RNA-seq of AM and MDM infected with TB

Data cleaning

Kim Dill-McFarland, kadm@uw.edu

version April 16, 2020

Contents

Background **1**

Setup **1**

Read in and format data **2**
Summarize samples . . . 2

Data cleaning **3**
Sample filtering . . . 3
Summarize sample filtering . . . 6
Separate cell types . . . 6
Gene filtering - AM samples . . . 7
Gene filtering - MDM samples . . . 13

R session **18**

Background

Alveolar macrophages (AM) and monocyte-derived macrophages (MDM) were obtained from 6 donors and cultured with or without TB.

The purpose of this workflow is to complete basic data cleaning of metadata and RNA-seq libraries generated from the above experiments. This includes 1) removing low quality libraries, 2) removing outlying libraries, 3) filtering rare genes, and 4) normalizing for RNA composition and voom.

Setup

Load packages

```
# Data manipulation and figures
library(tidyverse)
  # Modify ggplot data order within facets
  library(drlib)

# Empirical analysis of digital gene expression data
## Data normalization
library(edgeR)
```

```r
#Create 'not in' operator
`%notin%` <- Negate(`%in%`)
```

Set seed

```r
set.seed(927)
```

Read in and format data

```r
meta <- read_csv("data/AM.MDM.data.cleaning.metrics.csv")
```

Cleaning metrics

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   sampID = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

Counts Incudes only exons.

```r
counts <- read_csv("data/AM.MDM.counts.paired.clean.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   geneName = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

Sample metadata Extracted from sample ID in metrics file. Other projects may have a separate table that needs to be loaded with a `read_` function.

```r
samp <- meta %>%
  select(sampID) %>%
  #Separate sampID into multiple columns
  separate(col = sampID, into = c("ptID", "cell", "TB"), sep="_",
           #Keep original sampID column
           remove = FALSE) %>%
  #Modify ptID to remove leading "AM" text
  mutate(ptID = gsub("AM", "", ptID))
```

Summarize samples

```
## # A tibble: 4 x 3
##   cell  TB        n
##   <chr> <chr> <int>
## 1 AM    Media     6
## 2 AM    TB        6
## 3 MDM   Media     6
## 4 MDM   TB        6
```

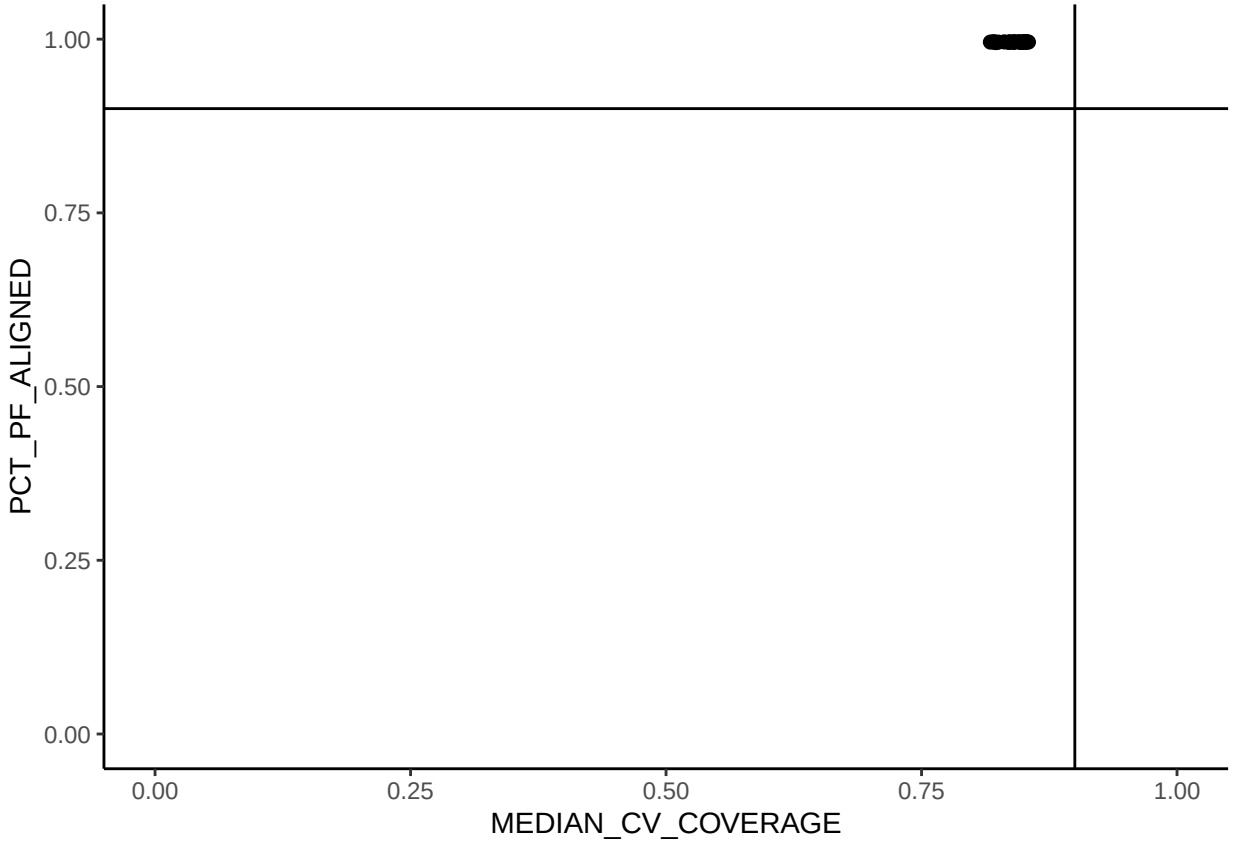
Data cleaning

Sample filtering

Assess median CV coverage vs. alignment percentage

You want median CV coverage to be low, indicating similar coverage of all genes in a sample. You want high alignment percentage, indicating that the reads in a sample were high-quality and successfully aligned to the genome.

```
#Plot the two metrics against each other
meta %>%
  ggplot(aes(x=MEDIAN_CV_COVERAGE, y=PCT_PF_ALIGNED)) +
  geom_point(size=2) +
  theme_classic() +
  #Force axes to go from 0 to 1
  lims(x=c(0,1), y=c(0,1)) +
  #Add horizontal/vertical lines at desired quality cutoffs
  geom_hline(yintercept = 0.9) +
  geom_vline(xintercept = 0.9)
```

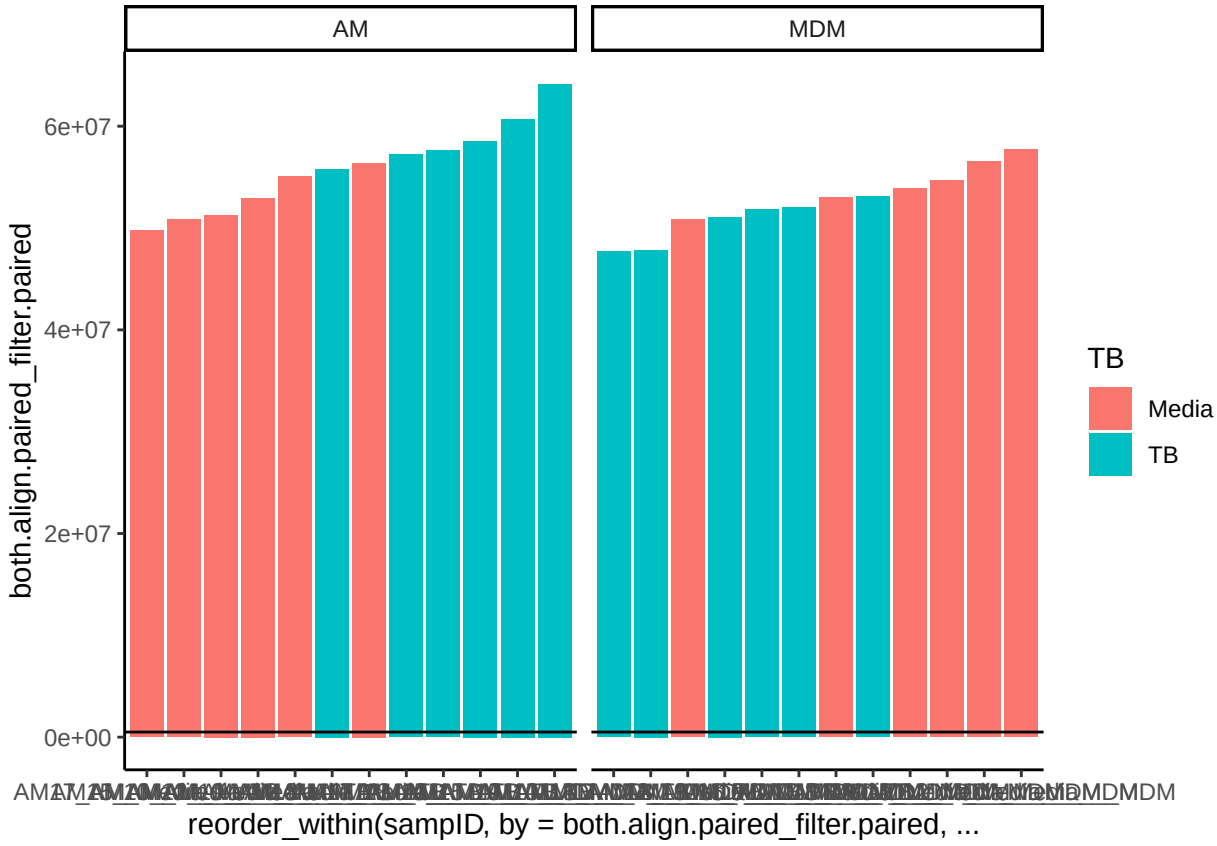


Assess total sequences

Commonly, a minimum of 500,000 sequences (horizontal line below) is needed for reasonably coverage of the human genome.

```
#Order by min to max sequences in each cell type
meta %>%
  #Create samp variables for coloring plot
```

```
  separate(col = sampID, into = c("ptID", "cell", "TB"), sep="_",
           remove = FALSE) %>%

  ggplot(aes(
    #Order sampID (x axis) by the total seqs within each cell type
    x=reorder_within(sampID,
                     by=both.align.paired_filter.paired,
                     within=cell),
    #Set y variable
    y=both.align.paired_filter.paired,
    #Fill bars by TB status
    fill=TB)) +
  #Column plot
  geom_col() +
  theme_classic() +
  # Facet to separate cell types into 2 adjoining plots
  # format y ~ x, columns ~ rows
  facet_wrap(~cell, scales = "free_x") +
  #Add cutoff line at 500,000
  geom_hline(yintercept = 500000)
```



Filter by quality

These data are high-quality and no samples need to be removed. However, if this were not the case, we would use the 2 plots to define reasonable quality cutoffs and remove any samples that fail to meet those cutoffs. You need to do with before PCA assessment as poor-quality samples will impact PC calculations.

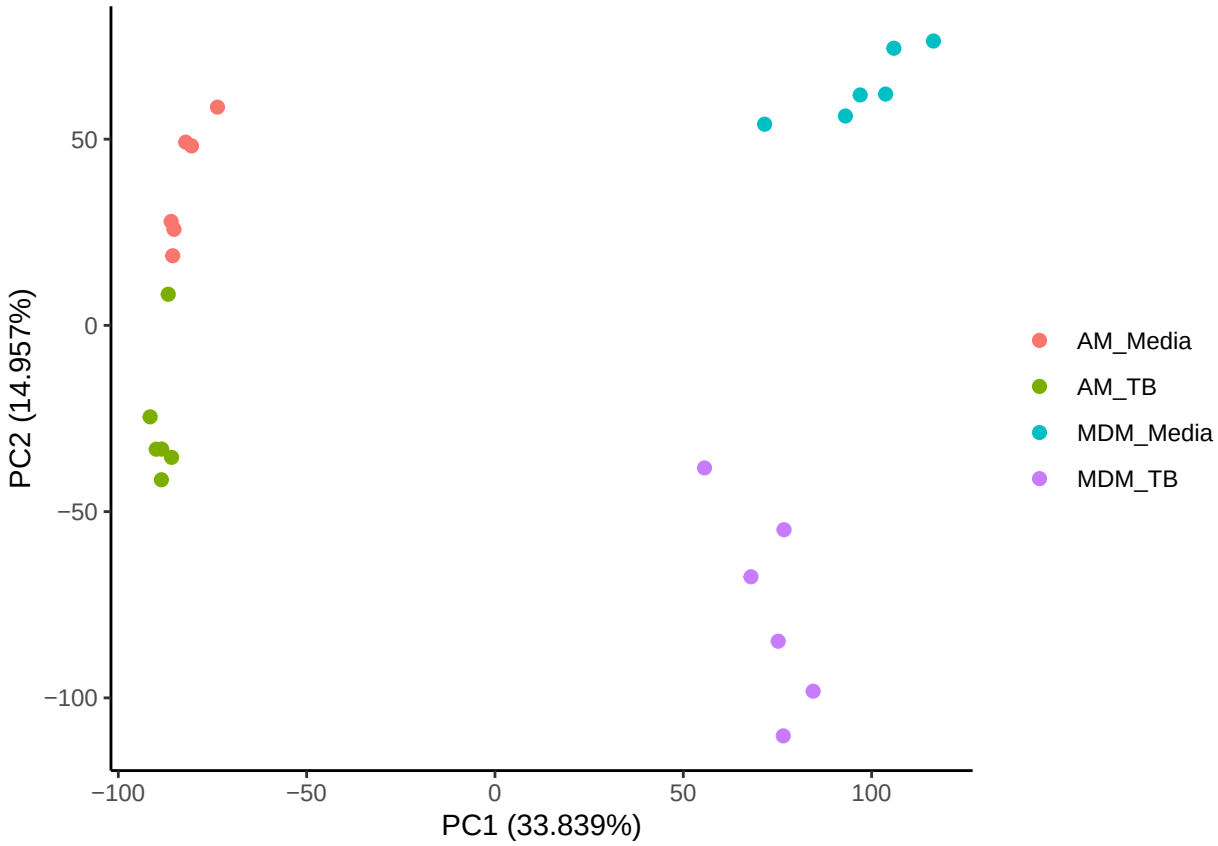
Assess PCA outliers

An outlier is generally defined as a sample that is > 3 standard deviations away from the group mean on any PC axis. However, this is not a hard rule and generally, you can look at the PCA and easily ID potential outliers.

```
PCA <- counts %>%
  #Put geneName column into rownames b/c prcomp does not allow non-numeric columns
  column_to_rownames("geneName") %>%
  #Convert to log2 counts per million
  cpm(log=TRUE) %>%
  #transpose table
  t() %>%
  #Calculate the PCA
  prcomp()

#Extract the % variation explained by each axis
summary(PCA)$importance
```

```
##                             PC1      PC2      PC3      PC4      PC5      PC6
## Standard deviation     88.12615 58.58885 41.54851 34.71198 32.74331 27.61458
## Proportion of Variance  0.33839  0.14957  0.07522  0.05250  0.04671  0.03323
## Cumulative Proportion   0.33839  0.48796  0.56318  0.61568  0.66239  0.69562
##                             PC7      PC8      PC9     PC10     PC11     PC12
## Standard deviation     25.99336 25.44649 24.16821 22.63999 22.09265 21.06364
## Proportion of Variance  0.02944  0.02821  0.02545  0.02233  0.02127  0.01933
## Cumulative Proportion   0.72506  0.75327  0.77872  0.80106  0.82232  0.84166
##                            PC13     PC14     PC15     PC16     PC17     PC18
## Standard deviation     20.54934 19.81623 19.31023 18.41286 18.16348 17.76465
## Proportion of Variance  0.01840  0.01711  0.01625  0.01477  0.01437  0.01375
## Cumulative Proportion   0.86006  0.87717  0.89341  0.90819  0.92256  0.93631
##                            PC19     PC20     PC21     PC22     PC23         PC24
## Standard deviation     17.67967 17.57731 17.08206 16.78604 16.32696 4.73723e-13
## Proportion of Variance  0.01362  0.01346  0.01271  0.01228  0.01162 0.00000e+00
## Cumulative Proportion   0.94993  0.96339  0.97611  0.98838  1.00000 1.00000e+00
```

```
#Extract PC values to use in plot
#For PCA$x to be a data frame so it can work with tidyverse functions
PCA.dat <- as.data.frame(PCA$x) %>%
  #Take rownames and move to a data column named sampID
  rownames_to_column("sampID") %>%
  #Merge with sample metadata, matching rows based on sampID
  full_join(samp, by="sampID")

PCA.dat %>%
  #Create color variable to show both cell type and TB status
  mutate(color.var = paste(cell, TB, sep="_")) %>%

  ggplot(aes(x=PC1,y=PC2, color=color.var)) +
  geom_point(size=2) +
  theme_classic() +
  #Change axes labels
  #Use importance info to label PCA
  labs(x = "PC1 (33.839%)", y = "PC2 (14.957%)",
       #Change legend label
       color="")
```



Filter outliers

There are no outliers in these data. If there were, you would stepwise remove each outlier, starting with the furtherest, and assess PCA after each removal. Since PCA is relative to all samples in the data, this stepwise method is necessary as apparent outliers may disappear or new outliers may appear when a further outlier is removed.

Summarize sample filtering

No samples were removed based on quality.

Separate cell types

Since strong differences are apparent between cell types, gene filtering should be completed separately. This ensures that genes that are lowly abundant or rare in one cell type are not filtered in the other cell type where they are more abundant. Importantly, if our research question was to compare AM and MDM, they would need to be gene filtered together with different cutoffs that below.

Separate samples by cell type.

```
#Filter sample metadata
#List samples IDs in groups of interest
AM.vec <- samp %>%
  filter(cell == "AM") %>%
  select(sampID) %>%
  unlist(use.names = FALSE)

MDM.vec <- samp %>%
```

```
  filter(cell == "MDM") %>%
  select(sampID) %>%
  unlist(use.names = FALSE)

#Select groups from count data
counts.AM <- counts %>%
  select(geneName, all_of(AM.vec))

counts.MDM <- counts %>%
  select(geneName, all_of(MDM.vec))
```

Gene filtering - AM samples

Filter protein coding (pc) genes

Load key and filter to pc genes with valid HGNC names.

Note that I've changed the separator used when combining the HGNC symbols. I found that a couple genes contain an "_" and thus, do not want to use this as a separator.

Note that I also added a filter to keep only genes that are present in the count data. This removes the need to filter the key when we make the DGEList object next.

```
# Filtering to protein coding
key <- read_tsv("data/EnsemblToHGNC_GRCh38.txt",
                na=c(NA, "", ".", "N/A")) %>%
  filter(gene_biotype == "protein_coding" &
           !is.na(hgnc_symbol)) %>%
  #remove duplicates if exist
  distinct() %>%
  #Sort by name
  rename(geneName = ensembl_gene_id) %>%
  arrange(geneName) %>%
  #Combine duplicate annotations
  group_by(geneName, gene_biotype) %>%
  summarize(hgnc_symbol = paste(hgnc_symbol, collapse=":")) %>%
  #Keep only genes present in count data
  filter(geneName %in% counts$geneName) %>%
  arrange(geneName)
```

```
## Parsed with column specification:
## cols(
##   ensembl_gene_id = col_character(),
##   gene_biotype = col_character(),
##   hgnc_symbol = col_character()
## )
```

View ENSEMBL IDs with multiple HGNC names.

```
key %>% filter(grepl(":", hgnc_symbol))
```

```
## # A tibble: 1 x 3
## # Groups:   geneName [1]
##   geneName        gene_biotype   hgnc_symbol
##   <chr>           <chr>          <chr>
## 1 ENSG00000269433 protein_coding OPN1MW2:OPN1MW
```

Filter count table to protein coding genes.

```
counts.AM.pc <- counts.AM %>%
  filter(geneName %in% key$geneName)
```

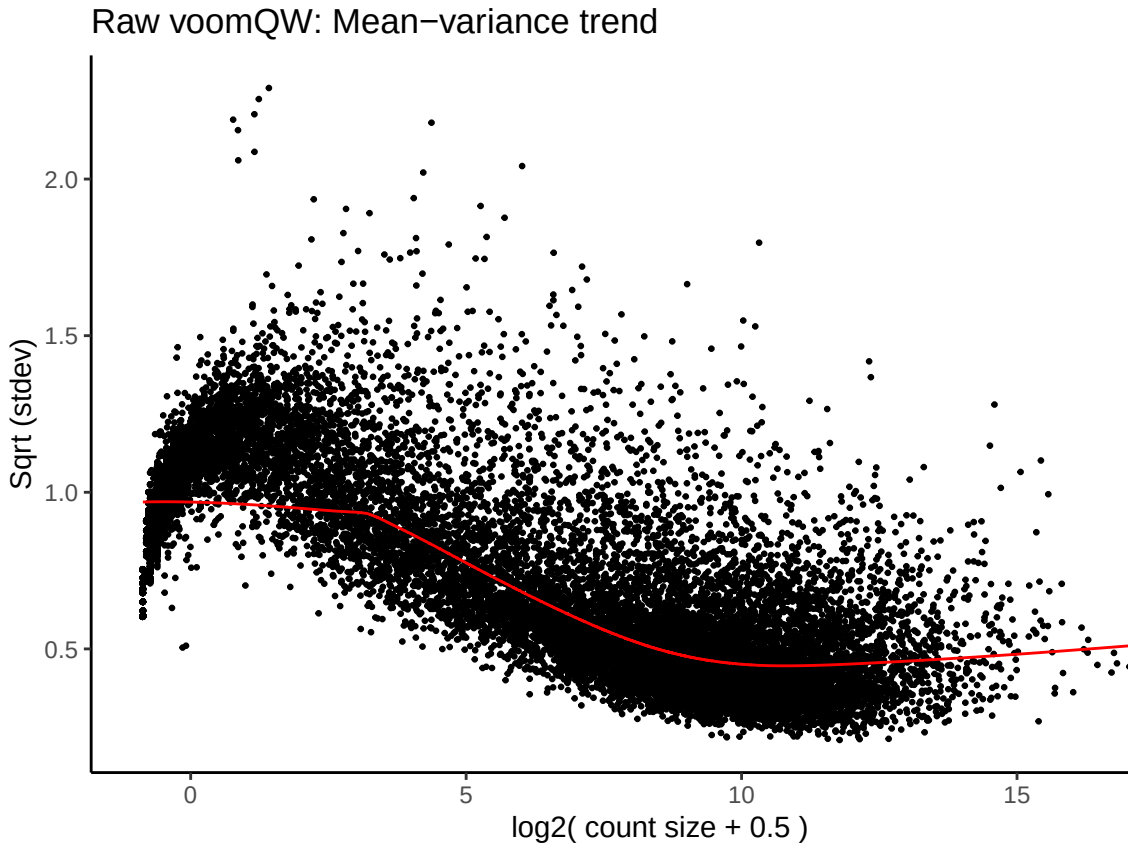
Create DGEList object

For use in gene filtering with edgeR

```
dat.AM.pc <- DGEList(
  #Move geneName data to rownames and convert to matrix
  counts = as.matrix(column_to_rownames(counts.AM.pc, "geneName")),
  #Filter to AM samples and convert to matrix
  samples = as.matrix(filter(samp, cell == "AM")),
  genes = key)
```

Filter rare genes

```
# Plot distribution of raw genes
temp <- voomWithQualityWeights(
  counts = dat.AM.pc,
  design = model.matrix(~TB, data = dat.AM.pc$samples),
  plot=FALSE, save.plot=TRUE
)

MV.plot1 <- data.frame(
  x = temp$voom.xy$x,
  y = temp$voom.xy$y,
  linex = temp$voom.line$x,
  liney = temp$voom.line$y) %>%

  ggplot() +
  geom_point(aes(x=x, y=y), size=0.5) +
  geom_path(aes(x=linex, y=liney), color="red") +
  theme_classic() +
  labs(x="log2( count size + 0.5 )", y="Sqrt (stdev)",
       title="Raw voomQW: Mean-variance trend")

MV.plot1
```



Assess mean-variance

The goal is to reduce the downward tail on the left of the plot. These genes are low mean abundance (x-axis) and low variance (y-axis). We want to remove these to:

1. Reduce zeros in the data set
2. Improve statistical power in later analyses as a result of fewer genes (= lower FDR penalty)

You don't need to remove 100% of the tail. Instead, try a number of cutoffs until you reach a reasonable medium between data loss and quality of retained data.

Filter with edgeR We won't use this function as I find it's too strict for simple experimental designs like the media vs. TB setup here.

Using `edgeR::filterByExpr( )`, you keep only non-rare genes that have:

1. at least `min.count` reads scaled to counts per million of the median library size (k) in the number of samples equal to the smallest group of interest in the statistical model
2. at least `min.total.count` total reads across all samples in the data

Example with the default values.

```
#Determine abundant genes
genes.to.keep <- filterByExpr(
  dat.AM.pc,
  design=model.matrix(~TB, data=dat.AM.pc$samples),
  min.count=10,
  min.total.count=15)

# Filter data
dat.AM.pc.abund2 <- dat.AM.pc[genes.to.keep,]
```

Filter by min count in min number of samples I developed a function that filters genes to those with `min.CPM` (counts per million) in at least `min.pct` (percent of samples) or `min.sample` (number of samples). It does not take the experimental design into account.

```
#Get Kim's function from GitHub
source("https://raw.githubusercontent.com/kdillmcfarland/R_bioinformatic_scripts/master/RNAseq_rare_gene

#Filter
rare.gene.filter(dat = dat.AM.pc,
                 min.sample = 3,
                 min.CPM = 0.5,
                 name = "dat.AM.pc.abund")
```

Re-assess mean-variance The above filtering removed:

```
#Total genes
nrow(dat.AM.pc$genes)-nrow(dat.AM.pc.abund$genes)
```

```
## [1] 6232
```

```
#Percent of genes
(nrow(dat.AM.pc$genes)-nrow(dat.AM.pc.abund$genes))/nrow(dat.AM.pc$genes)*100
```

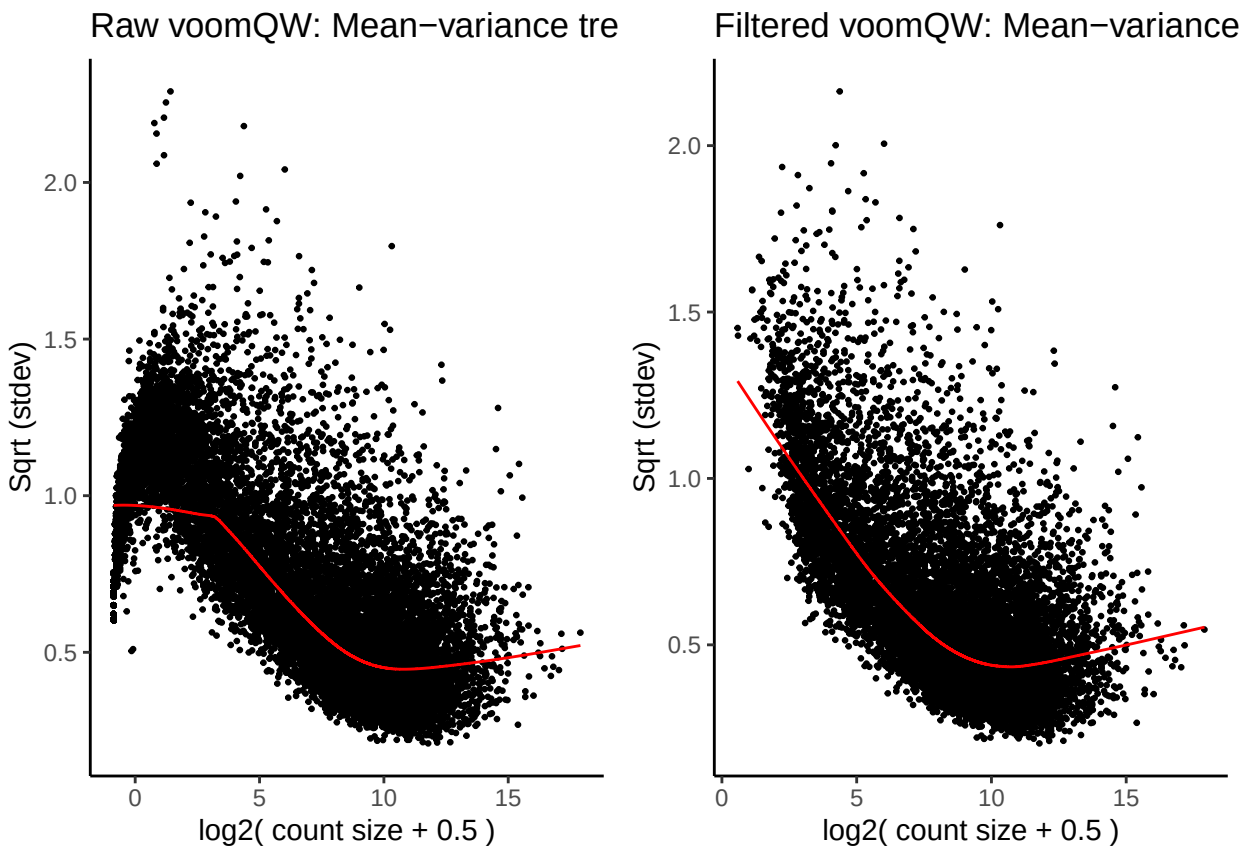
```
## [1] 32.65563
```

We look at the mean-variance again to see how the tail has changed.

```
# Plot distribution of raw genes
temp <- voomWithQualityWeights(
  counts = dat.AM.pc.abund,
  design = model.matrix(~TB, data = dat.AM.pc.abund$samples),
  plot=FALSE, save.plot=TRUE
)

MV.plot2 <- data.frame(
  x = temp$voom.xy$x,
  y = temp$voom.xy$y,
  linex = temp$voom.line$x,
  liney = temp$voom.line$y) %>%

  ggplot() +
  geom_point(aes(x=x, y=y), size=0.5) +
  geom_path(aes(x=linex, y=liney), color="red") +
  theme_classic() +
  labs(x="log2( count size + 0.5 )", y="Sqrt (stdev)",
       title="Filtered voomQW: Mean-variance trend")

#Plot the orig and filtered plots together
library(cowplot)
```

```
##
## ************************************************************
## Note: As of version 1.0.0, cowplot does not change the
##   default ggplot2 theme anymore. To recover the previous
##   behavior, execute:
##   theme_set(theme_cowplot())
```

```
## ********************************************************
plot_grid(MV.plot1, MV.plot2)
```



Normalize for RNA composition

Calculate factors to scale library sizes.

```
dat.AM.pc.abund.norm <- calcNormFactors(dat.AM.pc.abund)
```

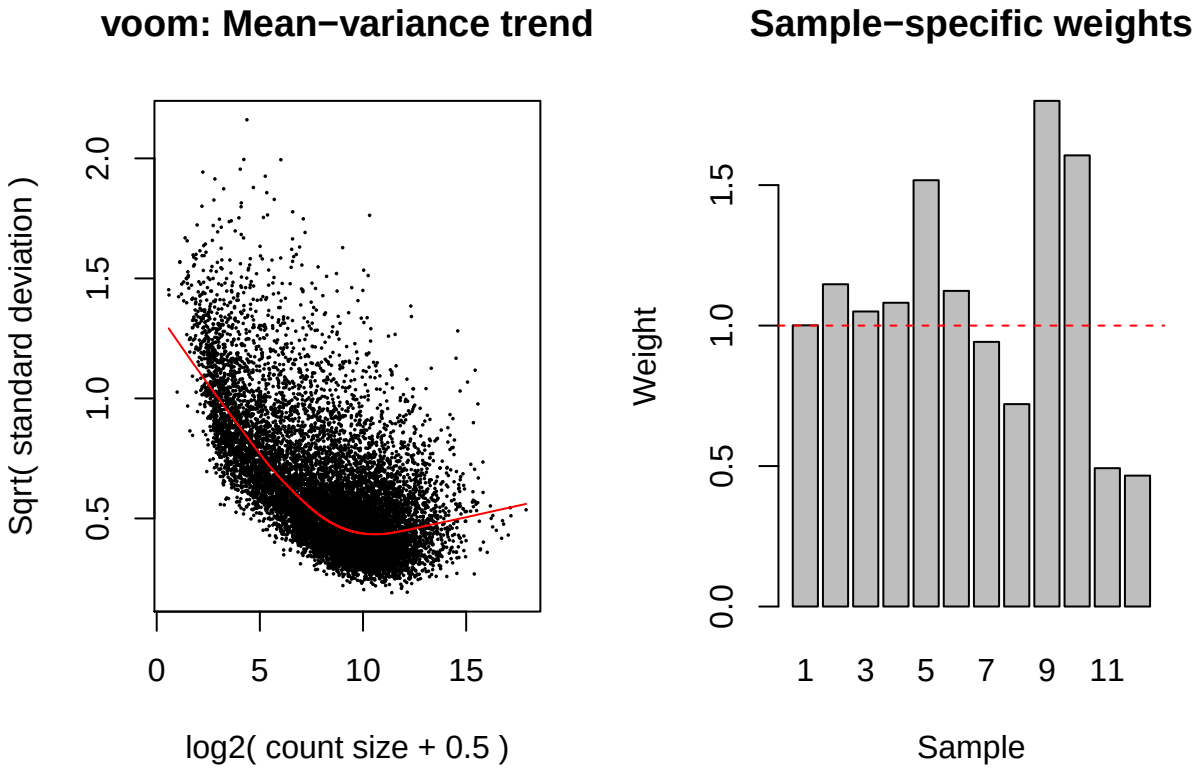
Normalize with voom

Allow the voomQW function to plot the default plot instead of our.

```
dat.AM.pc.abund.norm.voom <- voomWithQualityWeights(
  dat.AM.pc.abund.norm,
  design=model.matrix(~TB, data = dat.AM.pc.abund.norm$samples),
  plot=TRUE)
```

Table 1: AM data cleaning summary

Samples removed	
High median CV coverage	0
Low alignment percentage	0
Low total sequences	0
PCA outlier	0
Genes removed	
Rare	6232

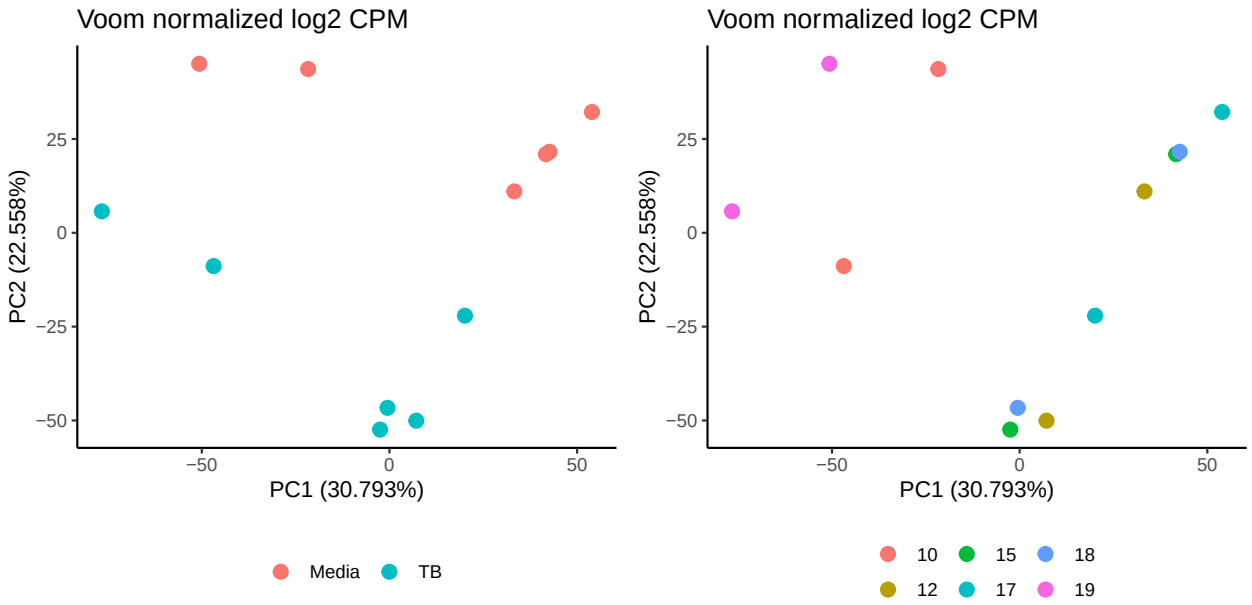


Summarize AM cleaning

Filtering

```
##
## Attaching package: 'kableExtra'
```

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



PCA

Save AM data Write as a single RData to easily load everything into R.

```
save(dat.AM.pc.abund.norm.voom,
     file="data/AM.clean.RData")
```

Write counts as table for use in other programs.

```
write_csv(
  as.data.frame(dat.AM.pc.abund.norm.voom$E),
  path = "data/AM.clean.counts.csv")
```

Gene filtering - MDM samples

Filter protein coding (pc) genes

Use same key as MDM samples. Filter count table to protein coding genes.

```
counts.MDM.pc <- counts.MDM %>%
  filter(geneName %in% key$geneName)
```

Create DGEList object

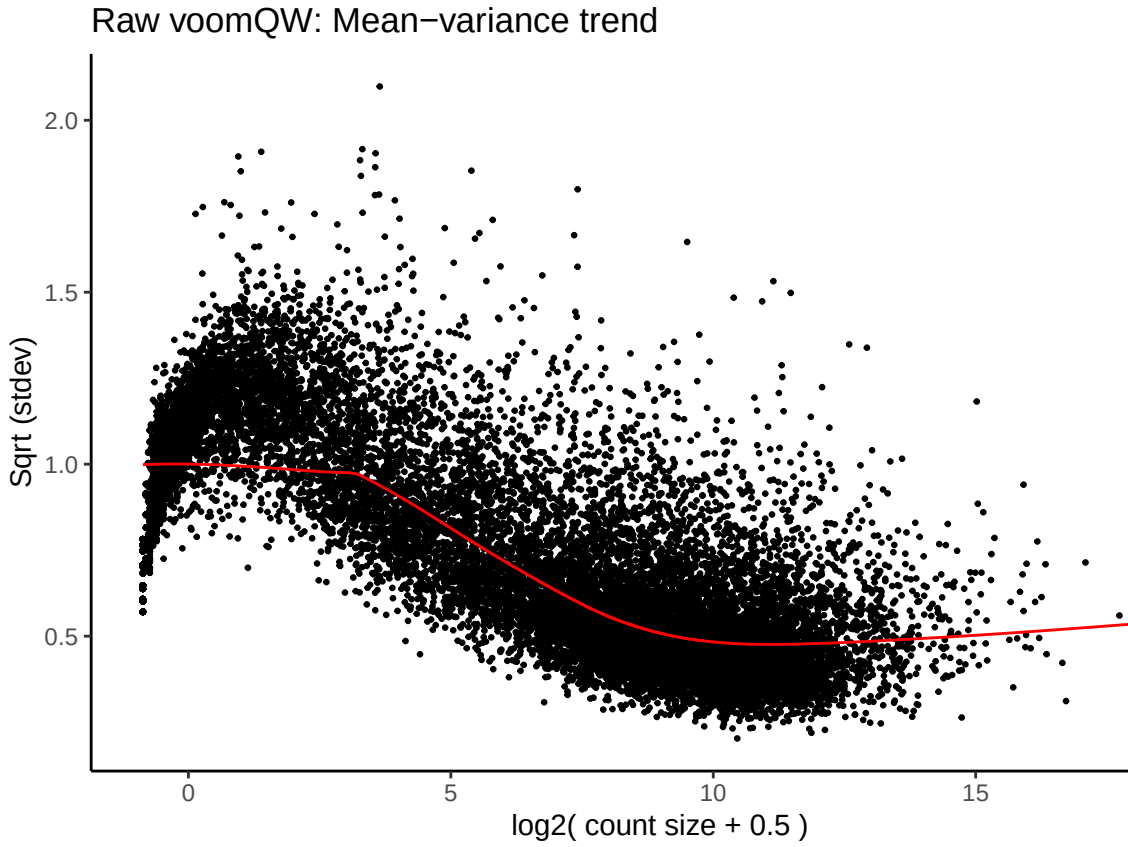
For use in gene filtering with edgeR

```
dat.MDM.pc <- DGEList(
  #Move geneName data to rownames and convert to matrix
  counts = as.matrix(column_to_rownames(counts.MDM.pc, "geneName")),
  #Filter to MDM samples and convert to matrix
  samples = as.matrix(filter(samp, cell == "MDM")),
  genes = key)
```

Filter rare genes

```

# Plot distribution of raw genes
temp <- voomWithQualityWeights(
  counts = dat.MDM.pc,
```

```
  design = model.matrix(~TB, data = dat.MDM.pc$samples),
  plot=FALSE, save.plot=TRUE
)

MV.plot1 <- data.frame(
  x = temp$voom.xy$x,
  y = temp$voom.xy$y,
  linex = temp$voom.line$x,
  liney = temp$voom.line$y) %>%

  ggplot() +
  geom_point(aes(x=x, y=y), size=0.5) +
  geom_path(aes(x=linex, y=liney), color="red") +
  theme_classic() +
  labs(x="log2( count size + 0.5 )", y="Sqrt (stdev)",
       title="Raw voomQW: Mean-variance trend")

MV.plot1
```



Assess mean-variance

The goal is to reduce the downward tail on the left of the plot. These genes are low mean abundance (x-axis) and low variance (y-axis). We want to remove these to:

1. Reduce zeros in the data set
2. Improve statistical power in later analyses as a result of fewer genes (= lower FDR penalty)

You don't need to remove 100% of the tail. Instead, try a number of cutoffs until you reach a reasonable medium between data loss and quality of retained data.

Filter with edgeR We won't use this function as I find it's too strict for simple experimental designs like the media vs. TB setup here.

Using `edgeR::filterByExpr( )`, you keep only non-rare genes that have:

1. at least `min.count` reads scaled to counts per million of the median library size (k) in the number of samples equal to the smallest group of interest in the statistical model
2. at least `min.total.count` total reads across all samples in the data

Example with the default values.

```
#Determine abundant genes
genes.to.keep <- filterByExpr(
  dat.MDM.pc,
  design=model.matrix(~TB, data=dat.MDM.pc$samples),
  min.count=10,
  min.total.count=15)

# Filter data
dat.MDM.pc.abund2 <- dat.MDM.pc[genes.to.keep,]
```

Filter by min count in min number of samples I developed a function that filters genes to those with `min.CPM` (counts per million) in at least `min.pct` (percent of samples) or `min.sample` (number of samples). It does not take the experimental design into account.

```
#Get Kim's function from GitHub
source("https://raw.githubusercontent.com/kdillmcfarland/R_bioinformatic_scripts/master/RNAseq_rare_gene

#Filter
rare.gene.filter(dat = dat.MDM.pc,
                 min.sample = 3,
                 min.CPM = 0.5,
                 name = "dat.MDM.pc.abund")
```

Re-assess mean-variance The above filtering removed:

```
#Total genes
nrow(dat.MDM.pc$genes)-nrow(dat.MDM.pc.abund$genes)
```

```
## [1] 6247
```

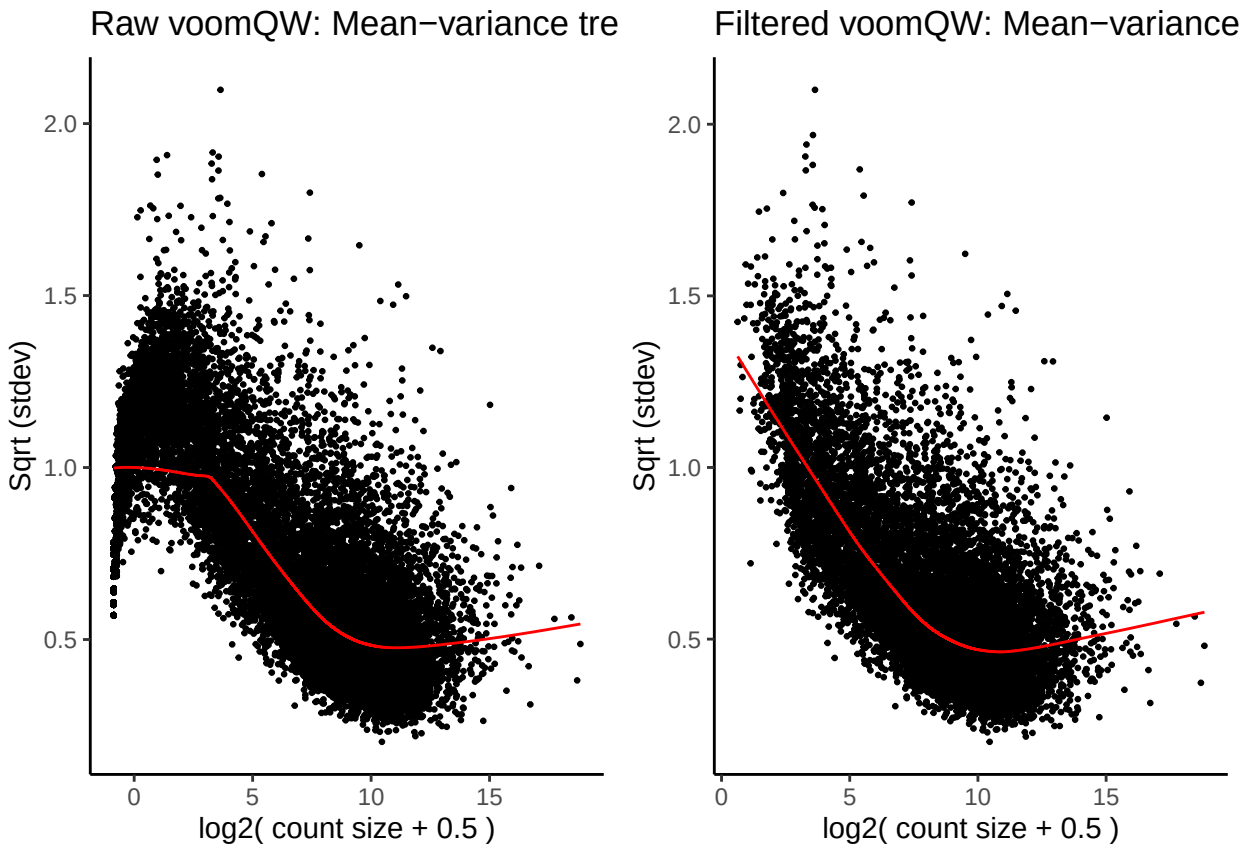
```
#Percent of genes
(nrow(dat.MDM.pc$genes)-nrow(dat.MDM.pc.abund$genes))/nrow(dat.MDM.pc$genes)*100
```

```
## [1] 32.73423
```

We look at the mean-variance again to see how the tail has changed.

```
# Plot distribution of raw genes
temp <- voomWithQualityWeights(
  counts = dat.MDM.pc.abund,
  design = model.matrix(~TB, data = dat.MDM.pc.abund$samples),
  plot=FALSE, save.plot=TRUE
)

MV.plot2 <- data.frame(
  x = temp$voom.xy$x,
  y = temp$voom.xy$y,
```

```
  linex = temp$voom.line$x,
  liney = temp$voom.line$y) %>%

  ggplot() +
  geom_point(aes(x=x, y=y), size=0.5) +
  geom_path(aes(x=linex, y=liney), color="red") +
  theme_classic() +
  labs(x="log2( count size + 0.5 )", y="Sqrt (stdev)",
       title="Filtered voomQW: Mean-variance trend")

#Plot the orig and filtered plots together
library(cowplot)
plot_grid(MV.plot1, MV.plot2)
```



Normalize for RNA composition

Calculate factors to scale library sizes.

```
dat.MDM.pc.abund.norm <- calcNormFactors(dat.MDM.pc.abund)
```

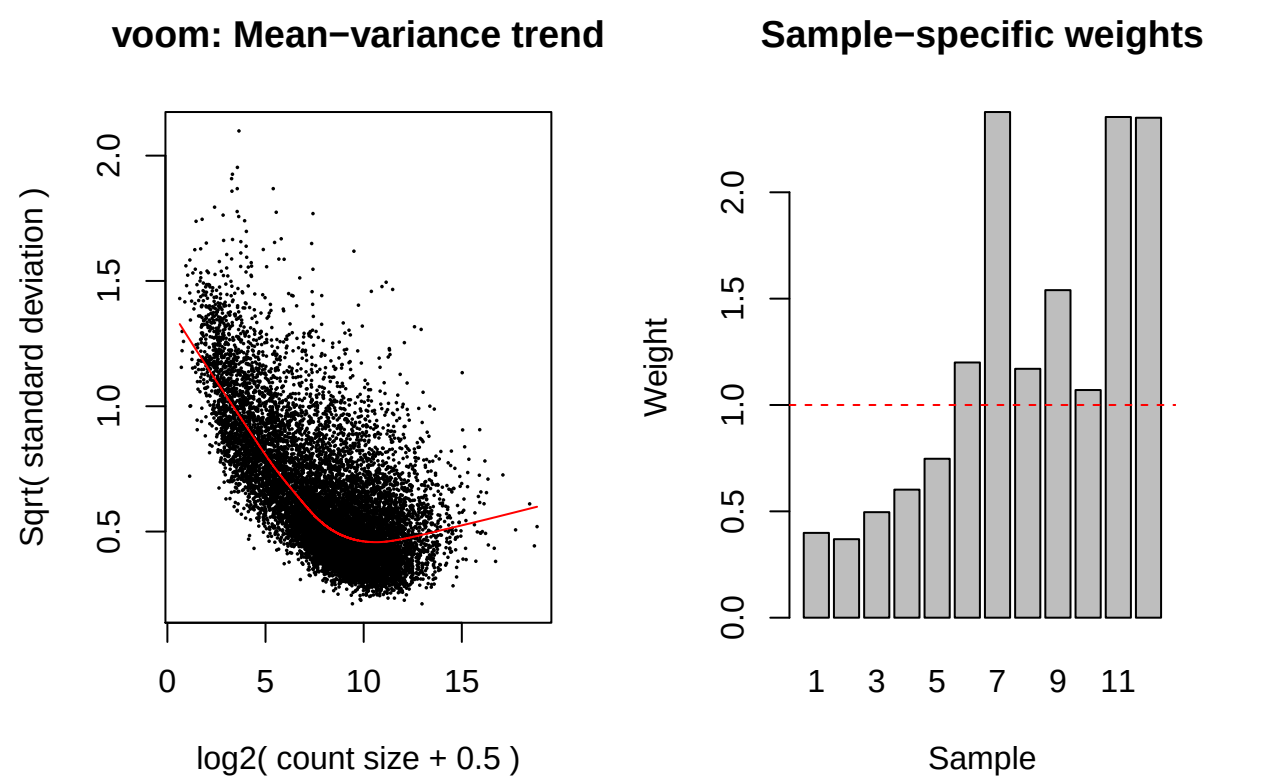
Normalize with voom

Allow the voomQW function to plot the default plot instead of our.

```
dat.MDM.pc.abund.norm.voom <- voomWithQualityWeights(
  dat.MDM.pc.abund.norm,
  design=model.matrix(~TB, data = dat.MDM.pc.abund.norm$samples),
  plot=TRUE)
```

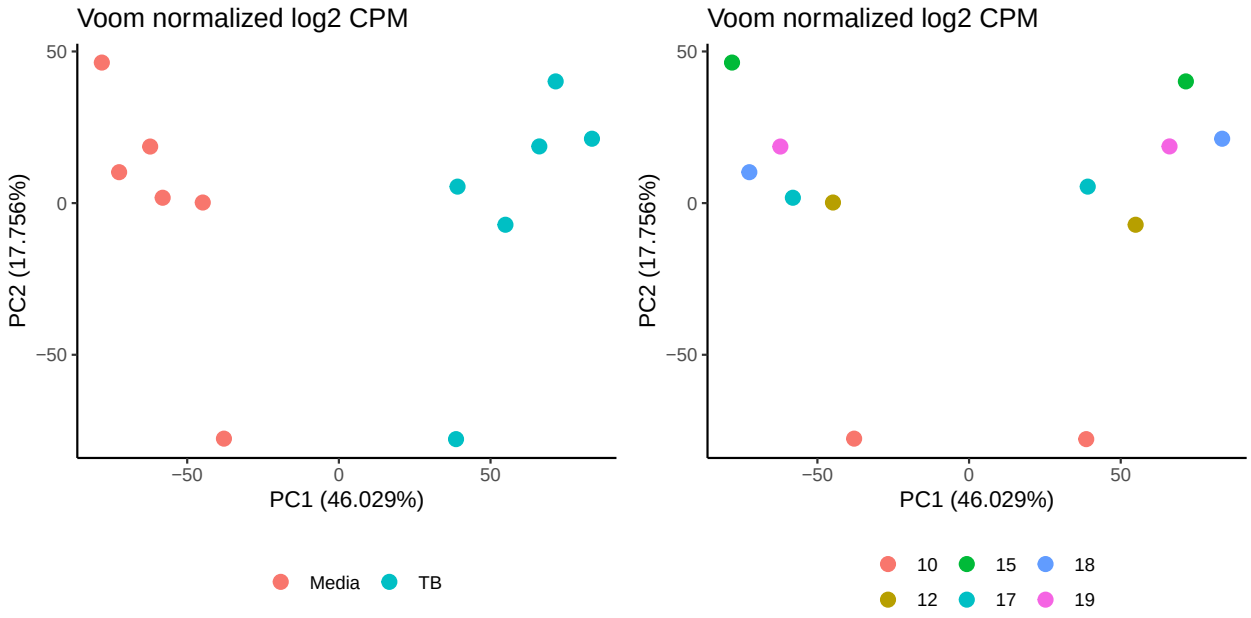
Table 2: MDM data cleaning summary

Samples removed	
High median CV coverage	0
Low alignment percentage	0
Low total sequences	0
PCA outlier	0
Genes removed	
Rare	6247



Summarize MDM cleaning

Filtering



PCA

Save MDM data Write as a single RData to easily load everything into R.

```
save(dat.MDM.pc.abund.norm.voom,
     file="data/MDM.clean.RData")
```

Write counts as table for use in other programs.

```
write_csv(
  as.data.frame(dat.MDM.pc.abund.norm.voom$E),
  path = "data/MDM.clean.counts.csv")
```

R session

```
sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Catalina 10.15.4
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] kableExtra_1.1.0 knitr_1.28       cowplot_1.0.0    edgeR_3.26.8
##  [5] limma_3.40.6     drlib_0.1.1      forcats_0.5.0    stringr_1.4.0
##  [9] dplyr_0.8.5      purrr_0.3.3      readr_1.3.1      tidyr_1.0.2
## [13] tibble_3.0.0     ggplot2_3.3.0    tidyverse_1.3.0
```

```
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.4.6      locfit_1.5-9.4    lubridate_1.7.8   lattice_0.20-41
##  [5] assertthat_0.2.1  digest_0.6.25     utf8_1.1.4        R6_2.4.1
##  [9] cellranger_1.1.0  backports_1.1.6   reprex_0.3.0      evaluate_0.14
## [13] httr_1.4.1        pillar_1.4.3      rlang_0.4.5       readxl_1.3.1
## [17] rstudioapi_0.11   Matrix_1.2-18     rmarkdown_2.1     labeling_0.3
## [21] webshot_0.5.2     munsell_0.5.0     broom_0.5.5       compiler_3.6.1
## [25] modelr_0.1.6      xfun_0.13         pkgconfig_2.0.3   htmltools_0.4.0
## [29] tidyselect_1.0.0  viridisLite_0.3.0 fansi_0.4.1       crayon_1.3.4
## [33] dbplyr_1.4.2      withr_2.1.2       grid_3.6.1        nlme_3.1-147
## [37] jsonlite_1.6.1    gtable_0.3.0      lifecycle_0.2.0   DBI_1.1.0
## [41] magrittr_1.5      scales_1.1.0      cli_2.0.2         stringi_1.4.6
## [45] farver_2.0.3      fs_1.4.1          xml2_1.3.1        ellipsis_0.3.0
## [49] generics_0.0.2    vctrs_0.2.4       tools_3.6.1       glue_1.4.0
## [53] hms_0.5.3         yaml_2.2.1        colorspace_1.4-1  rvest_0.3.5
## [57] haven_2.2.0
```
