

# Choroid Contamination: TTR Groups

Kennedi Todd

8/26/2021

Load packages.

```
library(BiocParallel)
library(dplyr)

## 
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
## 
##     filter, lag

## The following objects are masked from 'package:base':
## 
##     intersect, setdiff, setequal, union

library(edgeR)

## Loading required package: limma

library(ggplot2)
library(ggrepel)
library(gplots)

## 
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
## 
##     lowess

library(grid)
library(gridExtra)

## 
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
## 
##     combine
```

```

library(knitr)
library(stringr)
library(variancePartition)

## Loading required package: scales

## Loading required package: Biobase

## Loading required package: BiocGenerics

## Warning: package 'BiocGenerics' was built under R version 4.0.5

## Loading required package: parallel

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
## 
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## The following object is masked from 'package:gridExtra':
## 
##     combine

## The following object is masked from 'package:limma':
## 
##     plotMA

## The following objects are masked from 'package:dplyr':
## 
##     combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
## 
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
## 
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

```

```

## Welcome to Bioconductor
##
##      Vignettes contain introductory material; view with
##      'browseVignettes()'. To cite Bioconductor, see
##      'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Attaching package: 'variancePartition'

## The following object is masked from 'package:limma':
## 
##     classifyTestsF

```

Read in files.

```

# read in counts
counts <- read.delim("Files/Brain_Hippocampus",
                      header = TRUE,
                      sep = "\t",
                      row.names = 1)

# read in metadata
metadata <- read.delim(
  "Files/GTEx_Analysis_v8_Annotations_SubjectPhenotypesDS.txt",
  header = TRUE,
  sep = "\t")

# read in gene annotation
gtf.file <- "Files/gencode.v26.GRCh38.genes.gtf"
gtf.gr <- rtracklayer::import(gtf.file)
gtf.df <- as.data.frame(gtf.gr)
gtf.df <- gtf.df[,c("type","gene_type","gene_name","gene_id","seqnames")]
gtf.genes <- gtf.df[gtf.df$type == "gene",]

```

Reformat metadata and counts.

```

# reformat metadata
colnames(metadata) <- c("subject_id", "sex", "age", "hardy_scale")

# reformat counts
rownames(counts) <- counts$name
counts <- counts[gtf.genes$gene_id,] # put counts in same order as gtf.genes
all.equal(counts$name, gtf.genes$gene_id) # check

## [1] TRUE

counts <- counts[,-c(1,2,200,201)] # remove mean, median, gene_id, gene_name

```

Rename counts columns to match metadata.

```

# change all hyphens to periods
metadata$subject_id <- gsub("-", ".", metadata$subject_id)
rownames(metadata) <- metadata$subject_id

```

```
# parse subject_id out of counts column names
old.names <- colnames(counts) # GTEX.13PDP.0011.R1a.SM.5PNX5
new.names <- str_match(old.names, "(GTEX\\.|[\\d\\w]+)\\.\\d+")[,2]
head(cbind(old.names, new.names)) # check
```

```
##          old.names             new.names
## [1,] "GTEX.11DXW.0011.R1a.SM.DNZZD" "GTEX.11DXW"
## [2,] "GTEX.11EI6.0011.R1a.SM.D093L" "GTEX.11EI6"
## [3,] "GTEX.11GS4.0011.R1a.SM.D0129" "GTEX.11GS4"
## [4,] "GTEX.11GSO.0011.R1b.SM.57WD3" "GTEX.11GSO"
## [5,] "GTEX.11GSP.0011.R1a.SM.9QEJ3" "GTEX.11GSP"
## [6,] "GTEX.11ONC.0011.R1a.SM.57WD4" "GTEX.11ONC"
```

```
colnames(counts) <- new.names

# Put metadata rows in the same order as counts columns
metadata <- metadata[colnames(counts),]

# check
all.equal(rownames(metadata), colnames(counts))
```

```
## [1] TRUE
```

Create dge object

```
dge <- DGEList(counts,
                 samples = metadata,
                 genes = gtf.genes)
```

Look to see if any NA values or replicated gene names.

```
table(is.na(gtf.genes$gene_name))
```

```
##
## FALSE
## 56200
```

```
table(duplicated(gtf.genes$gene_name))
```

```
##
## FALSE  TRUE
## 54592 1608
```

There are no NA values. However, 1,608 gene names are repetitive.

View the repetitive gene names.

```
table(gtf.genes$gene_name[duplicated(gtf.genes$gene_name)])
```

##					
##	5_8S_rRNA	5S_rRNA	7SK	ACA59	ACA64
##	4	15	5	1	5
##	ACTR3BP2	AKAP17A	ALG1L9P	AMD1P2	ASMT
##	1	1	1	1	1
##	ASMTL	ASMTL-AS1	C2orf61	CD99	CD99P1
##	1	1	1	1	1
## Clostridiales-1		CRLF2	CSF2RA	CTSLP2	CYB561D2
##	2	1	1	1	1
##	DDX11L16	DHRSX	DHRSX-IT1	DNAJC9-AS1	DPH3P2
##	1	1	1	1	1
##	ELFN2	ELOCP24	FABP5P13	FAS-AS1	GOLGA8M
##	1	1	1	1	1
##	GTPBP6	IL3RA	IL9R	KRT18P53	LINC00102
##	1	1	1	1	1
##	LINC00106	LINC00484	LINC00685	LINC00901	LINC01115
##	1	1	1	1	1
##	LINC01238	LINC01297	LINC01347	LINC01422	LINC01481
##	1	1	1	1	1
##	LINC01598	LLOYNC03-29C1.1	LYNX1	MAL2	Metazoa_SRP
##	1	1	1	1	166
##	MIR3179-3	MIR3180-4	MIR3690	MIR6089	NBPF13P
##	1	1	1	1	1
##	OR7E47P	P2RY8	PLCXD1	PMS2P6	PPP2R3B
##	1	1	1	1	1
##	pRNA	PROX1-AS1	RAET1E-AS1	RGS5	RNA5-8S5
##	8	1	1	1	3
##	RNA5SP498	RP11-309M23.1	RP13-297E16.4	RP13-297E16.5	RP13-465B17.4
##	1	1	1	1	1
##	RP13-465B17.5	RPL14P5	RPS23P5	SCARNA11	SCARNA15
##	1	1	1	3	1
##	SCARNA16	SCARNA17	SCARNA18	SCARNA20	SCARNA21
##	4	3	1	6	3
##	SCARNA24	SCARNA6	SHOX	SLC25A6	snoMBII-202
##	1	1	1	1	1
## snoMe28S-Am2634		SNORA1	SNORA11	SNORA12	SNORA15
##	2	4	6	2	3
##	SNORA17	SNORA18	SNORA19	SNORA2	SNORA20
##	1	7	3	2	4
##	SNORA21	SNORA22	SNORA24	SNORA25	SNORA26
##	1	3	1	18	10
##	SNORA27	SNORA3	SNORA30	SNORA31	SNORA32
##	4	4	2	25	2
##	SNORA33	SNORA35	SNORA36	SNORA38	SNORA4
##	3	2	1	1	4
##	SNORA40	SNORA41	SNORA42	SNORA43	SNORA44
##	20	1	3	4	1
##	SNORA46	SNORA48	SNORA50	SNORA51	SNORA57
##	1	12	1	12	2
##	SNORA58	SNORA62	SNORA63	SNORA64	SNORA67
##	2	6	10	5	6
##	SNORA68	SNORA69	SNORA7	SNORA70	SNORA71
##	5	1	4	29	2
##	SNORA72	SNORA73	SNORA74	SNORA75	SNORA76

##	6	3	6	7	1
##	SNORA77	SNORA79	SNORA8	SNORA81	SNORA9
##	2	1	4	3	3
##	SNORD11	SNORD111	SNORD112	SNORD113	SNORD115
##	1	1	2	4	1
##	SNORD116	SNORD18	SNORD19	SNORD19B	SNORD2
##	2	1	1	1	1
##	SNORD22	SNORD23	SNORD27	SNORD28	SNORD30
##	1	1	1	1	1
##	SNORD33	SNORD37	SNORD38	SNORD39	SNORD41
##	1	2	3	3	1
##	SNORD42	SNORD45	SNORD46	SNORD5	SNORD51
##	1	3	2	2	1
##	SNORD56	SNORD59	SNORD60	SNORD63	SNORD65
##	6	1	2	3	3
##	SNORD66	SNORD67	SNORD70	SNORD74	SNORD75
##	1	1	1	6	1
##	SNORD77	SNORD78	SNORD81	snoU109	snoU13
##	4	1	2	5	31
##	snoU2_19	snoZ6	SPATA13	SPRY3	TRPC6P
##	2	3	1	1	1
##	U1	U2	U3	U4	U6
##	12	18	49	6	32
##	U7	U8	uc_338	VAMP7	Vault
##	5	21	31	1	1
##	WASH6P	WASIR1	Y_RNA	ZBED1	
##	1	1	737	1	

View the gene\_type for the repetitive genes. This will help determine if we should remove repetitive gene names.

```
table(gtf$genes$gene_type[duplicated(gtf$genes$gene_name)])
```

##	antisense	lincRNA
##	12	15
##	miRNA	misc_RNA
##	3	949
##	processed_pseudogene	protein_coding
##	9	21
##	rRNA	scaRNA
##	23	22
##	sense_intronic	snoRNA
##	2	468
##	snRNA	sRNA
##	73	2
##	transcribed_unprocessed_pseudogene	unprocessed_pseudogene
##	4	5

Remove duplicated gene names which are mainly non-coding RNAs.

```
removeDuplicated <- !duplicated(gtf.genes$gene_name) # true when not duplicated
dge <- dge[removeDuplicated,,keep.lib.sizes=FALSE]
dim(dge)
```

```
## [1] 54592 197
```

Remove mitochondrial genes. They are super abundant and can skew data.

```
removeMT <- dge$genes$seqnames != "chrM" # true when not chrM
dge <- dge[removeMT,,keep.lib.sizes = FALSE]
dim(dge)
```

```
## [1] 54555 197
```

Make gene\_name the row name.

```
rownames(dge$counts) <- dge$genes$gene_name
```

Change sex column from numbers to letters.

```
# male = 1, female = 2
sex_nums <- as.numeric(dge$samples$sex)
sex_letters <- gsub(1, "M", sex_nums)
sex_letters <- gsub(2, "F", sex_letters)
head(cbind(sex_nums, sex_letters)) # check
```

```
##      sex_nums sex_letters
## [1,] "1"       "M"
## [2,] "1"       "M"
## [3,] "1"       "M"
## [4,] "1"       "M"
## [5,] "2"       "F"
## [6,] "1"       "M"
```

```
dge$samples$sex <- sex_letters
```

---

### TTR TPM HISTOGRAM

---

Data is already in TPM (transcripts per million). Create histogram of TPM TTR values.

```
ttr.counts <- as.data.frame(dge$counts["TTR",])
colnames(ttr.counts) <- "ttr.counts"

log2.ttr.counts <- log2(ttr.counts + 0.01)
colnames(log2.ttr.counts) <- "log2.ttr.counts"
```

```
h1 <- ggplot(ttr.counts, aes(x = ttr.counts)) +
  geom_histogram(bins = 100, fill = "gray", color = "black") +
  labs(title = "A.", x=NULL, y=NULL) +
  xlab("TTR TPM") + ylab("# of Samples") +
```

```

geom_vline(xintercept = 20, col = "red") +
theme_bw()

h2 <- ggplot(ttr.counts, aes(x = ttr.counts)) +
geom_histogram(bins = 100, fill = "gray", color = "black") +
labs(title = "B.", x=NULL, y=NULL) +
xlab("TTR TPM") + ylab("# of Samples") +
xlim(0, 500) + geom_vline(xintercept = 20, col = "red") +
theme_bw()

h3 <- ggplot(log2.ttr.counts, aes(x = log2.ttr.counts)) +
geom_histogram(bins = 100, fill = "gray", color = "black") +
labs(title = "GTEX\nHippocampus", x=NULL, y=NULL) +
xlab("log2(TTR TPM)") + ylab("# of Samples") +
theme_bw() +
geom_vline(xintercept = log2(20) + 1, col = "blue") +
geom_vline(xintercept = log2(20) - 1, col = "blue")

```

Arrange graphs in grid.

```

plots1 <- list(h1,h2,h3)

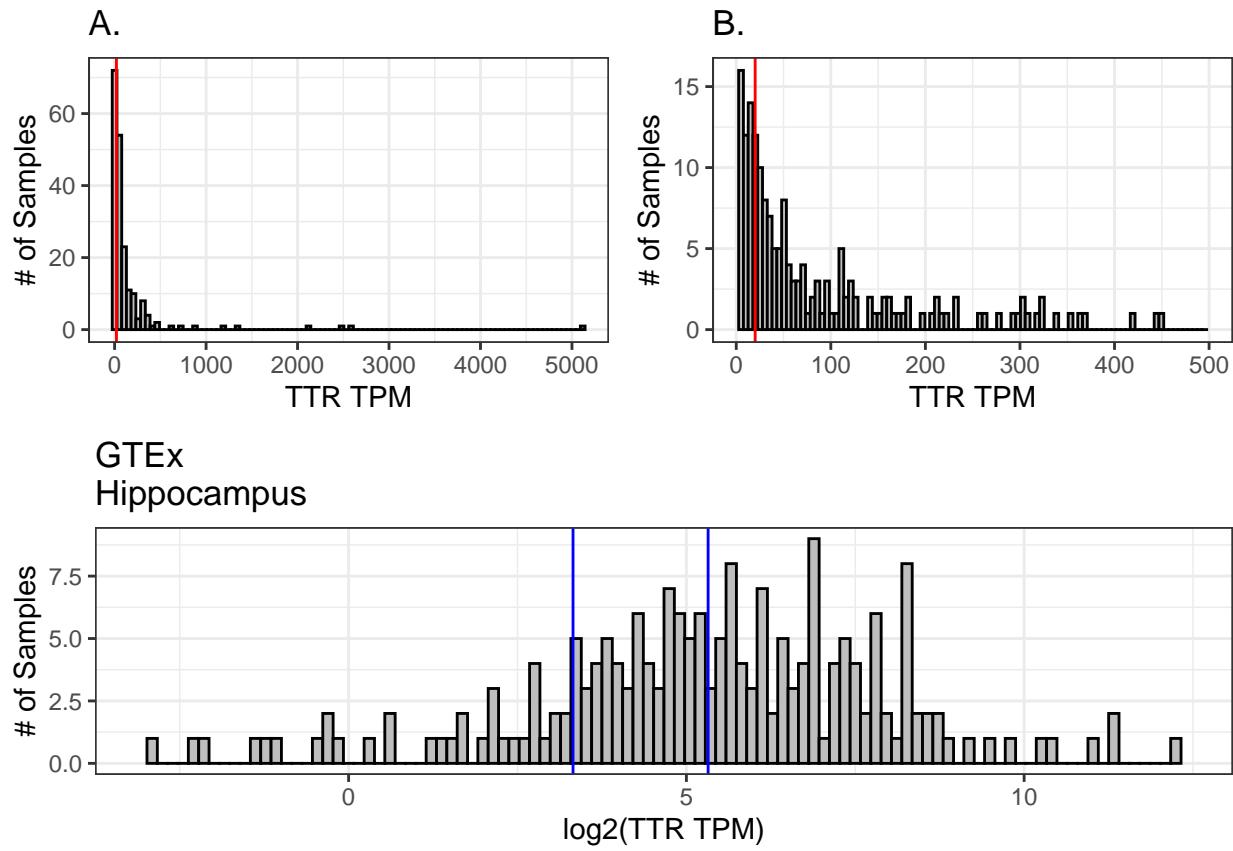
layout1 <- rbind(c(1,2),c(3))

grid1 <- grid.arrange(grobs = plots1, layout_matrix = layout1)

## Warning: Removed 9 rows containing non-finite values (stat_bin).

## Warning: Removed 2 rows containing missing values (geom_bar).

```




---

ASSIGN DIRTY/CLEAN/NEITHER

---

Lets looks at groups +/- 1 of the cutoff

```
cutoff <- log2(20)
clean_threshold <- cutoff - 1 # values <= this are clean
clean_threshold
```

```
## [1] 3.321928
```

```
cutoff
```

```
## [1] 4.321928
```

```
dirty_threshold <- cutoff + 1 # values >= this are dirty
dirty_threshold
```

```
## [1] 5.321928
```

Assign clean or dirty

```
clean_or_dirty <- vector()
vector.log2.ttr.counts <- as.vector(log2.ttr.counts$log2.ttr.counts)

for (i in 1:length(vector.log2.ttr.counts)){
```

```

if (vector.log2.ttr.counts[i] <= clean_threshold){
  clean_or_dirty <- c(clean_or_dirty, "clean")
}
else if (vector.log2.ttr.counts[i] >= dirty_threshold){
  clean_or_dirty <- c(clean_or_dirty, "dirty")
}
else {
  clean_or_dirty <- c(clean_or_dirty, "neither")
}
}
table(clean_or_dirty)

```

```

## clean_or_dirty
##   clean    dirty neither
##     34      100     63

```

```

dge$samples$group <- factor(clean_or_dirty)
head(dge$samples)

```

	group	lib.size	norm.factors	subject_id	sex	age	hardy_scale
## GTEX.11DXW	neither	295065.7		1 GTEX.11DXW	M	40-49	2
## GTEX.11EI6	dirty	255728.4		1 GTEX.11EI6	M	60-69	4
## GTEX.11GS4	dirty	286636.9		1 GTEX.11GS4	M	60-69	2
## GTEX.11GSO	neither	290896.5		1 GTEX.11GSO	M	60-69	2
## GTEX.11GSP	clean	288834.7		1 GTEX.11GSP	F	60-69	2
## GTEX.11ONC	clean	214826.5		1 GTEX.11ONC	M	60-69	2

```

h4 <- h3 +
  annotate("rect",
    xmin = -Inf,
    xmax = clean_threshold,
    ymin = 0,
    ymax=Inf,
    alpha=0.2,
    fill="deepskyblue") +
  annotate("rect",
    xmin = dirty_threshold,
    xmax = Inf,
    ymin = 0,
    ymax=Inf,
    alpha=0.2,
    fill="chocolate4")

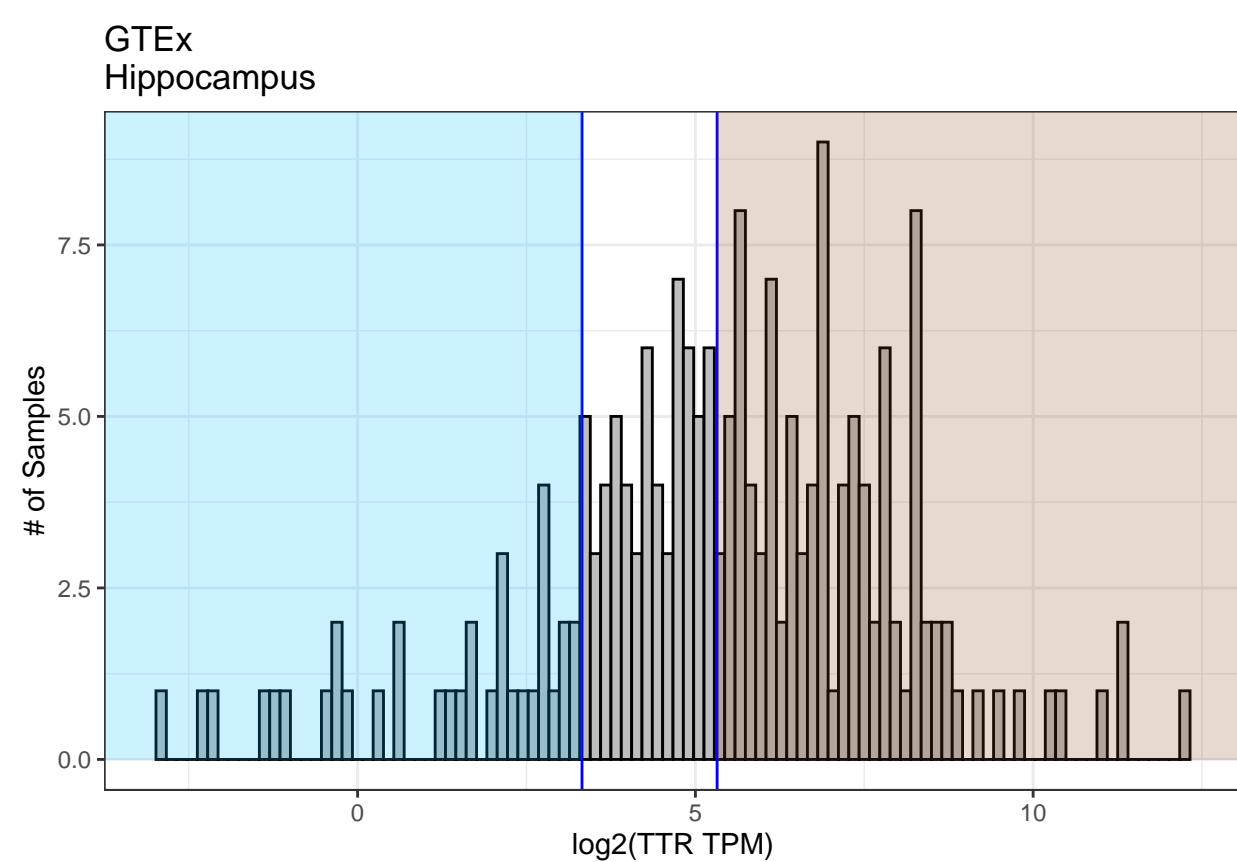
```

```

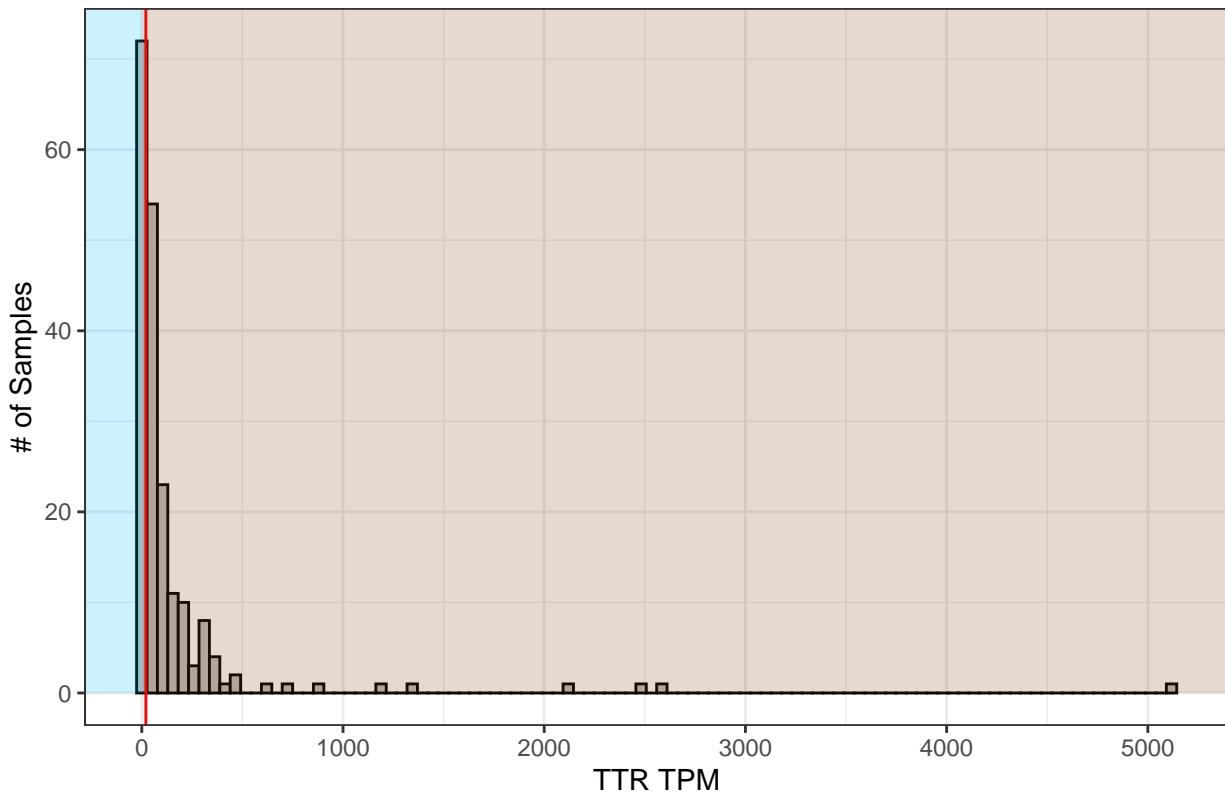
h5 <- h1 +
  annotate("rect",
    xmin = -Inf,
    xmax = clean_threshold,
    ymin = 0,
    ymax=Inf,
    alpha=0.2,
    fill="deepskyblue") +
  annotate("rect",
    xmin = dirty_threshold,
    xmax = Inf,
    ymin = 0,
    ymax=Inf,
    alpha=0.2,
    fill="chocolate4")

```

```
xmax = Inf,  
ymin = 0,  
ymax=Inf,  
alpha=0.2,  
fill="chocolate4")  
h4
```



A.



```
setwd("/Users/m214960/Documents/GTEx")
pdf(
  "Figures/GTEx_TTR_histogram.pdf",
  width = 4, height = 4)
h4
```

————— FILTER LOWLY EXPRESSED GENES —————

TPM to CPM

```
cpm <- cpm(dge)
```

Filter

```
# produces a logical matrix
thresh <- cpm > 2 # cpm greater than 2

# keep genes that have at least 3 TRUES in each row of thresh
keep <- rowSums(thresh) >= 3
summary(keep)
```

```
##      Mode    FALSE     TRUE
## logical   30225   24330
```

```
# redefine object, filter by logical
dge.filtered <- dge[keep,, keep.lib.sizes = FALSE]

dim(dge)
```

```
## [1] 54555 197
```

```
dim(dge.filtered)
```

```
## [1] 24330 197
```

Went from 54,592 genes to 18,845 genes.

---

#### TRIMMED MEAN OF M-VALUES

---

Now, we want to normalize gene expression distributions. We do this by normalizing library size differences in each sample. The method Trimmed Mean of M-values (TMM) is used with the calcNormFactors in the edgeR package. The normalization factors calculated here are used as a scaling factor for the library size.

Trimmed mean of M values (TMM) normalization estimates sequencing depth after excluding genes for which the ratio of counts between a pair of experiments is too extreme or for which the average expression is too extreme. The edgeR software implements a TMM normalization (Biostar Handbook 2020).

```
# creates another large DGEList
dge.filtered.norm <- calcNormFactors(dge.filtered, method = "TMM")

# norm factor distribution
summary(dge.filtered.norm$samples$norm.factors)
```

```
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
##  0.6441  0.9488  1.0124  1.0047  1.0652  1.2132
```

Looking to be around ~1.

---

-Variance Partition - See what options we have to choose from.

```
colnames(dge.filtered.norm$samples)
```

```
## [1] "group"          "lib.size"        "norm.factors"    "subject_id"     "sex"
## [6] "age"            "hardy_scale"
```

Specify variables to consider

```
register(SnowParam(4)) # work in parallel

# geneExpr: matrix of gene expression values
# info: information/metadata about each sample
geneExpr <- as.matrix(dge.filtered.norm$counts)
info <- as.data.frame(dge.filtered.norm$samples)
info$hardy_scale <- as.factor(info$hardy_scale)

# Age is USUALLY continuous so model it as a fixed effect "age"
```

```

# However, this metadata categorizes it into bins so it's categorical
# Sex is categorical, so model them as random effects "(1|sex)"
# Note the syntax
form <- ~ (1|group) +
  (1|age) +
  (1|sex) +
  (1|hardy_scale)

varPart <- fitExtractVarPartModel(geneExpr, form, info)

## Dividing work into 100 chunks...

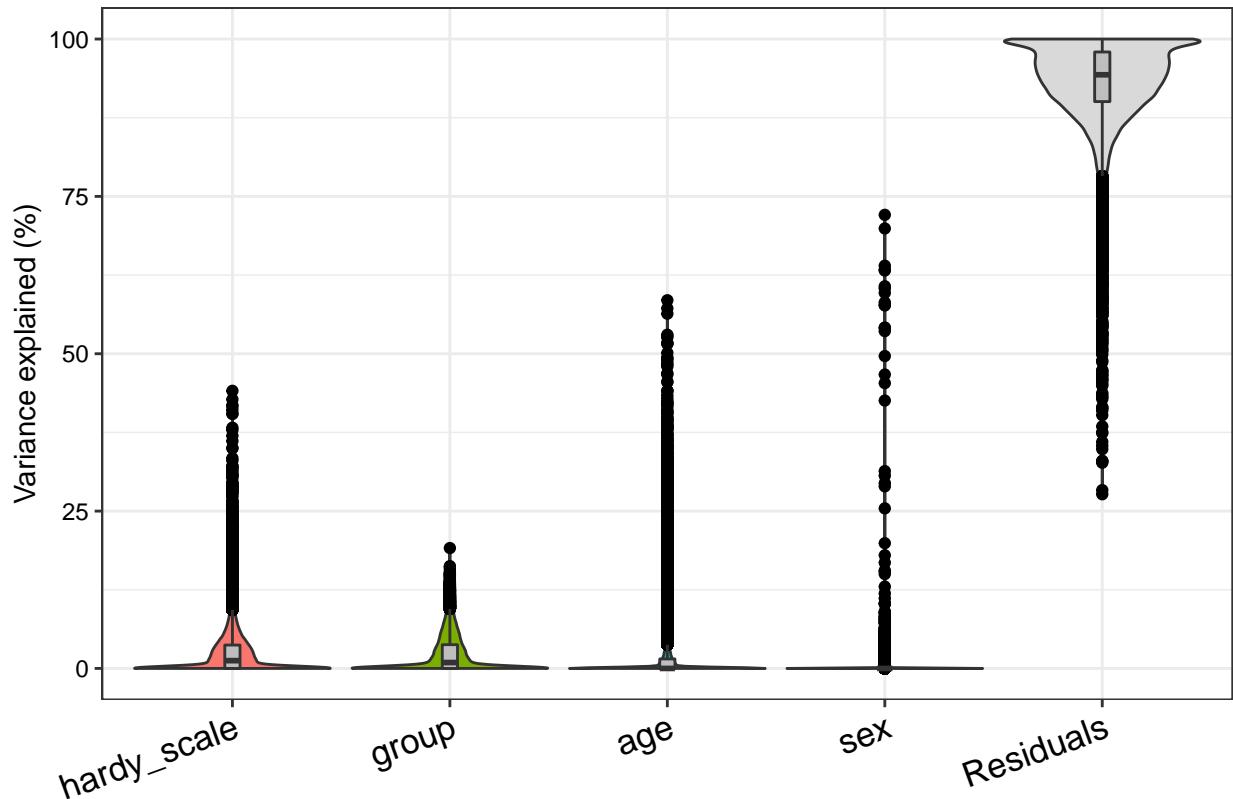
##
## Total:262 s

vp <- sortCols(varPart)

```

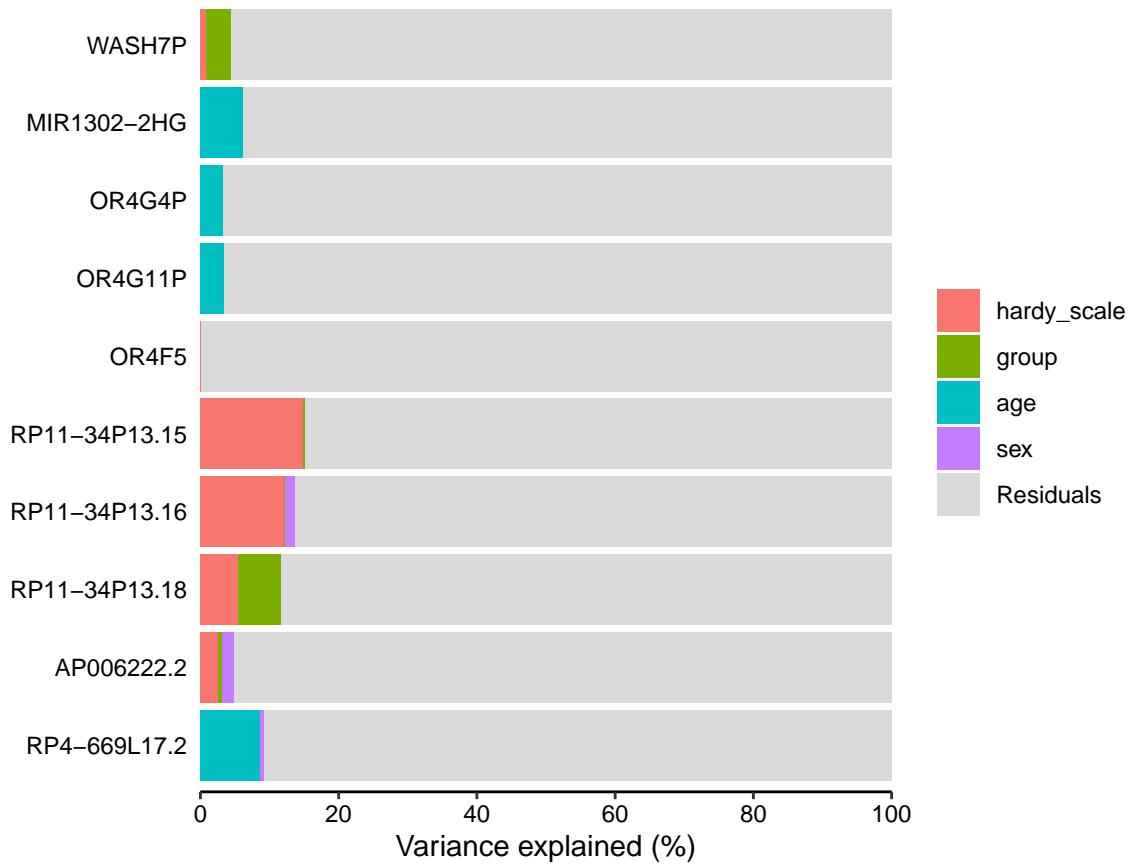
Violin plots

```
plotVarPart(vp)
```



Percent bars

```
plotPercentBars(vp[1:10,])
```



```
# sort genes based on variance explained by group
varPart.df <- as.data.frame(varPart)
order.varPart.df <- varPart.df[order(varPart.df$group, decreasing = TRUE),]
head(order.varPart.df["group"])
```

```
##           group
## HPD      0.1912756
## CDK6     0.1625918
## C22orf46 0.1613220
## GLIPR1L2 0.1581813
## JADE3    0.1536846
## RNF152   0.1519967
```

#### DESIGN MATRIX

```
sex <- dge.filtered.norm$samples$sex
age <- dge.filtered.norm$samples$age
```

Create a design matrix and specify the samples we want to compare to each other.

```
# interaction computes a factor which represents the interaction of the given factors
group <- interaction(dge.filtered$samples$group)

design <- model.matrix(~ 0 +
                        group +
```

```

    sex + # remove if subset by sex
    age)
colnames(design) <- make.names(colnames(design))
colnames(design)[1:2] <- c("clean", "dirty")

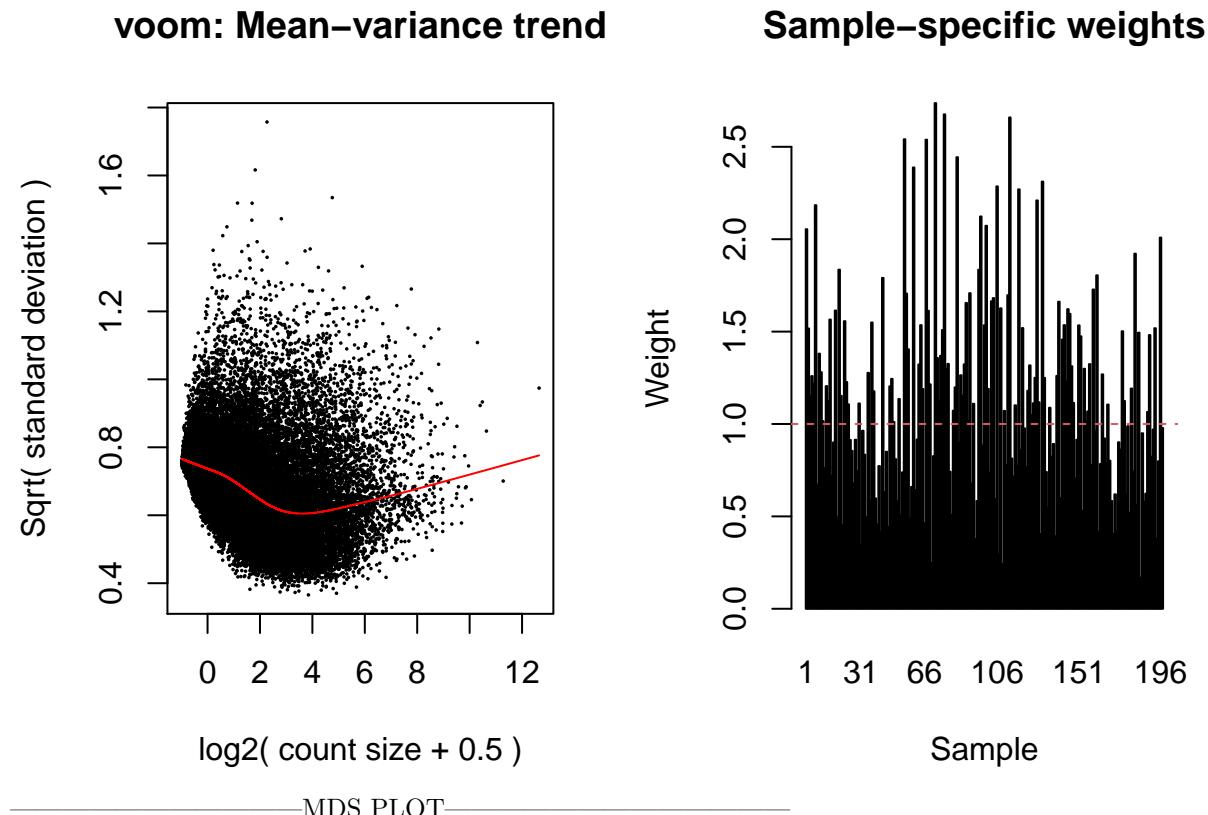
```

Run voom with quality weights. This combines observational-level weights with sample-specific quality weights in an experiment. Normalizes expression intensities so that the log-ratios have similar distributions across a set of samples. To quantile normalize, add normalize.method = “quantile”.

Quality weights in RNA-seq data improves data quality when samples with high variance are present. This variation is modeled by taking into account global intensity-dependent trends using voom.

Sample-specific variability is taken into account with a log-linear model that shares parameters between genes. <https://doi.org/10.1093/nar/gkv412>

```
v <- voomWithQualityWeights(dge.filtered.norm, design, plot = TRUE)
```



```

# blue is clean, brown is dirty
group_colors <- c("deepskyblue", "chocolate4", "gray")[v$targets$group]
# F or M for points
point_shapes <- v$targets$sex
# Check everything matches
head(data.frame(v$targets$group, group_colors, v$targets$sex, point_shapes))

```

```

##   v.targets.group group_colors v.targets.sex point_shapes
## 1      neither      gray          M           M
## 2       dirty     chocolate4         M           M

```

```

## 3      dirty   chocolate4      M      M
## 4    neither     gray      M      M
## 5      clean  deepskyblue      F      F
## 6      clean  deepskyblue      M      M

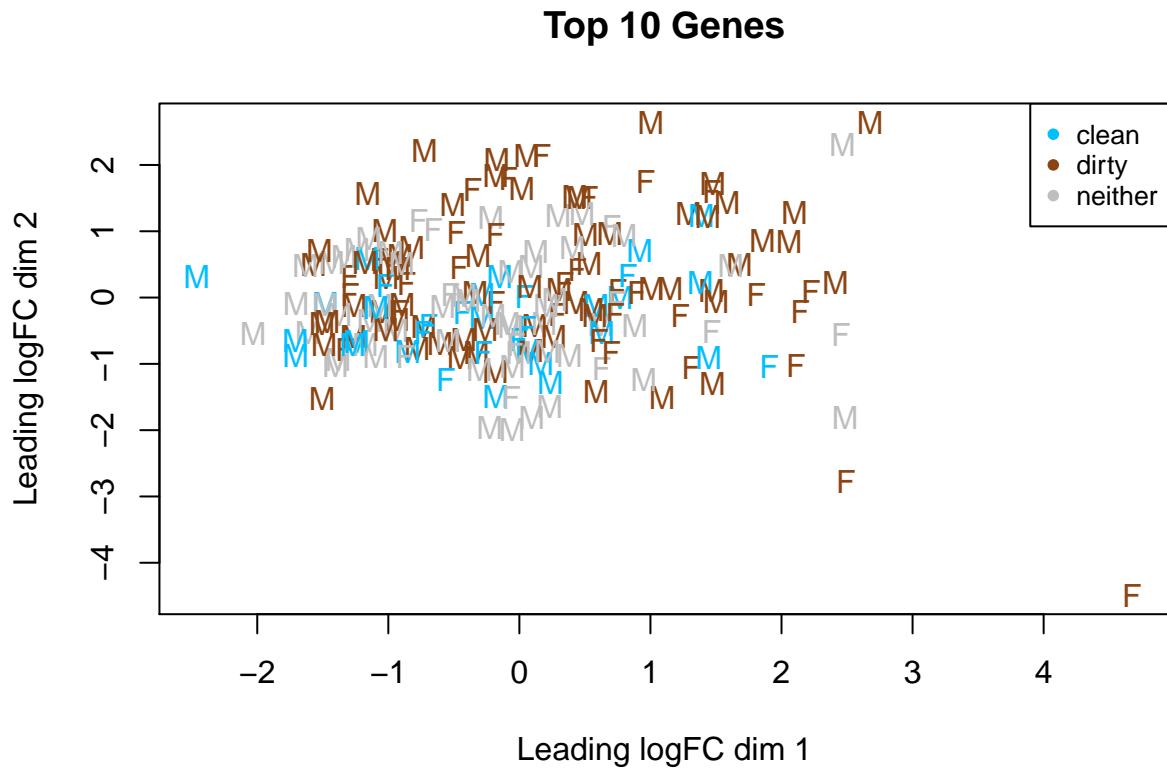
#pdf("Figures/GTEx_TTR_MDS_voom.pdf", width = 4, height = 4)

mds <- plotMDS(
  v, # our data object
  top = 10, # only looking at top 10 genes
  pch = point_shapes,
  cex = 1, # point size
  dim.plot = c(1,2), # specifying principal components to be plotted
  plot = TRUE, # if TRUE then plot is created on current graphics device
  col = group_colors # assigning our colors for each genotype
)

legend(
  "topright",
  pch = 16,
  legend = c("clean","dirty","neither"),
  col = c("deepskyblue","chocolate4","gray"),
  cex = 0.8
)

title("Top 10 Genes")

```




---

#### LINEAR MODELS

---

Fit linear models for comparisons of interest. Linear modeling in limma is carried out using the lmFit and

contrasts.fit functions.

What do lm.fit and contrasts.fit do? 1) Fit a separate model to the expression values for each gene. 2) An empirical Bayes moderation is carried out by borrowing info across all the gene to obtain more precise estimates of gene-wise variability. 3) The model's residual variances are plotted against average expression values in the next figure.

It can be seen from this plot that the variance is no longer dependent on the mean expression level.

```
# Fits linear model for each gene given a series of arrays
fit <- lmFit(
  v, # object containing log-expression values for a series of arrays
  design # design matrix
)
```

Contrast design for differential expression.

```
contrasts <-
  makeContrasts(
    dirty_vs_clean = dirty - clean,
    levels = colnames(design)
  )

head(contrasts)

##          Contrasts
## Levels      dirty_vs_clean
##   clean              -1
##   dirty               1
## groupneither        0
## sexM                0
## age30.39             0
## age40.49             0
```

Assign allComparisons to a vector for later use.

```
allComparisons <- colnames(contrasts)
allComparisons # check

## [1] "dirty_vs_clean"
```

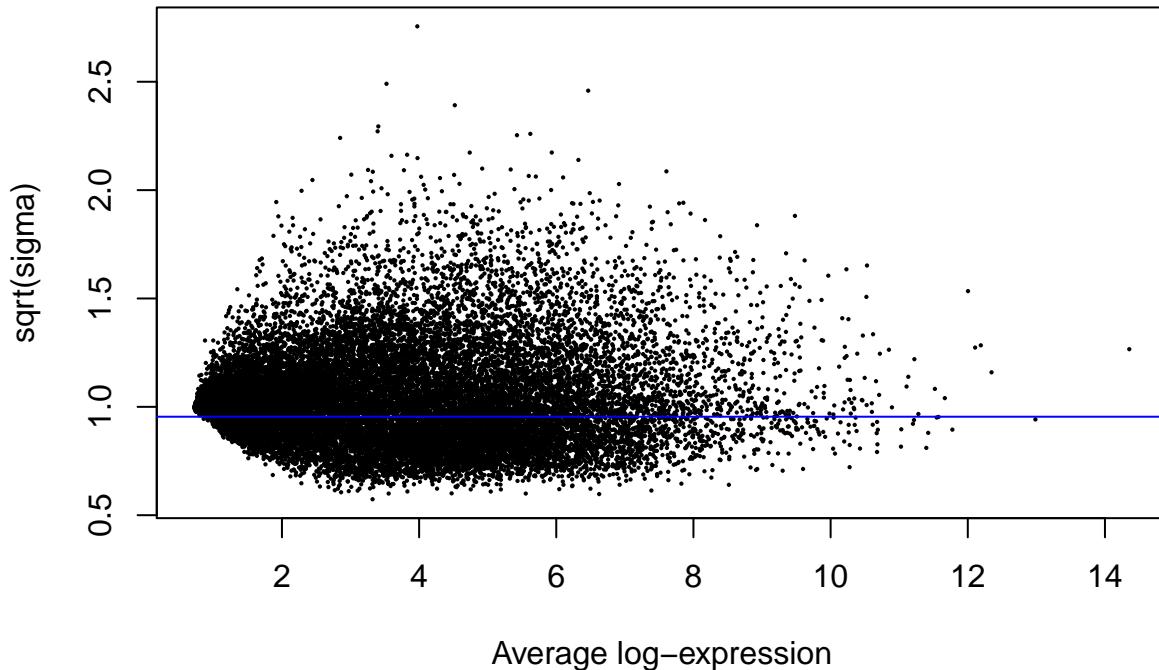
Run contrast analysis. Given a linear model fit, compute estimated coefficients and standard errors for a given set of contrasts.

```
vfit <- contrasts.fit(fit, contrasts = contrasts)
```

Compute differential expression based on the empirical Bayes moderation of the standard errors towards a common value.

```
veBayesFit <- eBayes(vfit)
plotSA(veBayesFit, main = "Final Model: Mean-variance Trend")
```

## Final Model: Mean–variance Trend



`decideTests()` identifies which genes are significantly differentially expressed for each contrast from a fit object containing p-values and test statistics.

```
sumTable <-  
  summary(decideTests(  
    vfit, # object  
    # by default the method = "separate"  
    adjust.method = "BH",  
    p.value = 0.05,  
    lfc = 1 # numeric, minimum absolute log2-fold change required  
  ))  
  
head(sumTable)  
  
##          dirty_vs_clean  
## Down              25  
## NotSig            24236  
## Up               69
```

### DEG TABLES

For each comparison... extract table of the top-ranked genes (DEGs) from a linear model fit AND output a table.

NOTE: Log2FC of 1 is equivalent to linear fold change of 2.

```
coef = 1  
for (i in allComparisons) {  
  
  # p < 1, log2fc > 0 -----
```

```

vTopTableAll <-
  topTable(
    veBayesFit,
    coef = coef,
    n = Inf,
    p.value = 1,
    lfc = 0
  )
#output as txt file
path <- paste("DEGs/DEGs_TTR_", i, "_FDRq1_Log2FC0.txt", sep = "")
write.table(
  vTopTableAll,
  path,
  sep = "\t",
  row.names = TRUE,
  quote = FALSE
)

# p < 0.05, log2fc > 1 -----
vTopTable1 <-
  topTable(
    veBayesFit,
    coef = coef,
    n = Inf,
    p.value = 0.05,
    lfc = 1
  )
path <- paste("DEGs/DEGs_TTR_", i, "_FDRq0.05_Log2FC1.txt", sep = "")
write.table(
  vTopTable1,
  path,
  sep = "\t",
  row.names = TRUE,
  quote = FALSE
)

# increment the coefficient -----
coef <- coef + 1
}

```

---

### -VOLCANO PLOTS-

Using the DEG tables we made, graph a volcano plot. Read in file.

```

dirty_vs_clean <-
  read.table(
    "DEGs/DEGs_TTR_dirty_vs_clean_FDRq1_Log2FC0.txt",
    header = TRUE,
    sep = "\t",
    stringsAsFactors = F
  )

```

```

dirty_vs_clean$gene_symbol <- rownames(dirty_vs_clean)

summary(decideTests(
  vfit, # object
  adjust.method = "BH", # by default the method = "separate"
  p.value = 0.05,
  lfc = 1
))

##          dirty_vs_clean
## Down              25
## NotSig           24236
## Up                69

color_values <- vector()
max <- nrow(dirty_vs_clean)

for(i in 1:max){
  if (dirty_vs_clean$adj.P.Val[i] < 0.05){
    if (dirty_vs_clean$logFC[i] > 1){
      color_values <- c(color_values, 1) # 1 when logFC > 1 and pval < 0.05
    }
    else if (dirty_vs_clean$logFC[i] < -1){
      color_values <- c(color_values, 2) # 2 when logFC < -1 and pval < 0.05
    }
    else{
      color_values <- c(color_values, 3) # 3 when -1 <= logFC <= 1 pval < 0.05
    }
  }
  else{
    color_values <- c(color_values, 3) # 3 when pval >= 0.05
  }
}

dirty_vs_clean$color_p0.05_lfc1 <- factor(color_values)

hadjpval <- (-log10(max(dirty_vs_clean$adj.P.Val[dirty_vs_clean$adj.P.Val < 0.05], na.rm=TRUE)))
hadjpval

## [1] 1.301873

p <-
  ggplot(data = dirty_vs_clean,
         aes(x = logFC, # x-axis is logFC
              y = -log10(adj.P.Val), # y-axis will be log10 of adj.P.Val
              color = color_p0.05_lfc1)) + # color is based on factored Color column
  geom_point(size = 2) + # create scatterplot, alpha makes points transparent
  theme_bw() +
  theme(legend.position = "none") + # no legend
  scale_color_manual(values = c("red", "blue", "grey")) + # set factor colors
  labs(

```

```

    title = "", # no main title
    x = expression(log[2](FC)), # x-axis title
    y = expression(-log[10] ~ "(FDR adjusted " ~ italic("p") ~ "-value)") # y-axis title
) +
theme(plot.title = element_text(size=12)) +
theme(axis.title.x = element_text(size = 12),
      axis.text.x = element_text(size = 12)) +
theme(axis.title.y = element_text(size = 12),
      axis.text.y = element_text(size = 12)) +
geom_hline(yintercept = hadjpval,
            colour = "#000000",
            linetype = "dashed") +
geom_vline(xintercept = 1,
            colour = "#000000",
            linetype = "dashed") +
geom_vline(xintercept = -1,
            colour = "#000000",
            linetype = "dashed") +
ggtitle("GTEEx\nFDRq < 0.05, -1 > LogFC > 1") +
geom_text_repel(data = subset(dirty_vs_clean,
                               dirty_vs_clean$color_p0.05_lfc1 == 1),
                aes(x = logFC, y= -log10(adj.P.Val), label = gene_name),
                color = "maroon",
                fontface="italic",
                max.overlaps = getOption("ggrepel.max.overlaps",
                                         default = 55)) +
geom_text_repel(data = subset(dirty_vs_clean,
                               dirty_vs_clean$color_p0.05_lfc1 == 2),
                aes(x = logFC, y= -log10(adj.P.Val), label = gene_name),
                color = "navyblue",
                fontface="italic",
                max.overlaps = getOption("ggrepel.max.overlaps",
                                         default = 30))

```

P

```

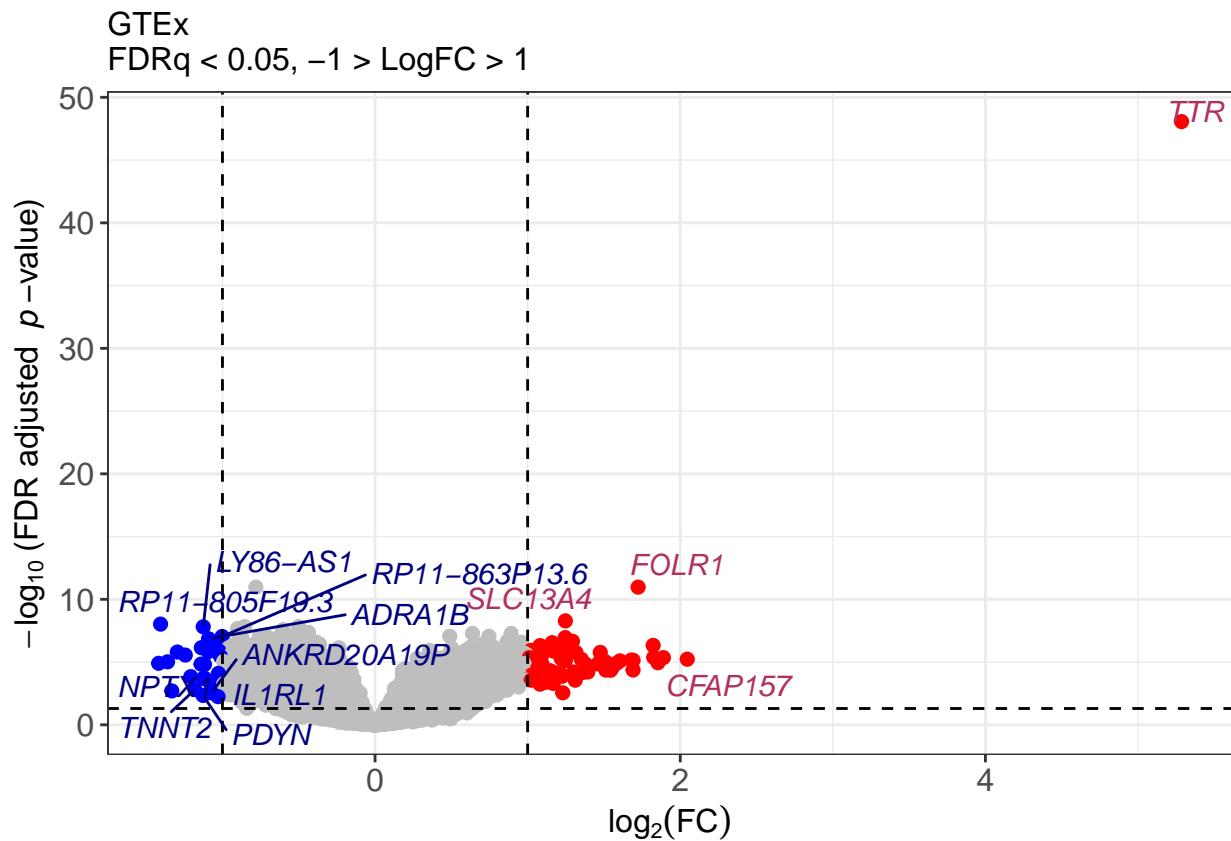
## Warning: ggrepel: 65 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps

```

```

## Warning: ggrepel: 16 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps

```



```
setwd("/Users/m214960/Documents/GTEx")
pdf("Figures/GTEx_TTR_volcano.pdf", width = 4, height = 4)
p
```

```
## Warning: ggrepel: 66 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```

```
## Warning: ggrepel: 19 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```