library(SummarizedExperiment)
library(grDevices)
source("~/R/lib/SubsettableListOfArrays.R")

MyRSE <- readRDS("MyGene_RSE.RDS")
MySLOA <- readRDS("DT_Model_SLOA_dupCor.RDS")
MyContrasts <- readRDS("DT_Model_dupCor_Contrasts.RDS")

#Set some options for drawing heatmaps
heatopts <- list(
               scale="none",  #plotting Logratio data, no Z scaling needed
               show_rownames=FALSE, #usally too many rows to print gene names
               #some font preferences
```

```
                fontsize_col = 18,
                fontsize = 14,
                fontsize_row = 14,
                border_color = NA,
                treeheight_row = 0,
                cluster_cols = FALSE,
                clustering_distance_rows = "correlation",
                #some color gradient options
                #color = colorRampPalette(c("blue", "white", "red"))(100),
                #My current 5 color favorite.
                #Try it with white or black in the middle
                color = colorRampPalette(
                  c("magenta", "blue3", "black", "red", "goldenrod1"))(256)
                #Using ColorBrewer
                #color = colorRampPalette(
                #        rev(brewer.pal(n = 7, name = "RdYlBu")))(100),
)
```

2 Contrast Signature Table from runContrasts

Code Block: Signature Table

```
#print to console
plot.new()
grid.table(MyContrasts$SigCountsSummary)
```

	P<0.01	10% FDR	10% LFDR	P<0.01 & 1.5FC	10% FDR & 1.5FC
RapidVsNorm	1642	1826	1468	342	2
StableVsNorm	1233	1059	926	216	0
RapidVsStable	185	0	0	35	0
TGFb_Norm	7085	9212	8458	1716	1801
TGFb_Rapid	6490	8650	7774	1941	2137
TGFb_Stable	5761	7831	6954	1375	1354
TGFb_RvsTGFb_N	620	6	31	33	1
TGFb_SvsTGFb_N	239	0	0	9	0
TGFb_RvsTGFb_S	102	0	0	7	0

```
#save to PNG
png(file = "ContrastSummaryTable.png", width = 7, height = 7,
    units = "in", res = 300)
plot.new()
grid.table(MyContrasts$SigCountsSummary)
invisible (dev.off())
```

3 JRT_Heatmap

Plot Rapid vs Norm Signature Genes

Prepare the 10% FDR, 2FC signature between Rapid vs. Normal Human Lung Fibroblasts

JRT_heatmap is a convenience wrapper around the CRAN package pheatmap. pheatmap has about 40 settable command line arguments. To make the commands shorter, I've reimplemented the options as a list. The idea is that you typically use the same or similar options for a batch of heatmaps. You can set the options at the top of your script, and then just swap in different data, change the title and plot the heatmap without listing bunch of options each time.

Peek at the source code to see a list of the full set of available parameters or view the help for pheatmap.

Code Block: Heatmap Example

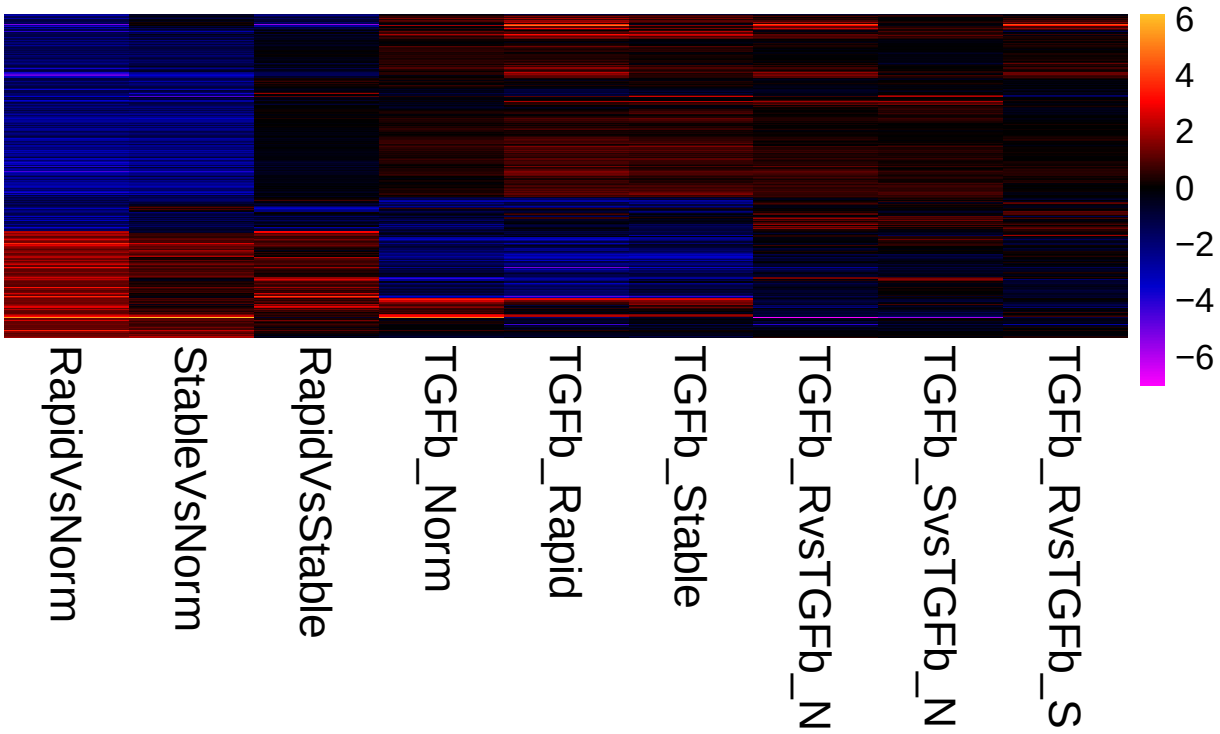
```
#Get the sig genes from the topTreat contrasts,
#then pull the associated logFC data from the topTable contrasts

#Grab a topTreat result: Rapid progressor vs Normal Human lung fibroblast
RvNsig <- MyContrasts$TopTableList$RapidVsNorm
RvNsig$GeneID = rownames(RvNsig)

#filter for FDR < 0.1 and foldchange > 2X
RvNsig <- filter(RvNsig, adj.P.Val <= 0.1, abs(logFC)>=1)

#Get the LogRatios from all 9 contrasts
LogRatios <- extractCol (MyContrasts$TopTableList, "logFC")
#filter to the signature genes defined above
LogRatios <- LogRatios[rownames(LogRatios) %in% RvNsig$GeneID, ]

#Here's a title that dyamically captures the number of genes plotted
heatopts$main <- paste("RvN Signature (10% FDR, 2X FC)\n2D Clusters : ",
                       nrow(LogRatios), " Genes", sep = "")
#print to the console
JRT_heatmap(LogRatios, heatopts)
```




```
#reprint to a PNG file
printopts <- heatopts
printopts$silent <- TRUE
printopts$filename <- "CommonDiseaseSig.png"
JRT_heatmap(LogRatios, printopts)
```

4 Profile Plot (aka MA plot): LogInt vs. LogRatio plots

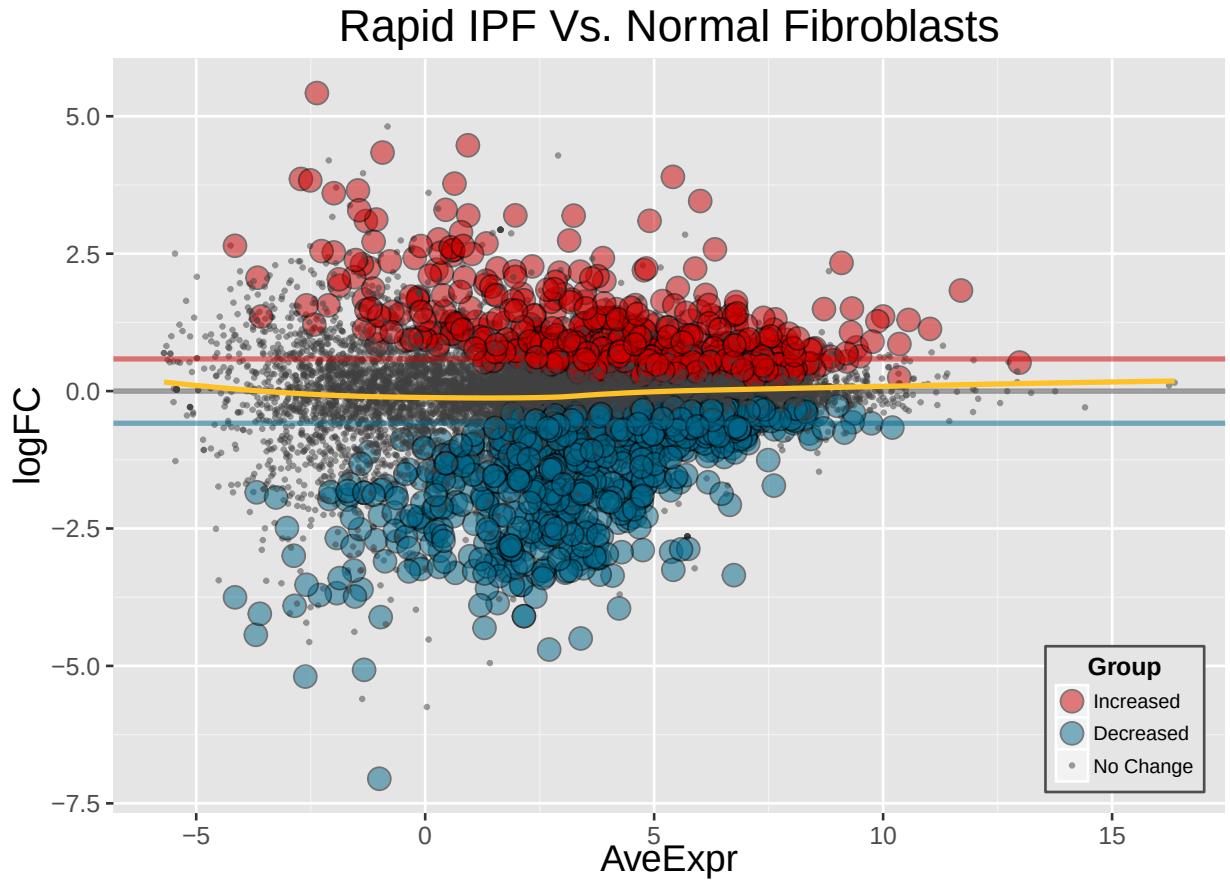
Function profilePlot uses defaults appropriate for topTable/topTreat dataframes.

The printAndSave function sends the plot to the console with a 12pt base font (suitable for knitr output) and save an image file with a 24pt base font (more suitable for PPT use). The file extension determines the image file type (allowed values include: .pdf, .png, .jpg, .tiff, .svg, bmp)

Code Block: Profile Plot Example

```
#grab a topTable dataframe
RvN <- MyContrasts$TopTableList$RapidVsNorm

#draw the plot
MyProfilePlot <- profilePlot(RvN,
                             title="Rapid IPF Vs. Normal Fibroblasts",
                             legendPosition = "se")
printAndSave(MyProfilePlot, "ProfilePlot.PNG")
```



4.1 Profile Plot Arguments & Defaults

- df
- logRatioCol = “logFC”
- logIntCol = “AveExpr”
- pvalCol = “P.Value”
- pthreshold=0.01
- xlab=NULL ylab=NULL title=NULL
- symbolSize = c(4 1 4)
- symbolShape = c(21 20 21)
- symbolColor = c(“black” “grey25” “grey0”)
- symbolFill = c(“red3” “grey25” “deepskyblue4”)
- alpha = 0.5
- sizeBySignificance = FALSE
- referenceLine = “grey25”
- foldChangeLines = log2(1.5)
- refLineThickness = 1
- lineFitType = “loess”
- lineFitColor = “goldenrod1”
- legendPosition = “right”
- baseFontSize = 12
- themeStyle = “grey”

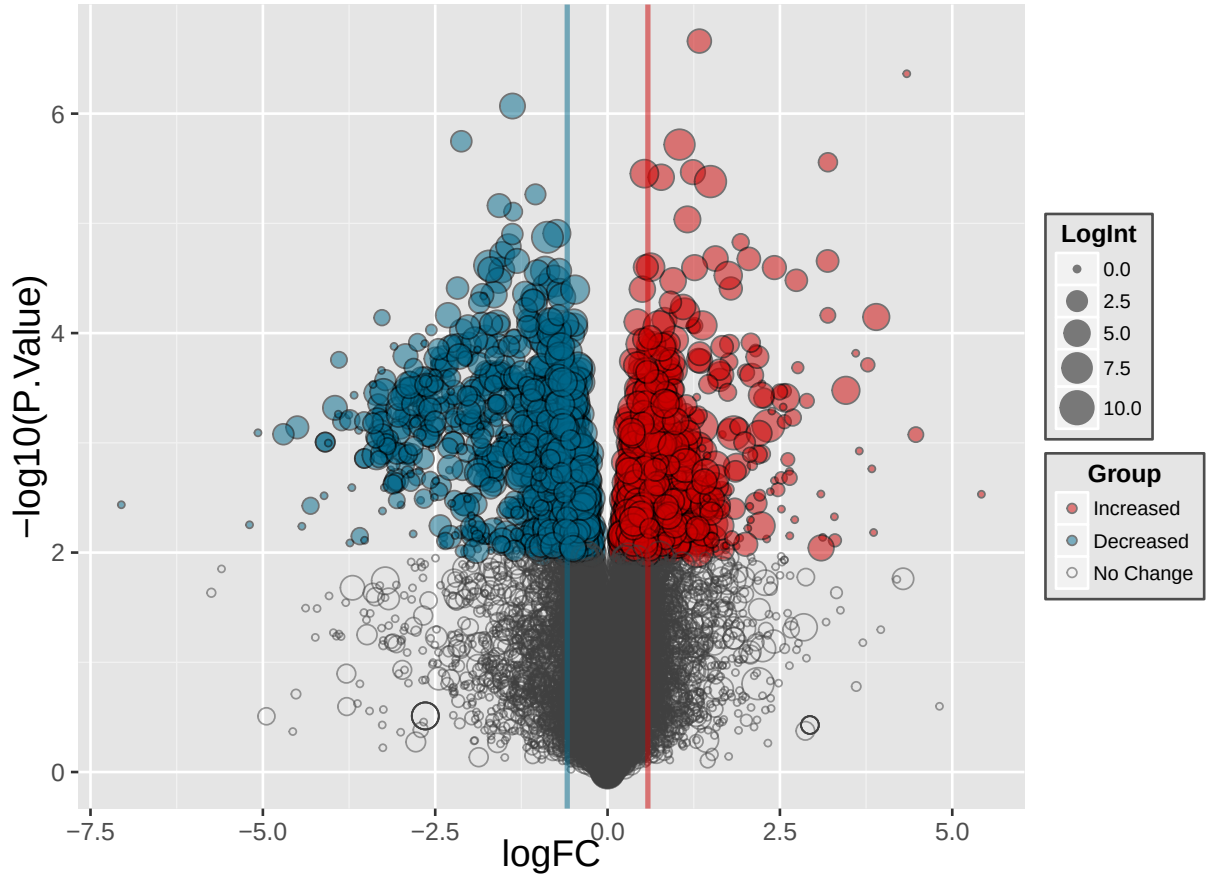
5 Volcano Plot: LogRatio vs. Negative Log Pvalue

Function volcanoPlot uses defaults appropriate for topTable/topTreat dataframes.

Code Block: Volcano Plot Example

```
#grab a topTable dataframe
RvN <- MyContrasts$TopTableList$RapidVsNorm

#draw the plot
MyVolcanoPlot <- volcanoPlot(RvN)
printAndSave(MyVolcanoPlot, "VolcanoPlot.PNG")
```



5.1 Volcano Plot Arguments & Defaults

- df
- logRatioCol = “logFC”
- logIntCol = “AveExpr”
- pvalCol = “P.Value”
- pthreshold=0.01
- xlab=NULL ylab=NULL title=NULL
- symbolSize = c(4 3.999 4)
- symbolShape = c(21 1 21)
- symbolColor = c(“black” “grey25” “grey0”)
- symbolFill = c(“red3” “grey25” “deepskyblue4”)
- alpha = 0.5
- sizeByIntensity = TRUE
- pthresholdLine = NULL
- foldChangeLines = log2(1.5)
- refLineThickness = 1
- legendPosition = “right”
- baseFontSize = 12
- themeStyle = “grey”

6 CDF Plot: Evaluate model performance

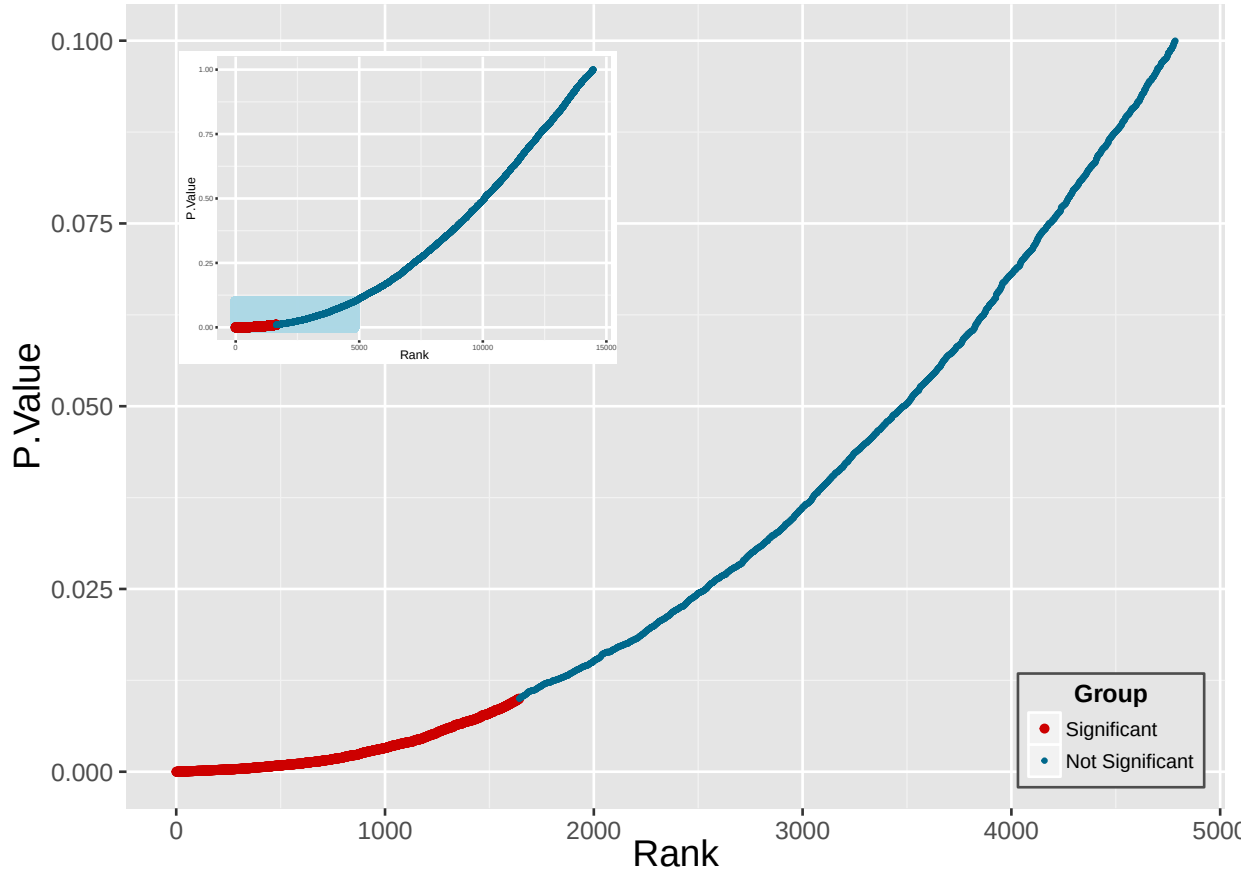
Plot the pvalue Rank vs. the actual pvalues. A straight diagonal line indicate no differential genes (null hypothesis satisfied), while differential genes appear as a break above the line at low pvalues.

The main plot shows pvalues <= 0.1 (user settable). An inset plot shows the full range of pvalues and a blue box indicates the region shown in the main plot. Genes with p<0.01 (user settable) are shown in red.

Code Block: CDF Plot Example

```
#grab a topTable dataframe
RvN <- MyContrasts$TopTableList$RapidVsNorm

#draw the plot
MyCDFPlot <- cdfPlot(RvN, plotFile="MyCdfPlot.PNG")
```



Note: printAndSave doesn't work with cdfPlot because a cdfPlot is really two plots. Therefore, cdfPlot has an argument to enable saving the plot to an image file.

6.1 CDF Plot Arguments & Defaults

- df
- pvalCol = “P.Value”
- pthreshold=0.01
- xlab=NULL ylab=NULL title=NULL insetTitle=NULL
- symbolSize = c(2 1)
- symbolShape = c(20 20)
- symbolColor = c(“red3” “deepskyblue4”)
- symbolFill = c(“red3” “deepskyblue4”)
- alpha = 1
- referenceLine = NULL
- refLineThickness = 1
- legendPosition = “se”
- baseFontSize = 12
- viewportX = 0.15
- viewportY = 0.93
- viewportWidth = 0.35
- themeStyle = “grey”
- pvalMax = 0.10
- printPlot = TRUE
- plotFile = NULL

7 Compare Plot: compare two signature

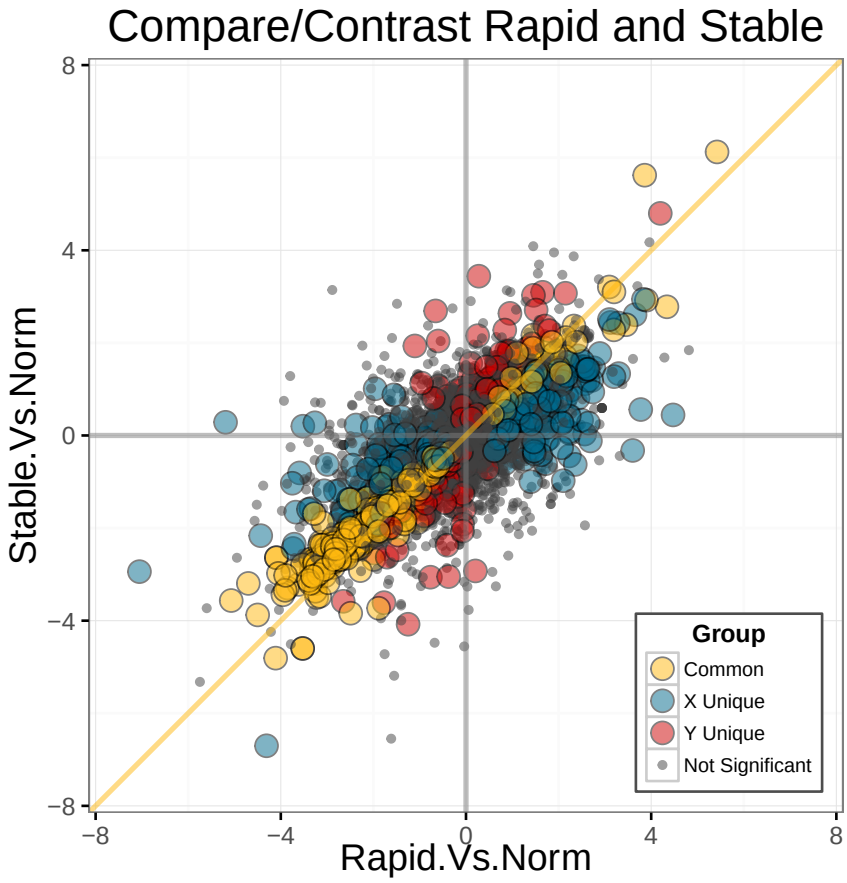
Plot the LogRatios of two contrasts, highlight common genes and genes unique to either contrast.

Code Block: Compare Plot Example

```
#Get two topTreat dataframes
RvN <- MyContrasts$TopTableList$RapidVsNorm
SvN <- MyContrasts$TopTableList$StableVsNorm

#combine the LogRatios and pvalues into a dataframe
compareData <- data.frame(cbind(Rapid.Vs.Norm = RvN$logFC,
                    Stable.Vs.Norm = SvN$logFC,
                    xp = RvN$P.Value,
                    yp = SvN$P.Value) )

MyComparePlot = comparePlot(compareData,
                            title = "Compare/Contrast Rapid and Stable",
                            legendPosition = "se")
printAndSave (MyComparePlot, "ComparePlot.PNG")
```



7.1 Compare Plot Arguments & Defaults

- df pthreshold=0.01
- xlab=NULL ylab=NULL title=NULL
- symbolSize = c(4 4 4 2)
- symbolShape = c(21 21 21 20)
- symbolColor = c(“black” “grey0” “grey1” “grey25”)
- symbolFill = c(“darkgoldenrod1” “deepskyblue4” “red3” “grey25”)
- alpha = 0.5
- crosshair=“grey50”
- referenceLine=“darkgoldenrod1”
- refLineThickness = 1
- dens2D = TRUE
- legendPosition = “right”
- baseFontSize = 12
- themeStyle = “bw”

8 Observation Plot: Show how individual genes behave across treatment groups

By default the obsPlot function displays each observation (Gene) in a separate plot with each treatment group plotted separately. By default, three layers are displayed: Boxplot, individual points and a square at the mean position. Optionally, a violin can be displayed and any or the other layers can be turned off as desired.

The plot is faceted by default. Setting facet = FALSE produces individual plots for each gene that are returned as a list of plots.

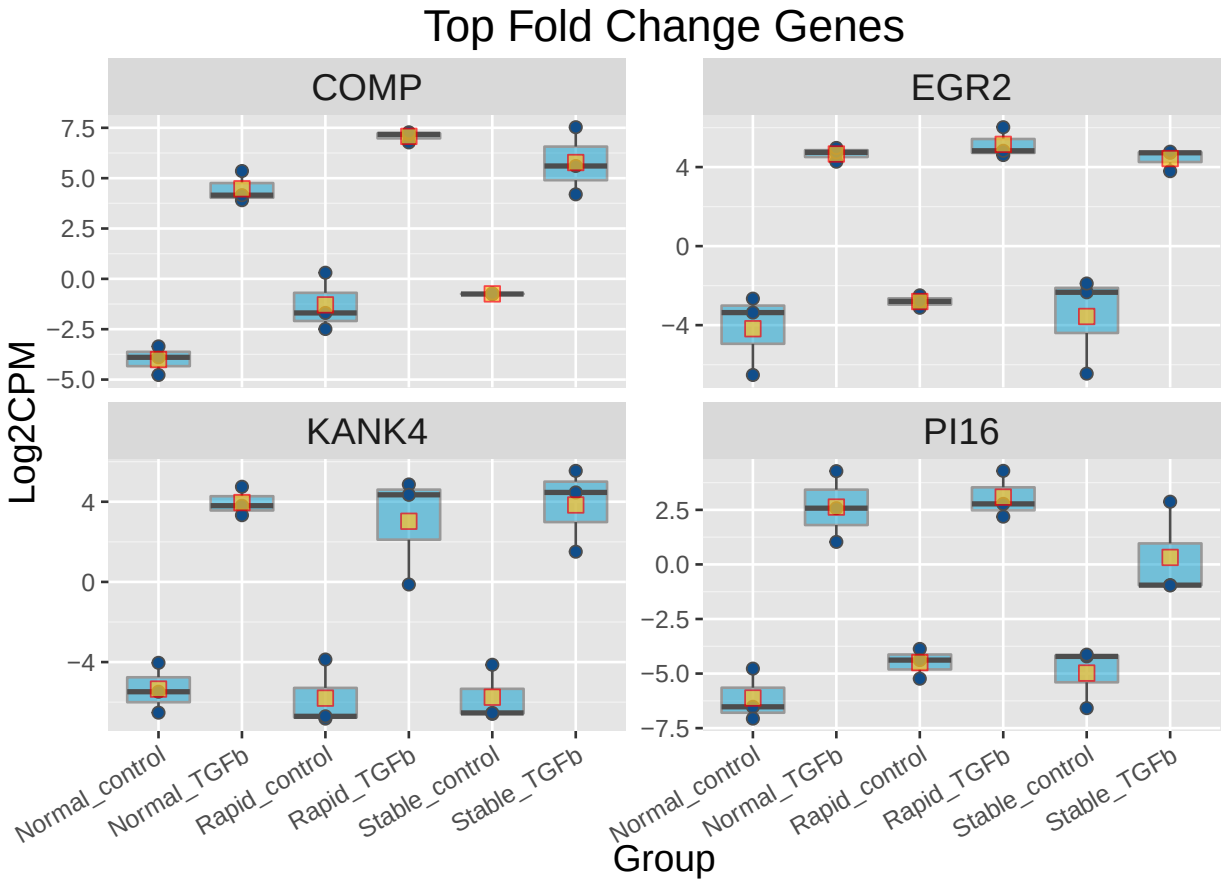
Code Block: Observation Plot Example

```
#Plot intensity data for Top 4 foldchanges with TFGb treatment

#Get the signature genes to plot from topTable data
TGFb_Norm <- MyContrasts$TopTableList$TGFb_Norm
#Stick the geneID into a column (because next step clobbers rownames)
TGFb_Norm$GeneID = rownames(TGFb_Norm)
#Filter for top signficant fold changes and sort descending on foldchange
TGFb_Norm %<>% filter(adj.P.Val < 0.1) %>% arrange(desc(logFC))
#keep the top 4 genes
MyGenes  <- TGFb_Norm$GeneID[1:4]

#Now get the normalize Log2CPM data for all samples from the
#SummarizedListOfArrays object
Log2CPM <- MySLOA$Log2CPM
#filter to our gene
Log2CPM <- Log2CPM[rownames(Log2CPM) %in% MyGenes,]
#trim the Omicsoft suffix from the gene symbols
rownames(Log2CPM) <- OmicsoftRefGeneID2GeneSym(rownames(Log2CPM))

#set up the blocking var to define treatment groups
#the ReplicateGroup field in the Design table should be appropriate for this
Design = colData(MyRSE) %>% as.data.frame
block <- Design$ReplicateGroup
#facet plot
obsPlot (Log2CPM, block, title = "Top Fold Change Genes")
```


Top Fold Change Genes
COMP
EGR2
KANK4
PI16
Log2CPM
Group
Normal_control
Normal_TGFb
Rapid_control
Rapid_TGFb
Stable_control
Stable_TGFb

8.1 Observation Plot Arguments & Defaults

- data #dataframe
- block = NULL
- obsNames = NULL #e.g. rownames(data)
- sampNames = NULL
- plotBy = “Gene” #separate plot for each of these
- valType = “Log2CPM” #value being plotted
- boxLayer = TRUE
- violinLayer = FALSE
- pointLayer = TRUE
- meanLayer = TRUE
- xlab=NULL ylab=NULL title=NULL
- boxColor = “grey30”
- boxFill = “deepskyblue3”
- boxAlpha = 0.5
- violinColor = “grey30”
- violinFill = “goldenrod1”
- ViolinAlpha = 0.5
- pointColor = “grey30”
- pointFill = “dodgerblue4”
- pointShape = 21 #fillable circle
- pointAlpha = 1
- pointSize = 2
- pointJitter = 0
- meanColor = “red2”
- meanFill = “goldenrod1”
- meanShape = 22 #fillable square
- meanAlpha = 0.7
- meanSize = 3
- legenPosition = “right”
- baseFontSize = 12
- themeStyle = “grey”
- facet = TRUE
- facetCol = NULL

- xAngle = 30 #rotate x axis labels
- scales = “free_y” #independent y axes for each plot

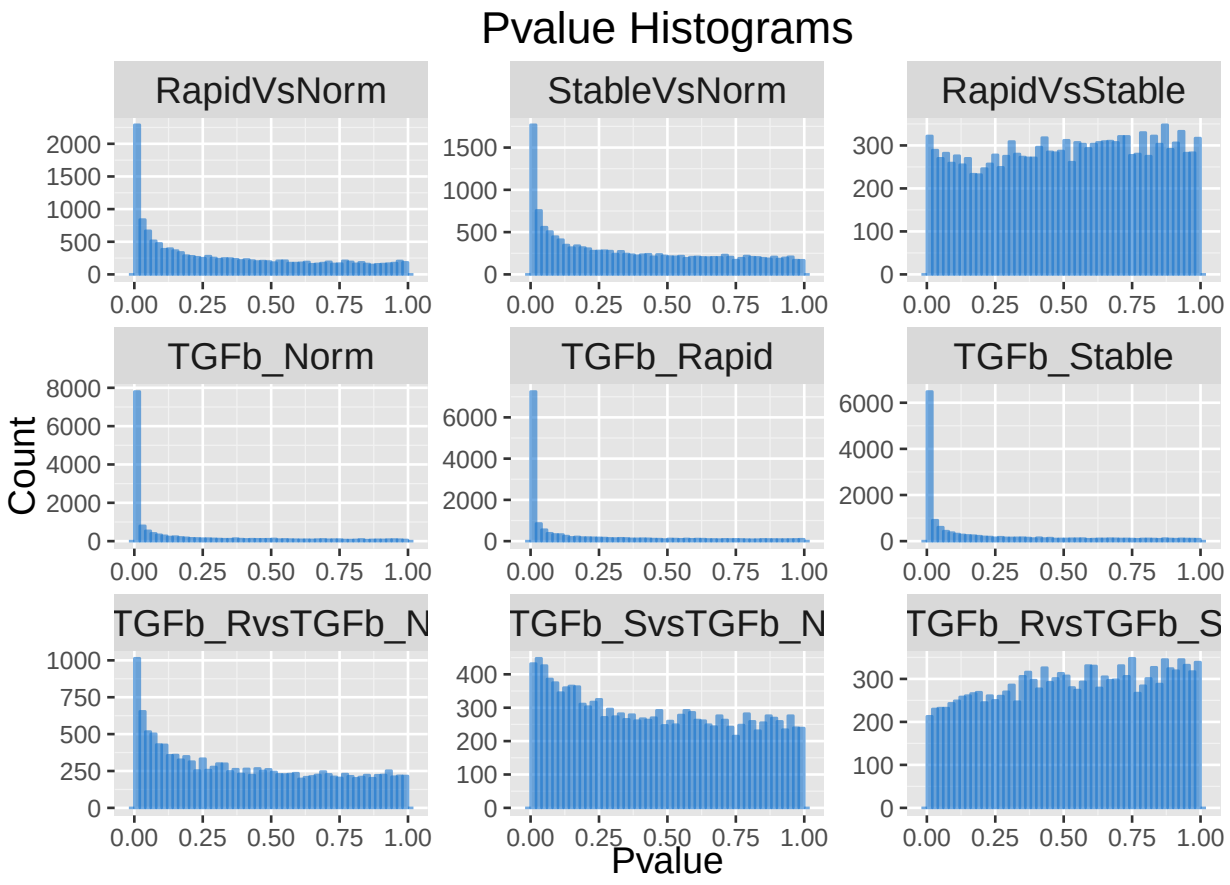
9 Pvalue Histogram: Evaluate Quality of Fit

Plot the LogRatios of two contrasts, highlight common genes and genes unique to either contrast.

Code Block: Pvalue Histogram Facet Plot Example

```
#Get all the pvalue columns from 9 contrasts (topTable dataframes)
MyPval <- extractCol(MyContrasts$TopTableList, "P.Value")

plotPvalHist(MyPval)
```



9.1 plotPvalHist Arguments & Defaults

- P.Val Facet = TRUE
- savePlot = FALSE
- fileNames = NULL
- binWidth = 0.02
- baseFontSize=12
- facetFontSize=NULL
- alpha = 0.6
- color = "dodgerblue3"
- fill = "dodgerblue3"

10 Custom Scaled Themes

All of these plotting tools employ a relative font size feature that Don Jackson brought to my attention. Each plot tool has a baseFontSize argument that refers to the size of the font used to draw the scales on the plot. The X and Y labels are set to 1.2X and the main title is set to 1.5X bold. The plot functions default to a baseFontSize = 12 which is suitable for knitr output. The printAndSave function automatically scales the baseFontSize to 24pt to save an image file.

Several themes have been created to take advantage of the scalable fonts and can be used with your own custom plots. These themes are actually functions that apply a theme and take a baseFontSize as an argument. The default theme for ggplot2 is called theme_grey and takes a single argument that specifies the base font size for the figure. I cloned that theme and adjusted the relative font sizes for elements as described above. The new theme is called "greyTheme". The greyTheme is applied just like theme_grey:

```
Myplot <- MyPlot + greyTheme(18)  #sets baseFontSize to 18pt

MyPlot <- MyPlot + greyTheme(12)  #adjusts all font to new setting.
```

ggplot also includes a second theme named theme_bw which is a clone of theme_grey with the grey background removed. Analogously, I've defined that counterpart to greyTheme named bwTheme.

Typically, a theme is added last to your plot. theme_grey, and by extension greyTheme, reset the legend position to "right". Thus these themes can be annoying if you're moving the legend around. Another theme has been designed that modifies just the baseFontSize and nothing else and thus won't mess up your highly customized plot format.

Function printAndSave relies on these themes based on relative font size settings. Thus printAndSave should be used in conjunction with greyTheme

```
MyPlotWithCustomizeLegend <- MyPlotWithCustomizeLegend + baseFont(18)
```

11 DGE.Tools Installation from GitHub

The DGE.Tools and zFPKM packages have been deposited in the BMS Github and can be installed with the following commands:

```
install_git("http://biogit.pri.bms.com/thompj27/zFPKM")
install_git("http://biogit.pri.bms.com/thompj27/DGE.Tools")
library(DGE.Tools)
Install.DGE.Tools.Dependencies()
```

There is still one critical piece of R source that breaks when encapsulated in a package. The SubsettableListOfArrays.R file still needs to be manually copied to your drive and sourced whenever you are manipulating SubsettableListOfArray objects (i.e. the output of runEdgeRNorm and runVoom functions).

Download SubsettableListOfArrays.R and place it somewhere on your computer. For example, I use "~/R/lib/" for my personal source library and source the file with:

```
source("~/R/lib/SubsettableListOfArrays.R")
```

Manual Installation: If the BMS github is down, or the git install is failing, you can also install manually by downloading these two files: DGE.Tools.tar.gz and zFPKM.tar.gz from the biohtml site:

http://bioinformatics.bms.com/active/biohtml/thompj27/DGE.Tools/DGE.Tools.tar.gz
http://bioinformatics.bms.com/active/biohtml/thompj27/DGE.Tools/zFPKM.tar.gz

Then type these commands to install (where "~/R/lib" is the folder you downloaded to):

```
install.packages("~/R/lib/zFPKM.tar.gz", repos=NULL, type="source")
install.packages("~/R/lib/DGE.Tools.tar.gz", repos=NULL, type="source")
library(DGE.Tools)
Install.DGE.Tools.Dependencies()
```

Holler if you have any trouble with the installation. We're still trying to figure out the most painless way to distribute R packages internally.

12 Session Info

Time required to process this report: *1.094643 mins*

R Session Info

Section	Name	Value
System	sysname	Windows
System	release	7
System	version	build 7601, Service Pack 1
System	nodename	PF029MHB
System	machine	x86
System	login	thompj27
System	user	thompj27
System	effective_user	thompj27
System	Directory	C:/Users/thompj27/Documents/R/lib/pkgsrc
R	Version	R version 3.2.3 (2015-12-10)
Packages	knitr	1.11 CRAN CRAN 2015-08-14
Packages	envDocument	2.1.1 BMS BMS 2015-10-24
Packages	SummarizedExperiment	1.0.2 NA NA NA
Packages	Biobase	2.30.0 NA NA NA
Packages	GenomicRanges	1.22.2 NA NA NA
Packages	GenomeInfoDb	1.6.1 NA NA NA
Packages	IRanges	2.4.6 NA NA NA
Packages	S4Vectors	0.8.5 NA NA NA
Packages	BiocGenerics	0.16.1 NA NA NA
Packages	DGE.Tools	1.0.4 NA NA NA
Packages	gridExtra	2.0.0 CRAN CRAN 2015-07-14 20:40:38
Packages	dplyr	0.4.3 CRAN CRAN 2015-09-01 18:15:02
Packages	magrittr	1.5 CRAN CRAN 2014-11-22 19:15:57
Packages	ggplot2	2.0.0 CRAN CRAN 2015-12-18 10:45:15
Packages	BiocInstaller	1.20.1 NA NA NA
Script	Path	C:/Users/thompj27/Documents/R/lib/pkgsrc/DGE.ToolsPlotGallery.Rmd
Script	Modified	2016-01-12 14:21:23