DGE.Tools Data Exploration Plotting Tools

John Thompson

13Feb2019

Contents

1 DGE.Tools2: Plotting tools

DGE.Tools2 includes an assortment of standard data exploration plotting tools. The intention here is to make common plot types dead easy to produce and provide a consistent look and feel. Generally, optional argument allow you to tweak the plots and since the plots are based on ggplot, you can further customize the resulting ggplot objects.

The Plotting functions are:

- profilePlot: LogInt vs. LogRatio (aka MA plot)
- volcanoPlot: LogRatio vs. NegLogP
- cdfPlot: Rank(pvalue) vs. NegLogP
- comparePlot: Compare LogRatios for two contrasts
- obsPlot2: Gene intensity boxplots (faceted)
- logRatioPlot: Plot logRatios +/- 95% confidence intervals
- checkSex: Plot X and Y chromosome genes to infer sex
- plotDisp: Plot dispersion or BCV vs log intensity (type of QC plot)
- plotNorm: Plot density or boxplots before/after normalization
- QCplots: Plot selected alignment QC metrics
- plotPvalHist: faceted Pvalue histogram
- ggplotMDS: Multidimensional Scaling (like PCA but uses a intensity-based distance metric)
- MDS_var_explained: Plot the % variance explained by each MDS component

Most of the plots are generated with ggplot, thus, the user can take a returned ggplot object and further modify it to their heart's content.

Code Block: Load some test data (IPF Fibroblast data).

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

setwd("~/R/lib/pkgsrc/DGE.Tools2/vignettes")
library(tidyverse)
library(magrittr)
library(DGEobj)
library(DGE.Tools2)
library(JRTutil)

outputPath <- "./output"

#get a dataset to work with from Stash
dgeObj <- getRDSobjFromStash("UCSD_Lung_Fibroblasts_P-20171107-0002_8Feb2018.RDS")

#apply a low intensity filter
```

```
dgeObj <- lowIntFilter(dgeObj, zfpkmThreshold=-3, countThreshold=10)
```

2 Table of Differential Gene Counts from Contrasts

Code Block: Signature Table

```
#grab a topTable dataframe
myContastList <- getType(dgeObj, "topTable")

#Plot with defaults
df <- summarizeSigCounts(myContastList)
knitr::kable(df)
```

	P.Value	adj.P.Val	Qvalue	qvalue.lfdr	ihw.adj_pvalue
TGFb10_vs_Veh	2381	4395	6588	2147	5000
TGFb25_vs_Veh	5495	7586	10174	5824	8156
IPF_vs_Normal	1146	212	753	416	819

```
#add a fold change threshold
df <- summarizeSigCounts(myContastList, fcThreshold = 2)
knitr::kable(df)
```

	P.Value	adj.P.Val	Qvalue	qvalue.lfdr	ihw.adj_pvalue
TGFb10_vs_Veh	686	942	1096	646	994
TGFb25_vs_Veh	2092	2400	2600	2154	2463
IPF_vs_Normal	239	70	171	114	178

Arguments to summarizeSigCounts allow you to specify which fields to include and define the thresholds for significance. see ?summarizeSigcounts

3 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/pdf output) and saves an image file with a 24pt base font which is 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
myTopTable <- dgeObj$TGFb25_vs_Veh

#draw the plot
MyProfilePlot <- profilePlot(myTopTable,
                             title="TGFbeta Signature",
                             legendPosition = "ne")
printAndSave(MyProfilePlot, file.path(outputPath, "ProfilePlot.PNG"))
```

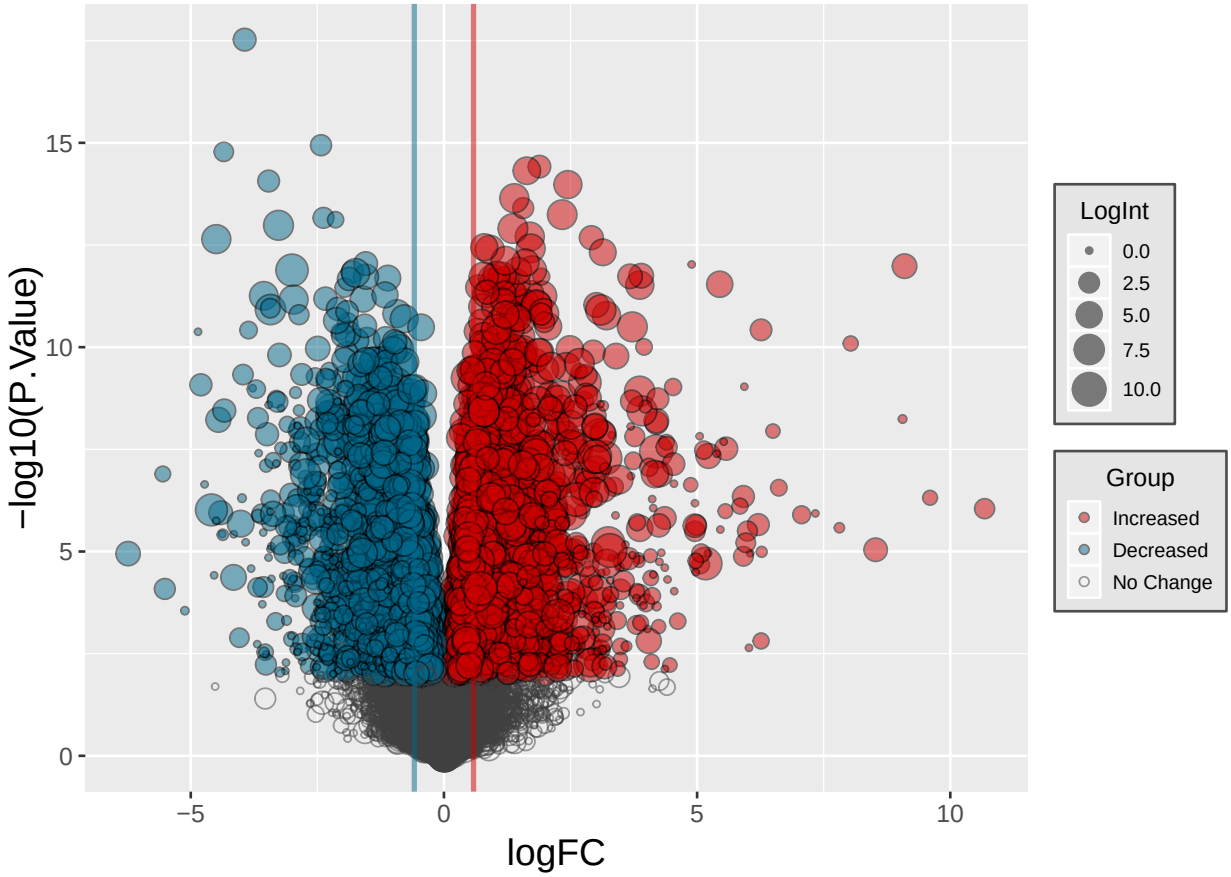
See ?profilePlot for options to modify various attributes (color, shape, tranparency, reference lines, etc)

4 Volcano Plot: LogRatio vs. Negative Log Pvalue

Function volcanoPlot uses defaults appropriate for topTable dataframes.

Code Block: Volcano Plot Example

```
#grab a topTable dataframe
myTopTable <- dgeObj$TGFb25_vs_Veh

#draw the plot
MyVolcanoPlot <- volcanoPlot(myTopTable)
printAndSave(MyVolcanoPlot, file.path(outputPath, "VolcanoPlot.PNG"))
```



See ?volcanoPlot for options to modify various attributes (color, shape, tranparency, reference lines, etc)

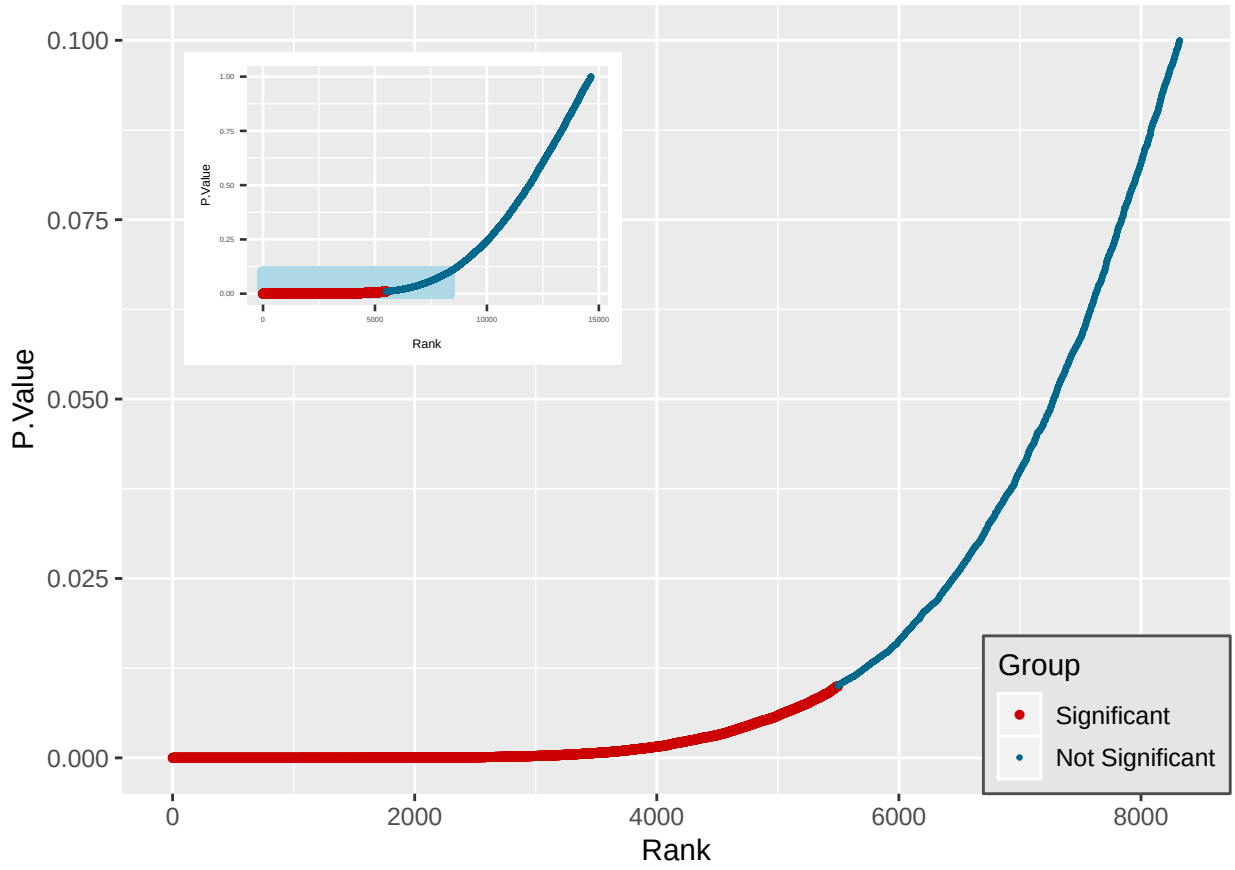
5 CDF Plot: Evaluate model performance

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
myTopTable <- dgeObj$TGFb25_vs_Veh

#draw the plot
MyCDFPlot <- cdfPlot(myTopTable, plotFile=file.path(outputPath, "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.

See ?cdfPlot options to modify various attributes (color, shape, tranparency, reference lines, etc)

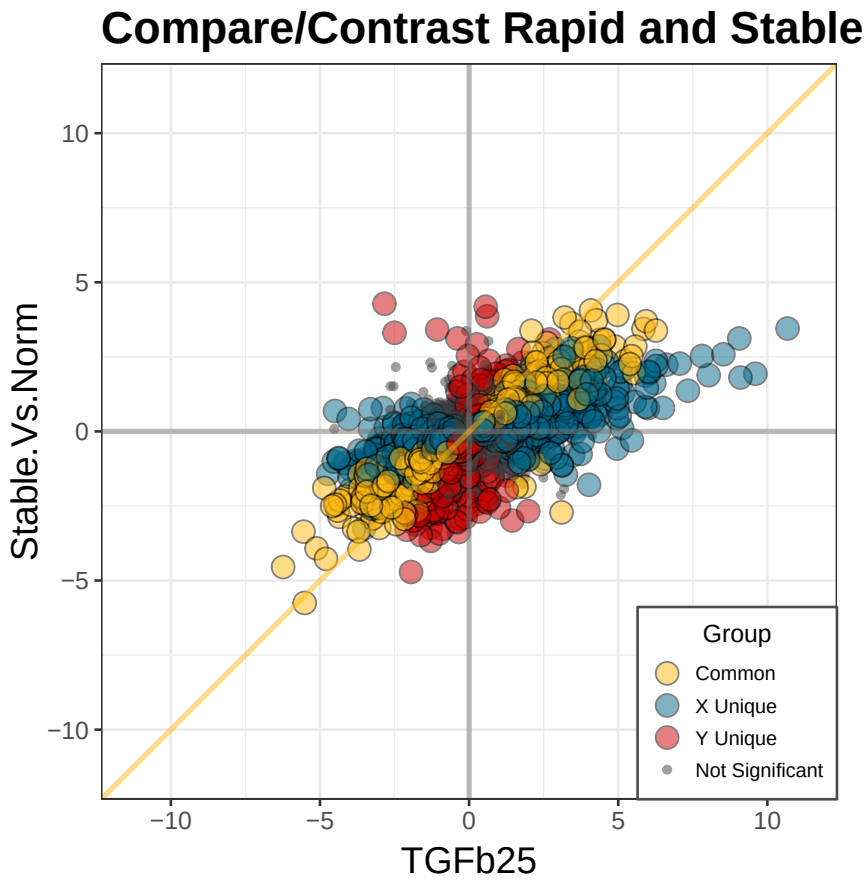
6 Compare Plot: compare two signature

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

The input for this plot requires pulling the log ratio and pvalue columns from two topTable dataframes together.

Code Block: Compare Plot Example

```
#Get two topTreat dataframes
#grab a topTable dataframe
tt1 <- dgeObj$TGFb25_vs_Veh
tt2 <- dgeObj$TGFb10_vs_Veh

#combine the LogRatios and pvalues into a dataframe
compareData <- data.frame(cbind(TGFb25= tt1$logFC,
                    Stable.Vs.Norm = tt2$logFC,
                    xp = tt1$P.Value,
                    yp = tt2$P.Value) )

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



See ?comparePlot options to modify various attributes (color, shape, tranparency, reference lines, etc)

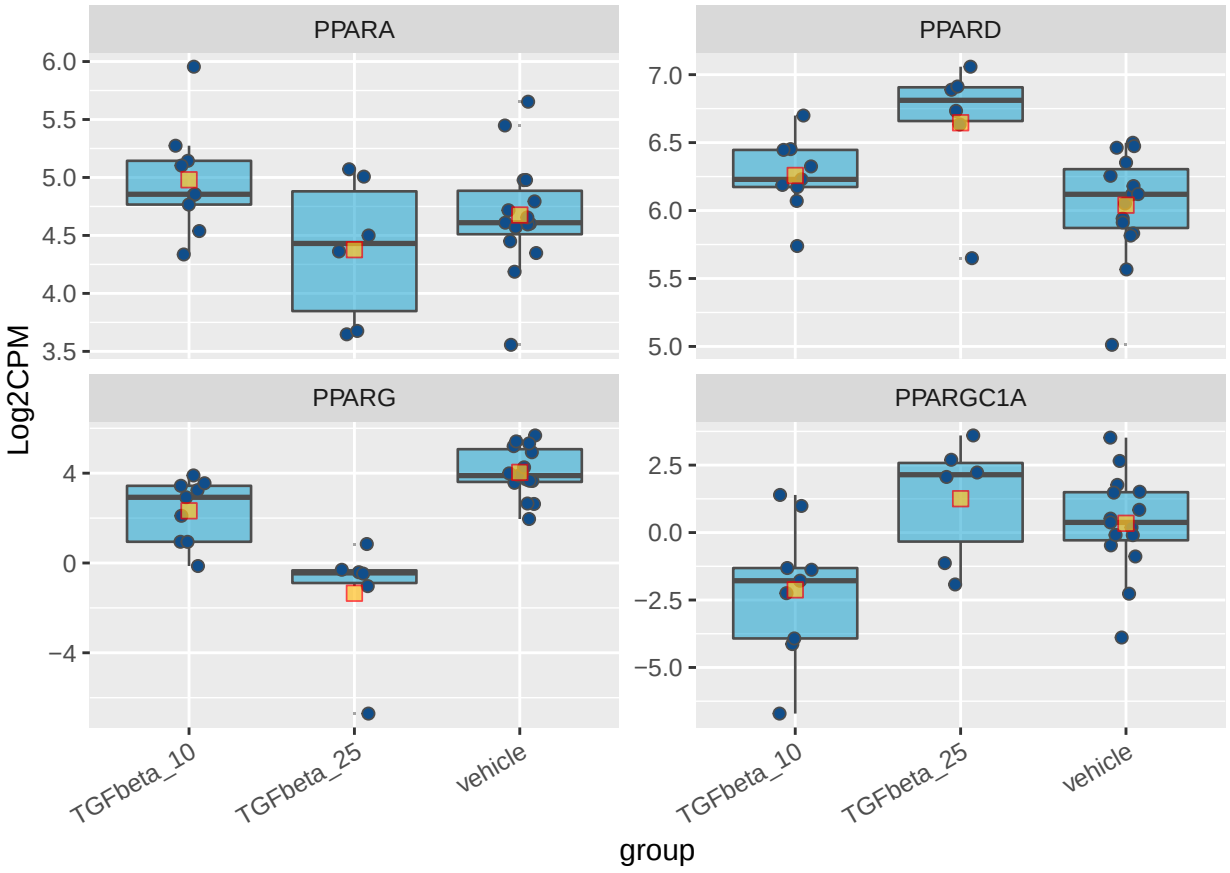
7 Observation Plot: Intensity boxplots by gene

This is a gene of interest plot, intended to plot a reasonable small number of genes, a separate plot for each gene with a box for each treatment group. It takes a tidy data frame as input and a tidyIntensity function is supplied to simplify creation of the input file.

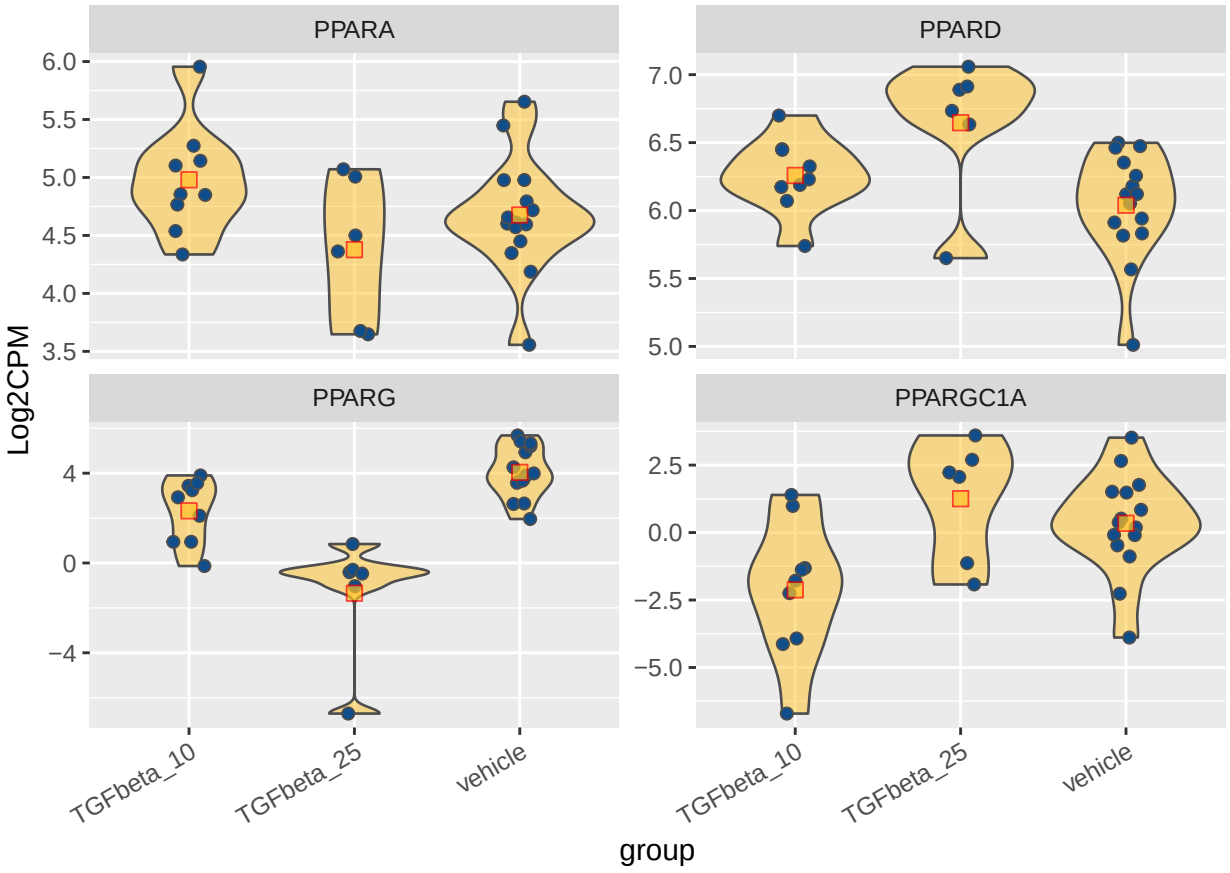
The plot is faceted by default. Setting facet = FALSE produces individual plots for each gene that are returned as a list of plots.

Code Block: Observation Plot Example

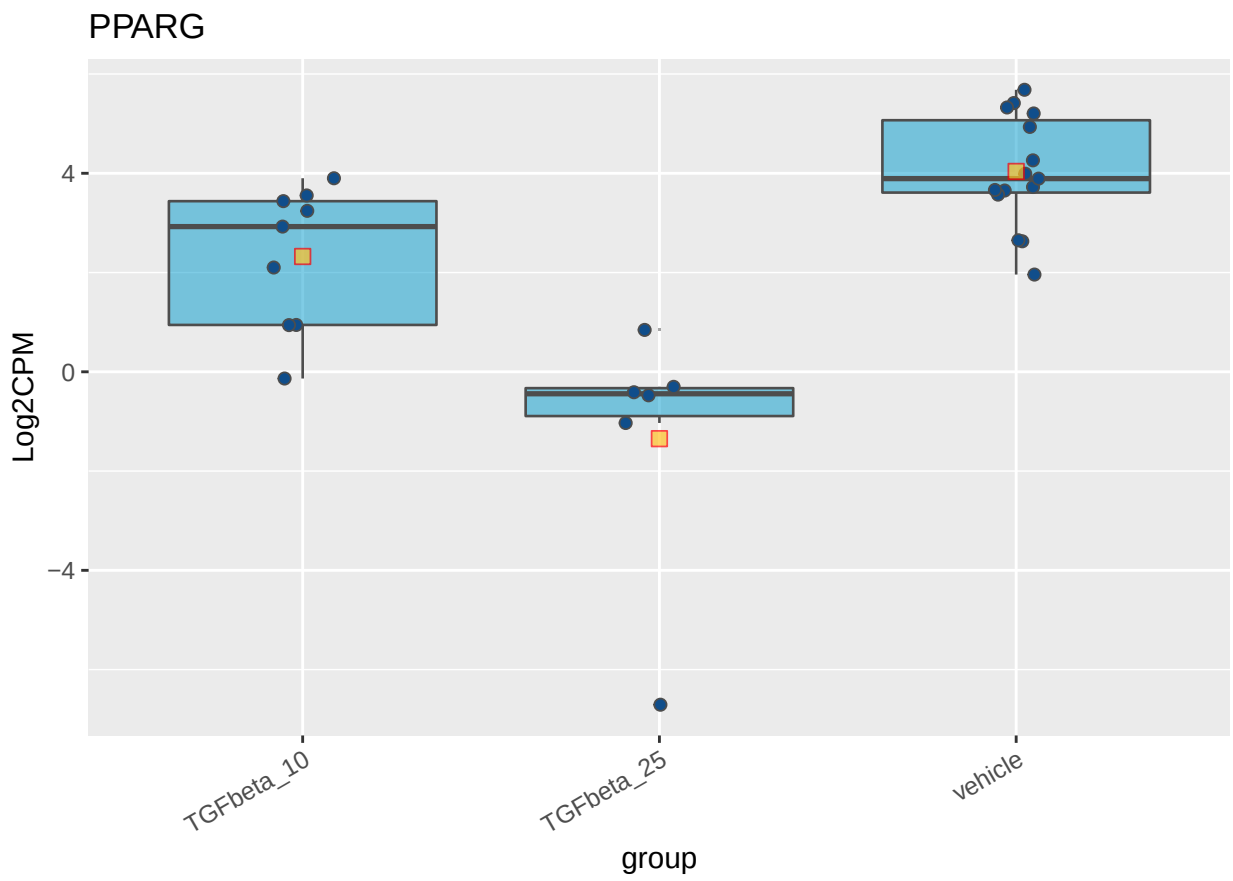
```
#get log2cpm data
log2cpm <- convertCounts(dgeObj$counts, unit="cpm", log=TRUE, normalize="tmm")
#filter for a smallish set of genes
idx <- stringr::str_detect(dgeObj$geneData$GeneName, "^PPAR")
log2cpm <- log2cpm[idx,]
#swap gene symbols for Ensembl IDs
rownames(log2cpm) <- dgeObj[idx,]$geneData$GeneName

#put in tidy format
tidyInt <- tidyIntensity(log2cpm,
                         rowidColname="GeneID",
                         keyColname="Sample",
                         valueColname="Log2CPM",
                         group=dgeObj$design$ReplicateGroup)

#Facetted boxplot
obsPlot2(tidyInt, plotByCol="GeneID",
         groupCol = "group",
         valueCol ="Log2CPM",
         pointJitter = 0.1,
         facetRow = 2)
```




```
#Facetted violin plot
obsPlot2(tidyInt, plotByCol = "GeneID",
         violinLayer = TRUE,
         boxLayer = FALSE,
         groupCol="group",
         valueCol = "Log2CPM",
         pointJitter = 0.1,
         facetRow = 2)
```




```
#return a list of individual plots
myplots <- obsPlot2(tidyInt, plotByCol="GeneID",
                    groupCol = "group",
                    valueCol ="Log2CPM",
                    pointJitter = 0.1,
                    facet = FALSE)
#plot the first one
printAndSave(myplots[[1]], file.path(outputPath, "obsPlot1.png"))
```

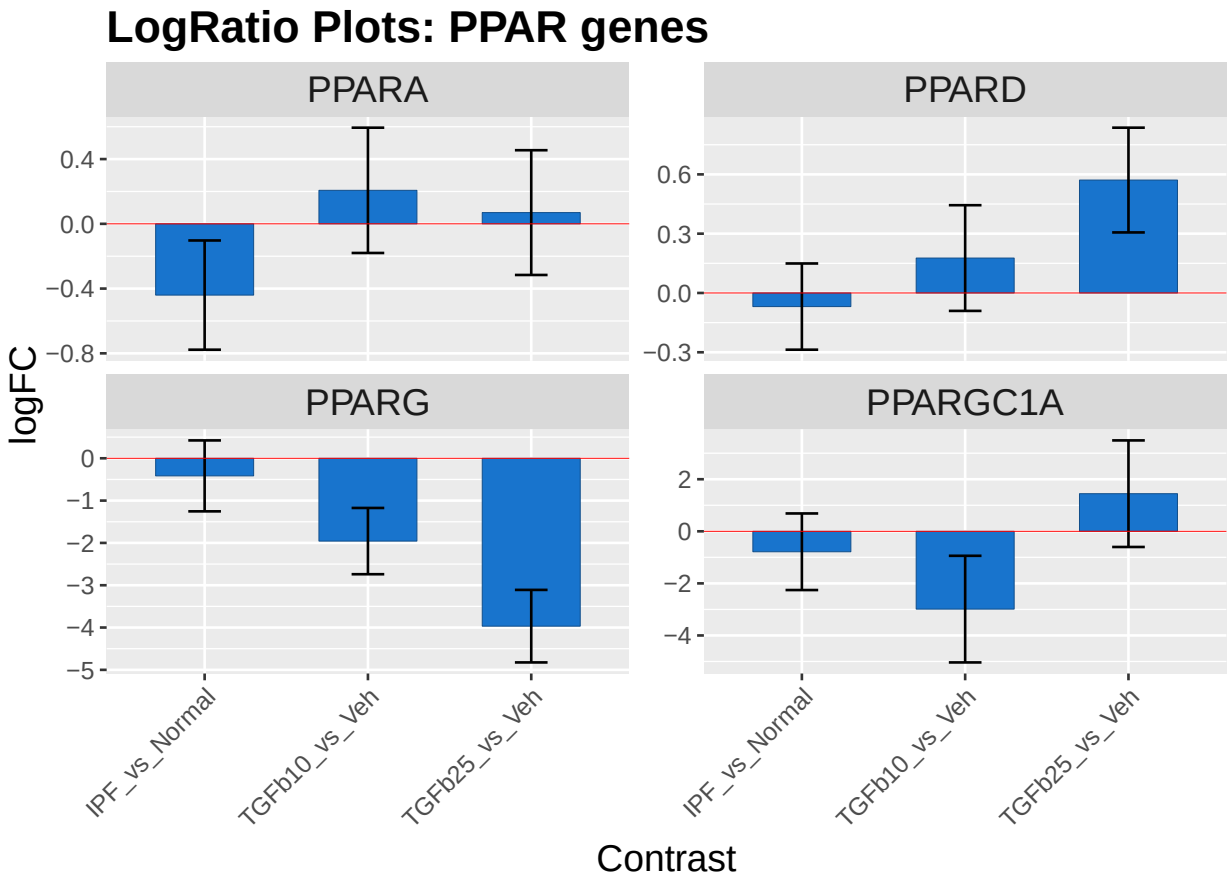


See ?obsPlot2 options to modify various attributes (color, shape, tranparency, reference lines, etc)

8 logRatioPlot

Plotting logRatio data with 95% confidence limits is generally preferable to plotting intensity data with the Observation Plot. This is because the ratio data is post-modeling and thus reflects adjustments for any "nusiance" factors included in the modeling (e.g. batch effect, demographic or sex effects).

```
#Put contrasts in tidy format keeping logFC, and confidence limits data
tidyDat <-tidyContrasts(dgeObj, rownameColumn="EnsgID", includeColumns = c("logFC", "CI.R", "CI.L"))

#add gene symbols from geneData
ens2genesym <- dgeObj$geneData %>%
               rownames_to_column(var="EnsgID") %>%
               select(EnsgID, GeneSymbol=GeneName)
tidyDat <-  left_join(tidyDat, ens2genesym)


#filter for a small set of genes of interest
idx <- stringr::str_detect(tidyDat$GeneSymbol, "^PPAR")
tidyDat <- tidyDat[idx,]

#simple barplot
myLogRatioPlot <- logRatioPlot(tidyDat,
                               facetColname = "GeneSymbol",
                               xColname = "Contrast",
                               title = "LogRatio Plots: PPAR genes",
                               facetCol = 2,
                               barWidth = 0.6
)

printAndSave(myLogRatioPlot, file.path(outputPath, "logRatioPlot.png"))
```



The companion function, tidyContrasts, aids in reformatting topTable contrast data for use with function logRatioPlot. tidyContrasts will accept a DGEobj as input and will find all topTable data in the DGEobj. tidyContrasts will also work with a list of topTable dataframes which provides flexibility to mix and match topTable data from multiple projects for plotting.

See ?logratio options to modify various attributes (color, shape, tranparency, reference lines, etc)

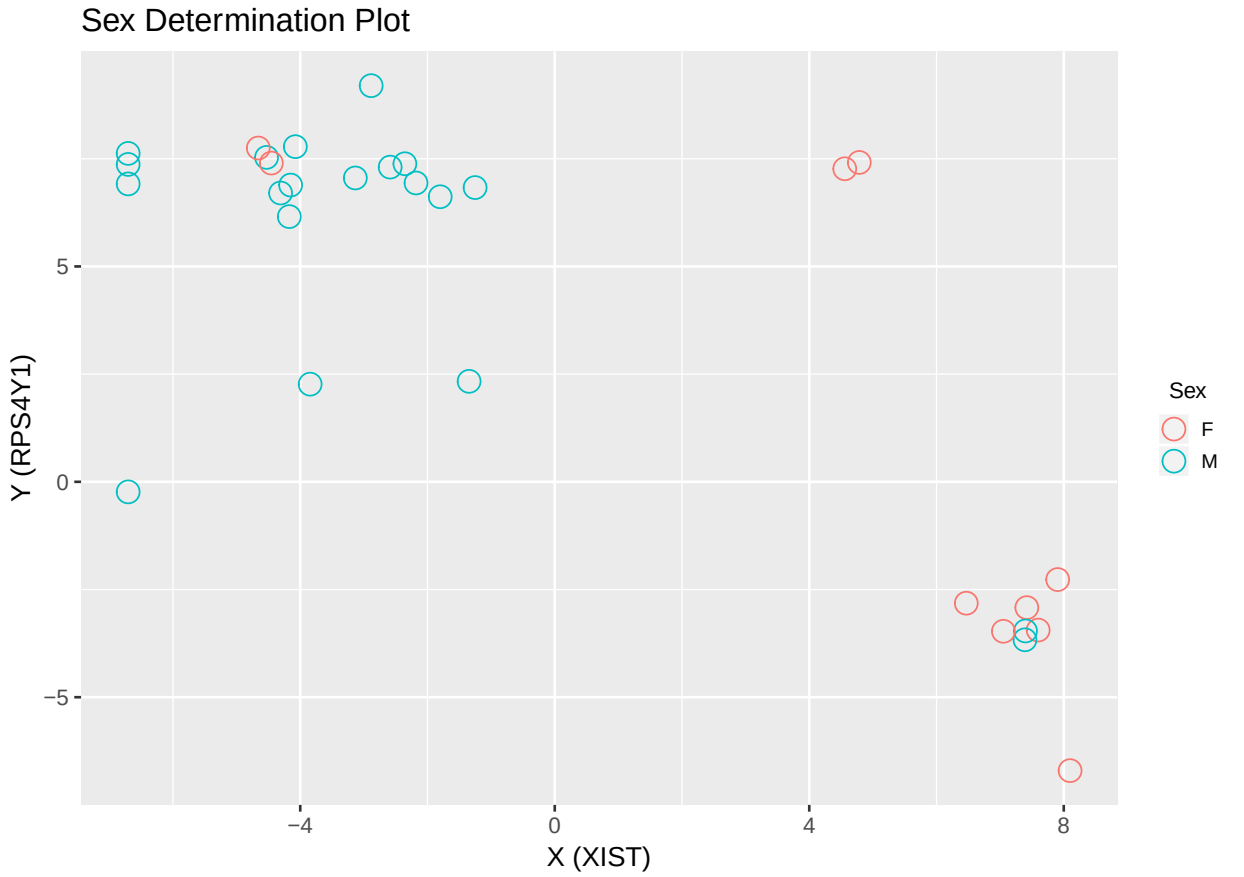
9 checkSex Plot

The checkSex function uses expression intensity data from the X and Y chromosomes to infer sex. When compared to actuall sex annotation, this can help to identify sample labeling problems and swaps.

The XIST gene us used as an X-linked reporter gene that is well expressed in most tissues. Since most Y-linked genes are expressed predominantly in testes, we scan for and use the highest expressed Y-linked gene for the analysis.

You can provide the name of the annotated sex column in the design table and it will be used to color code the points.

```
sexPlot <- checkSex(dgeObj, species="human", sexCol="Sex")
printAndSave(sexPlot, file.path(outputPath, "sexPlot.png"))
```

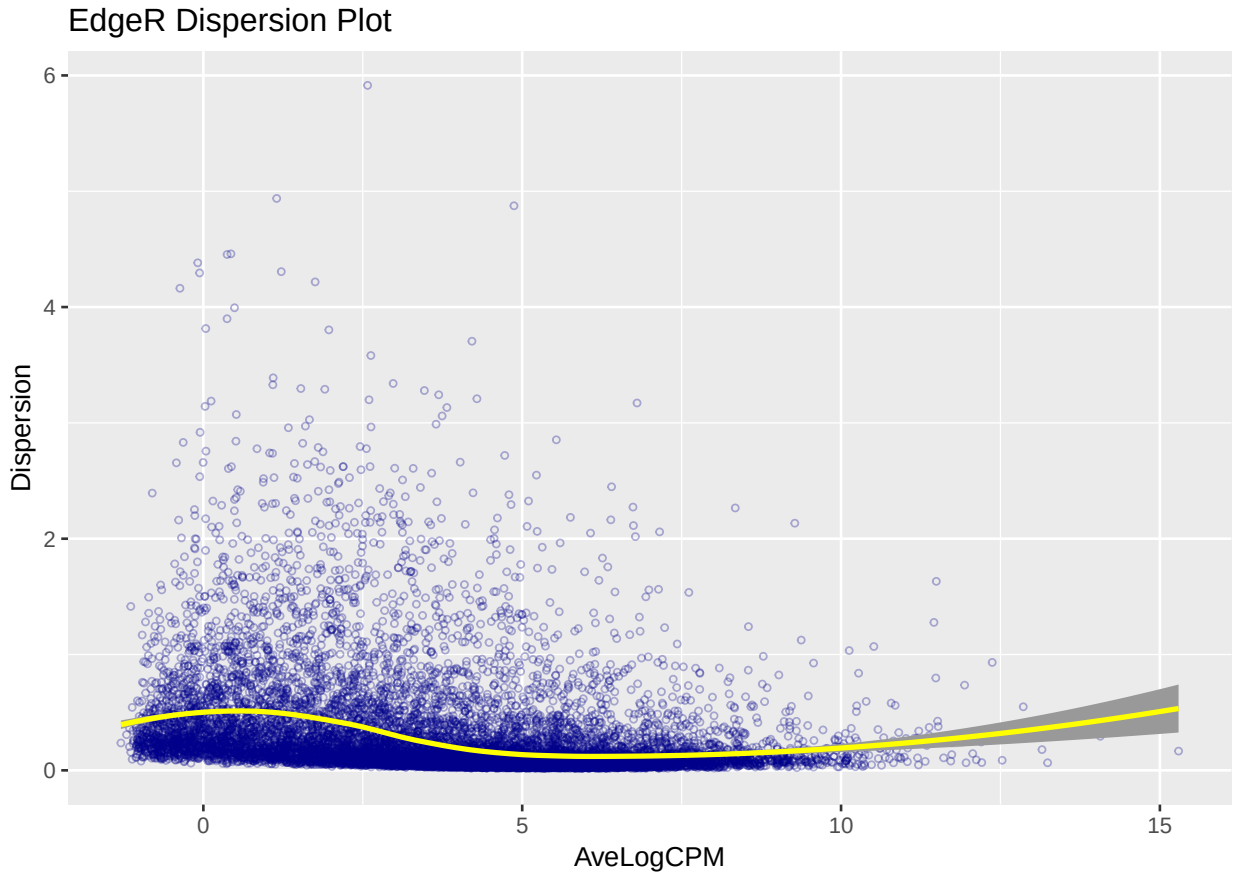


Note in this example each subject was tested twice. There are two points from the same individual that are annotated male and plot as a female and vice versa. Also, there are two points from one individual that plot in a position indicative of an XXY genotype.

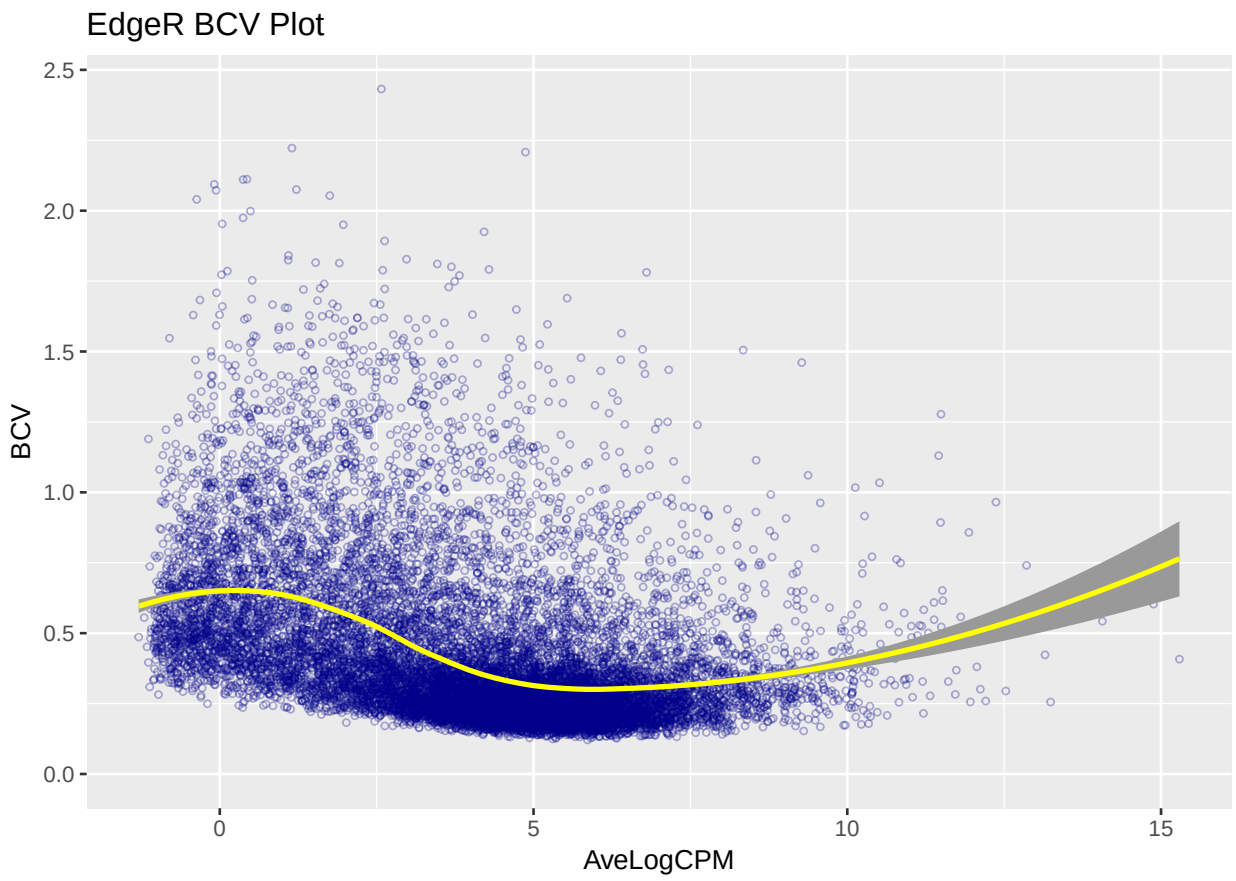
10 Dispersion Plot (plotDisp function)

Dispersion is a measure of variance that is plotted against intensity as a QC plot that illustrates heteroscedasticity.

```
# Note a DGEobj can have multiple design matrices or DGELists;  in this case
# only one of each is present
designMatrix <- getType(dgeObj, "designMatrix")[[1]]
dgelist <- getType(dgeObj, "DGEList")[[1]]
#dispersion plot
dispersionPlot <- plotDisp(dgelist, designMatrix, lineFit="loess", plotType="dispersion")
printAndSave(dispersionPlot, file.path(outputPath, "dispersionPlot.png"))
```




```
#BCV (biological coefficient of variation) plot
BCV_Plot <- plotDisp(dgelist, designMatrix, lineFit="loess", plotType="BCV")
printAndSave(BCV_Plot, file.path(outputPath, "BCV_Plot.png"))
```

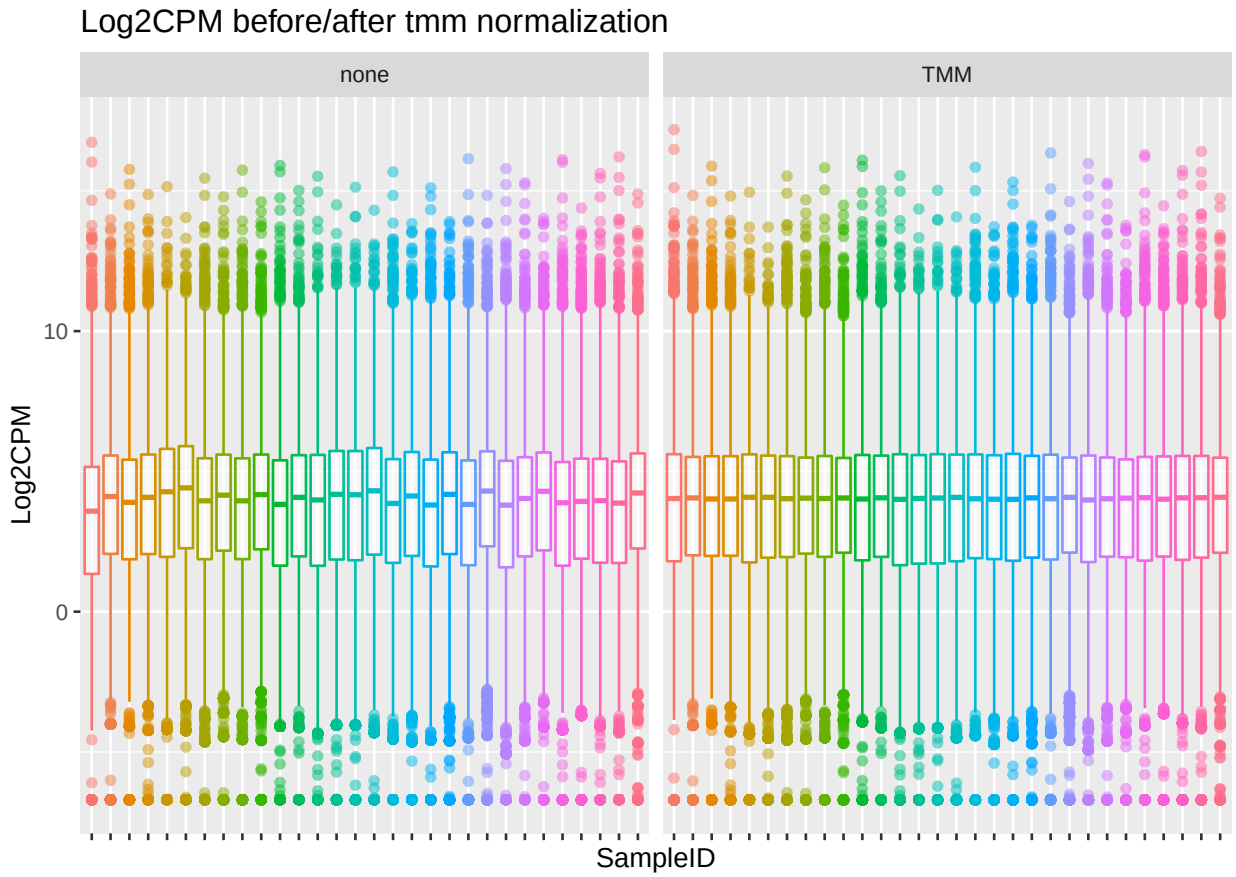

EdgeR BCV Plot
BCV
2.5
2.0
1.5
1.0
0.5
0.0
0
5
10
15
AveLogCPM

11 Plot Normalization (plotNorm function)

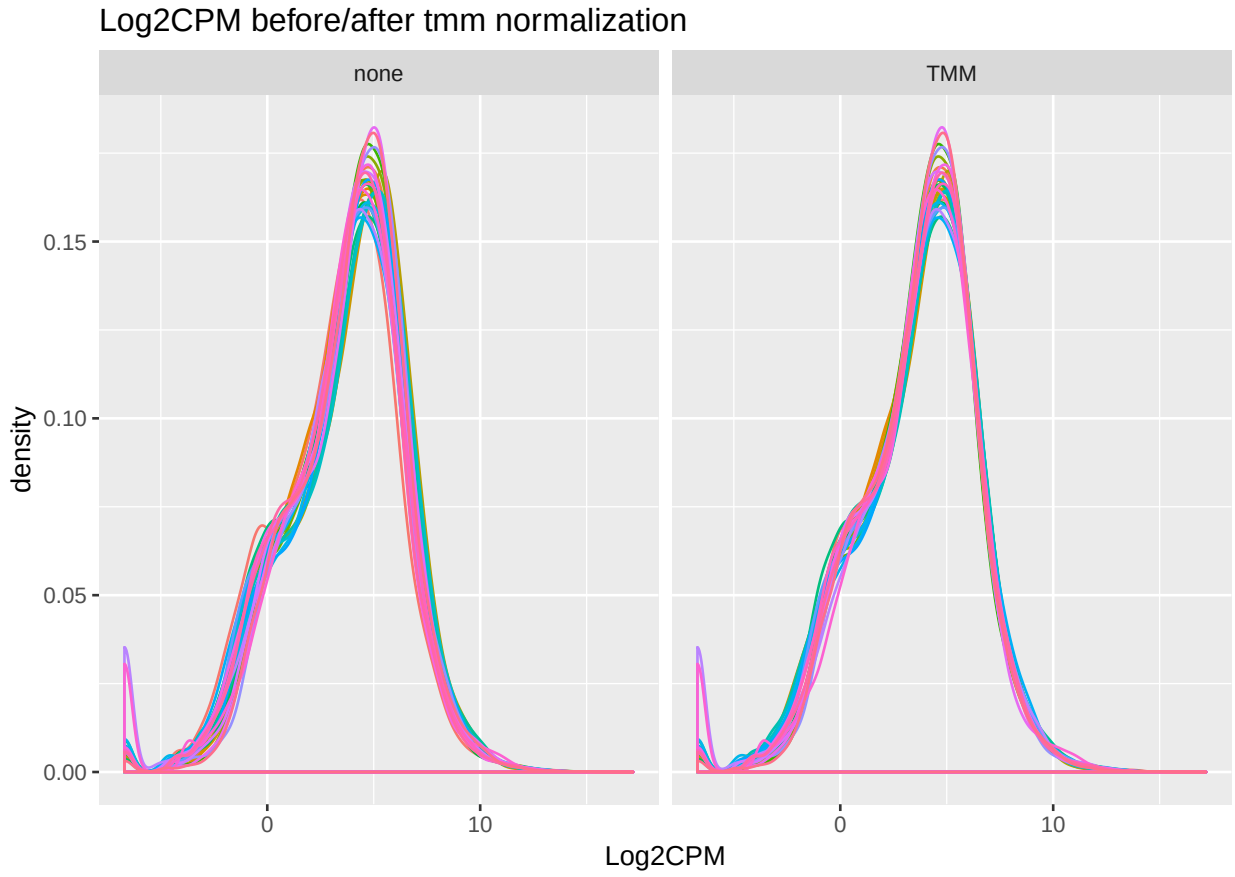
This function plots distributions before and after normalization as a two panel facetted plot. It uses edgeR::calcNormFactors and thus supports any normalization accepted by calcNormFactors.

Input can be a DGEobj or a raw counts matrix.
Output can be a box plot or density distribution.

```
boxNorm <- plotNorm(dgeObj, plotType="box", normalize="tmm")
printAndSave(boxNorm, file.path(outputPath, "boxNorm.png"))
```




```
densityNorm <- plotNorm(dgeObj, plotType="density", normalize="tmm")
printAndSave(densityNorm, file.path(outputPath, "densityNorm.png"))
```

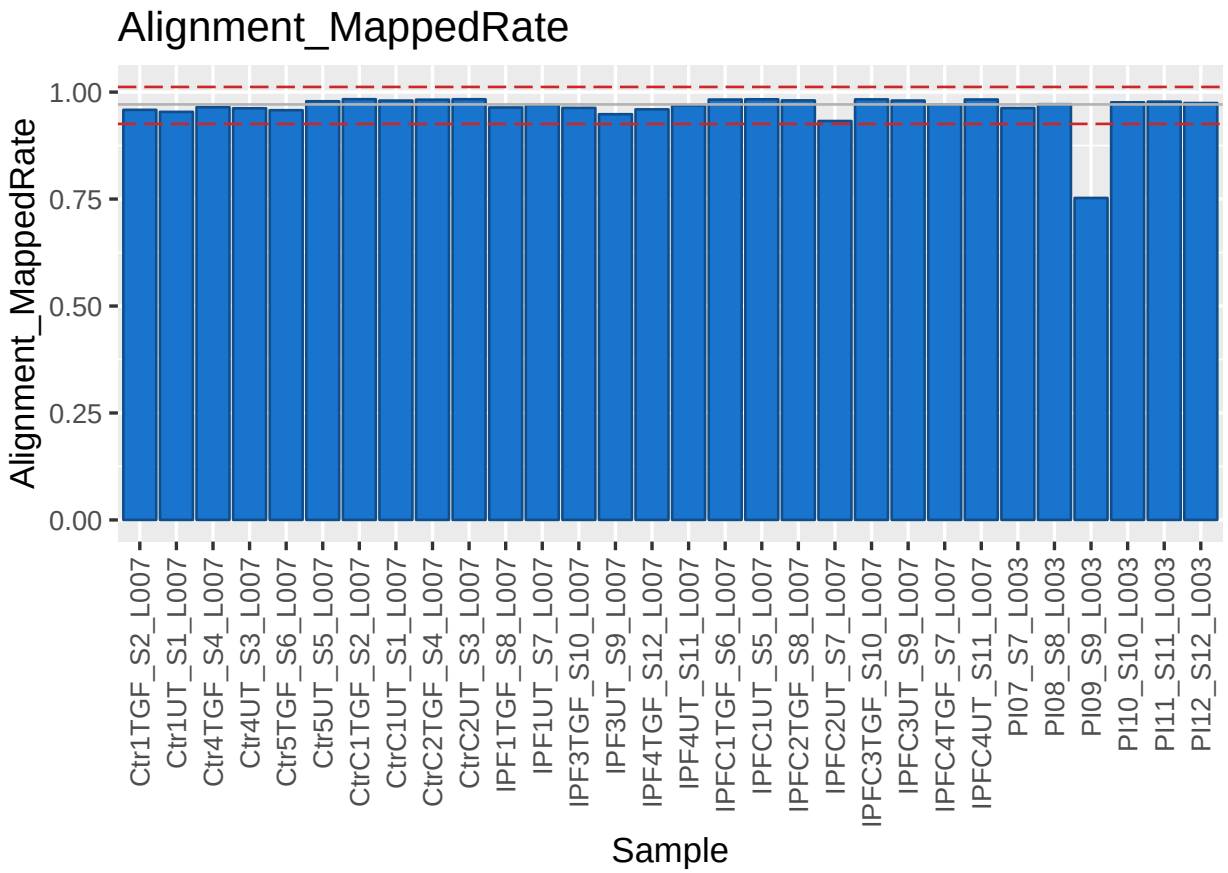

Log2CPM before/after tmm normalization
none
TMM
density
0.15
0.10
0.05
0.00
0
10
0
10
Log2CPM

12 Alignment QCplots (Function QCplots)

This function is intended to work on Omicsoft-style alignment QC data. QC parameters are named in the first column and subsequent columns hold data for each sample.

Since Omicsoft alignment QC includes hundreds of metrics, you typically want to limit the plots to a subset of the available metrics.

```
#Get some data from an Omicsoft project in S3
s3mount <- "Y:" #where you have mounted the S3 bucket "bmsrd-ngs-arrayserver"
s3path <- "/OmicsoftHome/output/P-20171107-0002/UCSD_Lung_Fibroblasts_P-20171107-0002_8Feb2018/Exporte
qcfilename <- "RNA-Seq.QCMetrics.Table.txt" #standard name for QC file in Omicosoft projects


qcdat <- readr::read_delim(file.path(s3mount, s3path, qcfilename), delim="\t")
#shorten the column names
colnames(qcdat) <- str_replace(colnames(qcdat), "_UCSD_Lung_Fibroblasts_P-20171107-0002_8Feb2018", ""

# qcdat <- qcdat[complete.cases(qcdat),]

#pick some Omicsoft Metrics from column 1 of the data frame
someFavMetrics <- c("Alignment_MappedRate", "Source_rRNA", "Profile_ExonRate",
                            "Profile_InterGene_FPK")

MyQCplots <- QCplots(qcdat, metricNames=someFavMetrics) #all defaults
# plots <- QCplots(qcdat, metricNames=someFavMetrics, plotType="bar", xAngle=90)
#draw the first plot
printAndSave(MyQCplots[[1]], file.path(outputPath, "QCplot.png"))
```



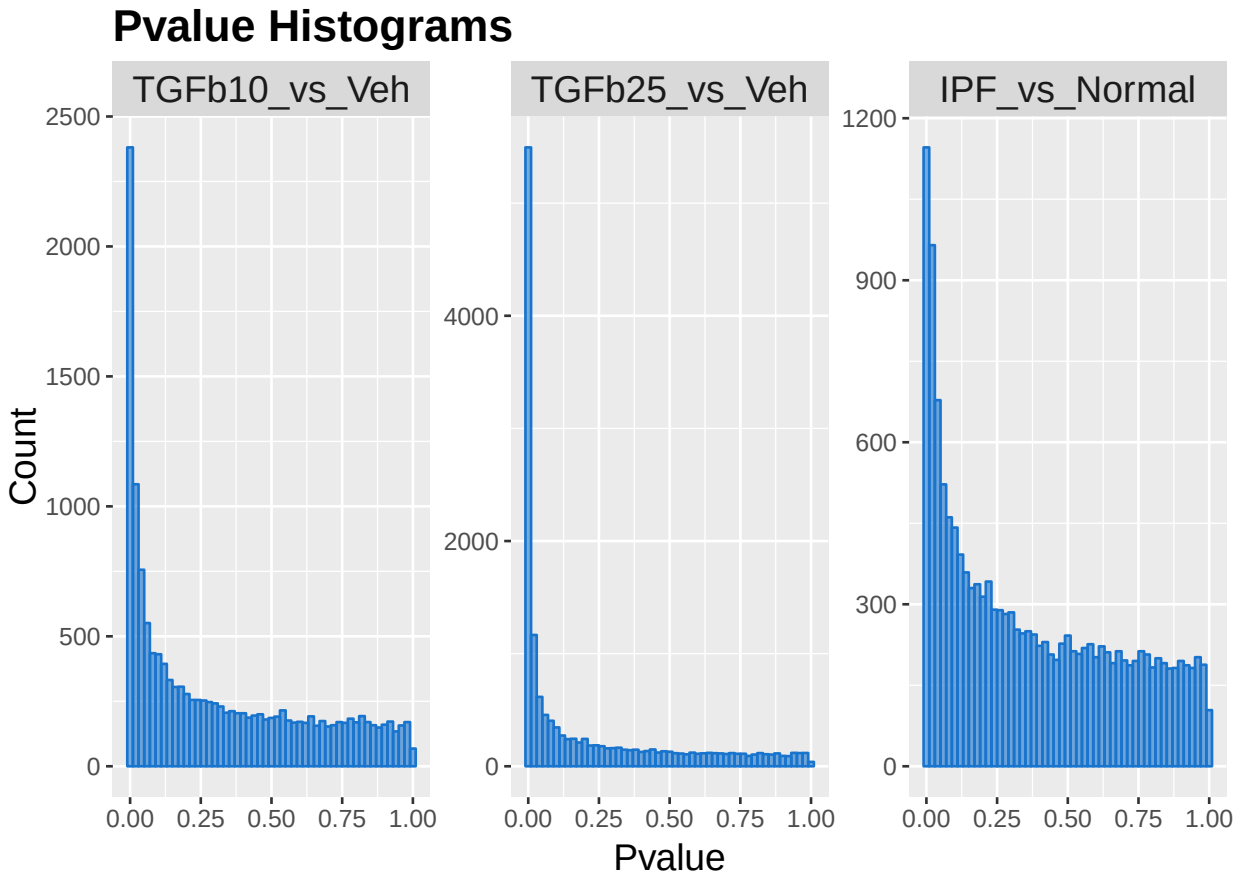
You need to download a QC data file from the S3 bucket bms-ngs-arrayserver or, preferrably, you can read data directly from S3 using software to mount an S3 bucket to a local location.

To map an S3 bucket to local storage:
For Mac: see s3fs
For PC: see Cloudberry Drive

13 Pvalue Histogram: Evaluate Quality of Fit

Code Block: Pvalue Histogram Facet Plot Example

```
topTableList <- getType(dgeObj, "topTable")

#Get all the pvalue columns from the contrasts
MyPval <- extractCol(topTableList, "P.Value")

pvalPlot <- plotPvalHist(MyPval)
printAndSave(pvalPlot, file.path(outputPath, "pvalHist.png"))
```



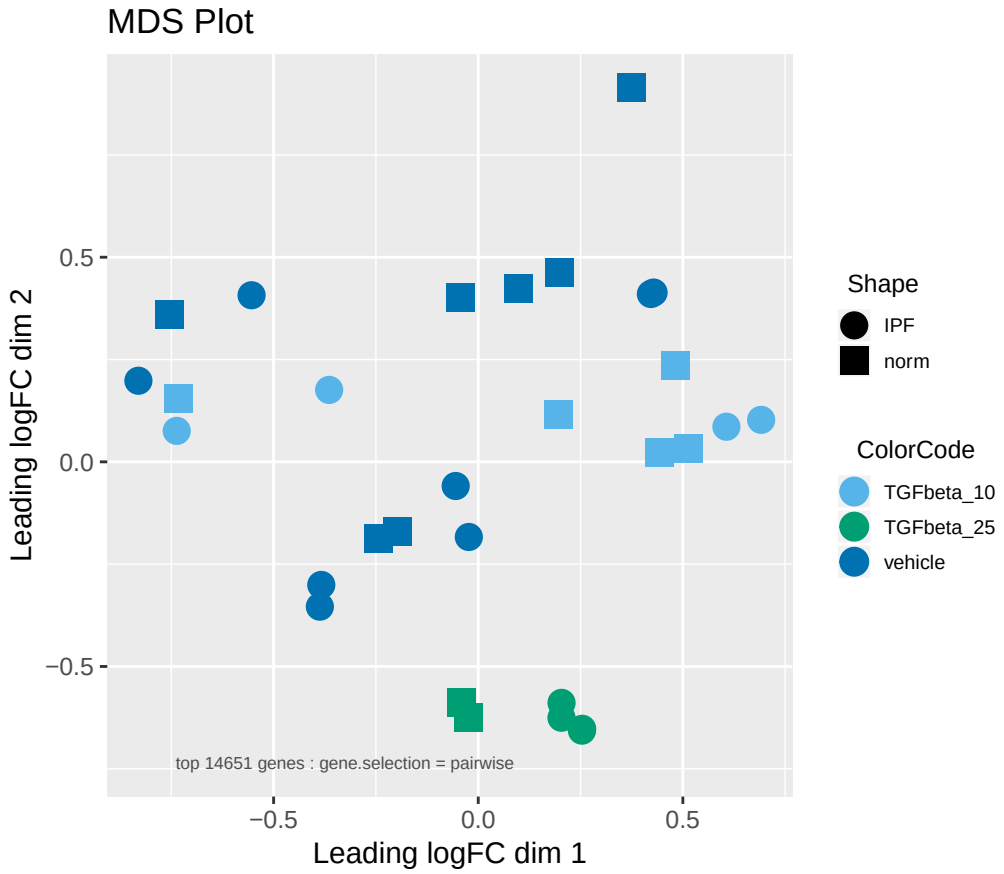
See ?plotPvalHist for optional arguments

14 ggplotMDS

Multidimentional scaling as implemented by limma::plotMDS is PCA-like but uses a distance metric instead of a correlation metric. As such, when you plot log2cpm values, the axes represent log2cpm units and you can interpret the distance between points accordingly.

ggplotMDS is a ggplot wrapper around limma::plotMDS

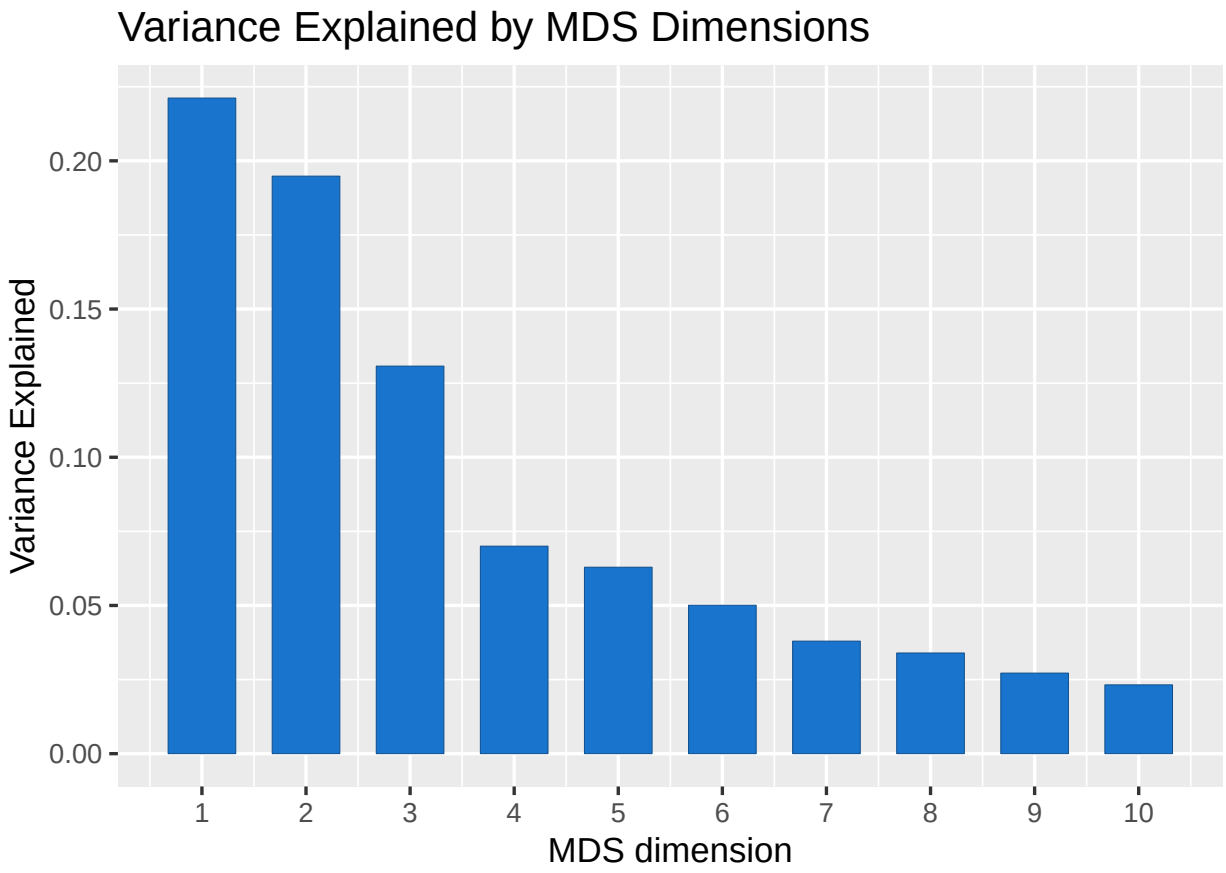
```
dgeList <- getItem(dgeObj, "DGEList")
result <- ggplotMDS(dgeList, colorBy = dgeObj$design$Treatment,
                    shapeBy = dgeObj$design$DiseaseStatus, labels = NULL)
printAndSave(result[[1]], file.path(outputPath, "MDSplot.png"))
```



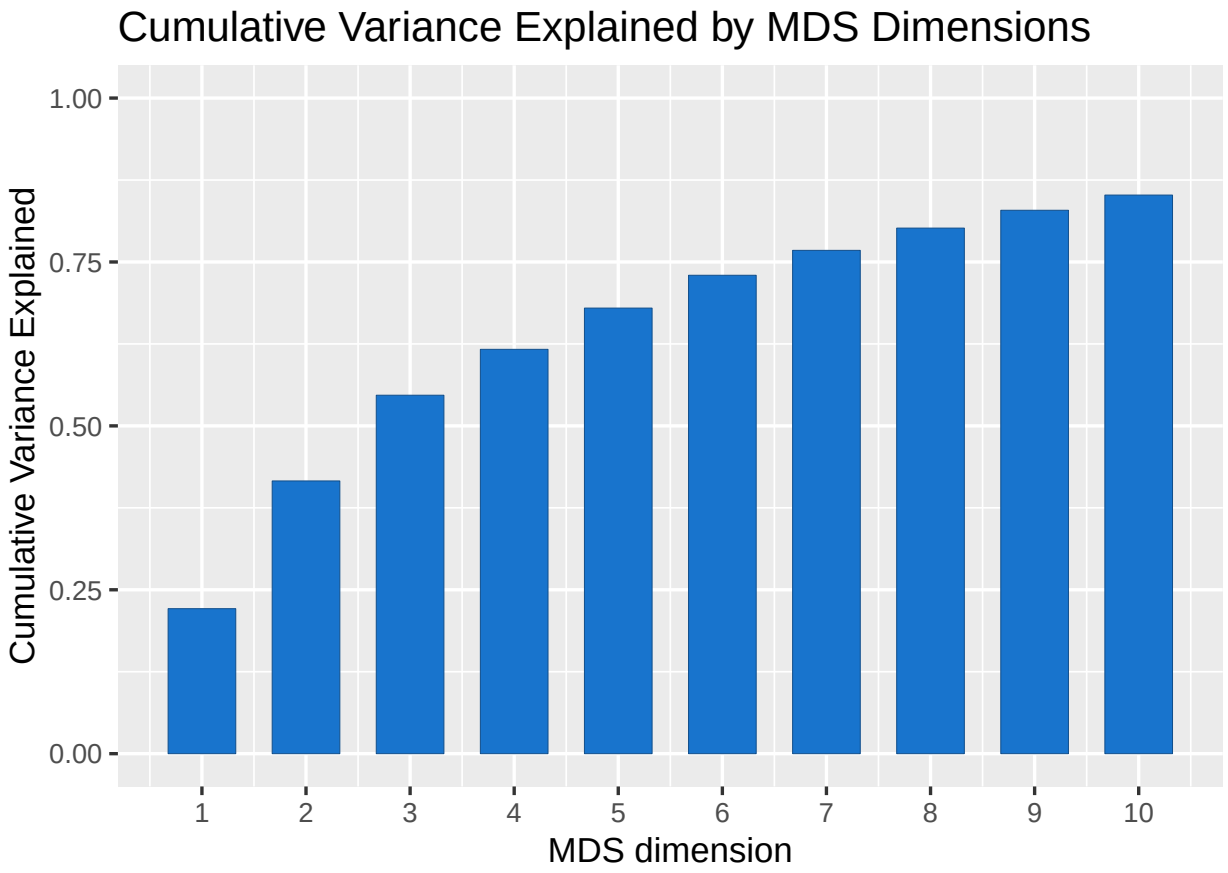
15 MDS_var_explained

Takes the mds object from the ggplotMDS result list (item 2) and returns a list with:
* a plot of the amount of variance explained by each component
* a cumulative plot of the variance explained
* the underlying data table

```
MDSvarResult <- MDS_var_explained(result[["mdsobj"]])

printAndSave(MDSvarResult$varexp, file.path(outputPath, "varExplained.png"))
```




```
printAndSave(MDSvarResult$cumvar, file.path(outputPath, "cumVarExplained.png"))
```


Cumulative Variance Explained by MDS Dimensions
Cumulative Variance Explained
1.00
0.75
0.50
0.25
0.00
1
2
3
4
5
6
7
8
9
10
MDS dimension

16 Session Info

Time required to process this report: *4.75285 mins*

R Session Info

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 14393)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] JRTutil_0.9.34      Biobase_2.42.0      BiocGenerics_0.28.0
##  [4] DGE.Tools2_0.9.63   DGEobj_0.9.32       magrittr_1.5
##  [7] forcats_0.4.0       stringr_1.4.0       dplyr_0.8.0.1
## [10] purrr_0.3.0         readr_1.3.1         tidyr_0.8.2
## [13] tibble_2.0.1        ggplot2_3.1.0       tidyverse_1.2.1
##
## loaded via a namespace (and not attached):
##   [1] readxl_1.3.0                backports_1.1.3
##   [3] NMF_0.21.0                  plyr_1.8.4
##   [5] igraph_1.2.4                assertive.files_0.0-2
##   [7] lazyeval_0.2.1              splines_3.5.2
##   [9] BiocParallel_1.16.6         zFPKM_1.4.1
##  [11] GenomeInfoDb_1.18.2         gridBase_0.4-7
##  [13] sva_3.30.1                  lpsymphony_1.10.0
##  [15] digest_0.6.18               foreach_1.4.4
##  [17] htmltools_0.3.6             checkmate_1.9.1
##  [19] memoise_1.1.0               assertive.numbers_0.0-2
##  [21] cluster_2.0.7-1             doParallel_1.0.14
##  [23] aws.signature_0.4.4         openxlsx_4.1.0
##  [25] limma_3.38.3                annotate_1.60.0
##  [27] modelr_0.1.4                matrixStats_0.54.0
##  [29] prettyunits_1.0.2           colorspace_1.4-0
##  [31] blob_1.1.1                  rvest_0.3.2
##  [33] ggrepel_0.8.0               assertive.strings_0.0-3
##  [35] haven_2.1.0                 xfun_0.5
##  [37] crayon_1.3.4                RCurl_1.95-4.11
##  [39] jsonlite_1.6                genefilter_1.64.0
##  [41] sendmailR_1.2-1             survival_2.43-3
##  [43] iterators_1.0.10            glue_1.3.0
##  [45] registry_0.5                gtable_0.2.0
##  [47] zlibbioc_1.28.0             XVector_0.22.0
```

```
##  [49] DelayedArray_0.8.0          scales_1.0.0
##  [51] pheatmap_1.0.12             rngtools_1.3.1
##  [53] DBI_1.0.0                   IHW_1.10.1
##  [55] edgeR_3.24.3                bibtex_0.4.2
##  [57] canvasXpress_1.22.9         Rcpp_1.0.0
##  [59] progress_1.2.0              xtable_1.8-3
##  [61] foreign_0.8-71              readat_1.8.0
##  [63] bit_1.1-14                  stats4_3.5.2
##  [65] rex_1.1.2                   htmlwidgets_1.3
##  [67] httr_1.4.0                  RColorBrewer_1.1-2
##  [69] pkgconfig_2.0.2             XML_3.98-1.17
##  [71] locfit_1.5-9.1              labeling_0.3
##  [73] tidyselect_0.2.5            rlang_0.3.1
##  [75] reshape2_1.4.3              AnnotationDbi_1.44.0
##  [77] munsell_0.5.0               cellranger_1.1.0
##  [79] tools_3.5.2                 pathological_0.1-2
##  [81] cli_1.0.1                   generics_0.0.2
##  [83] RSQLite_2.1.1               assertive.reflection_0.0-4
##  [85] broom_0.5.1                 aws.s3_0.3.12
##  [87] fdrtool_1.2.15              evaluate_0.13
##  [89] yaml_2.2.0                  processx_3.2.1
##  [91] RNASeqPower_1.22.1          knitr_1.21
##  [93] bit64_0.9-7                 zip_1.0.0
##  [95] assertive.sets_0.0-3        nlme_3.1-137
##  [97] slam_0.1-44                 ggiraph_0.6.0
##  [99] xml2_1.2.0                  biomaRt_2.38.0
## [101] compiler_3.5.2              rstudioapi_0.9.0
## [103] testthat_2.0.1              statmod_1.4.30
## [105] stringi_1.3.1               highr_0.7
## [107] ps_1.3.0                    gdtools_0.1.7
## [109] lattice_0.20-38             assertive.base_0.0-7
## [111] Matrix_1.2-15               psych_1.8.12
## [113] pillar_1.3.1                data.table_1.12.0
## [115] bitops_1.0-6                GenomicRanges_1.34.0
## [117] qvalue_2.14.1               assertive.types_0.0-3
## [119] R6_2.4.0                    assertive.properties_0.0-4
## [121] gridExtra_2.3               IRanges_2.16.0
## [123] codetools_0.2-16            assertthat_0.2.0
## [125] SummarizedExperiment_1.12.0 pkgmaker_0.27
## [127] withr_2.1.2                 mnormt_1.5-5
## [129] S4Vectors_0.20.1            GenomeInfoDbData_1.2.0
## [131] mgcv_1.8-26                 hms_0.4.2
## [133] grid_3.5.2                  rmarkdown_1.11
## [135] lubridate_1.7.4             base64enc_0.1-3
```