Load and Process Dat from E-MTAB files and ROC

Michael A. Gilchrist

17 Jul 2020

Preliminary Information

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

To compile in emacs use M-n e

Purpose

The purpose of this document is to process empirical measurements of mRNA abundances using mRNA-seq technologies. Data is in These measurements are based on multiple measurements. The values are generally the means of the counts. Unfortunately we don't know anything about the sd of the values used to calculate the mean counts.

Load Libraries

```
library(Biostrings) ## process first to avoid conflicts
```

```
## Loading required package: BiocGenerics
```

```
## 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 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, which.max, which.min

## Loading required package: S4Vectors

## Loading required package: stats4

##
## Attaching package: 'S4Vectors'

## The following object is masked from 'package:base':
##
##     expand.grid

## Loading required package: IRanges

## Loading required package: XVector

##
## Attaching package: 'Biostrings'

## The following object is masked from 'package:base':
##
##     strsplit
```

```
library(tidyr)
```

```
##
## Attaching package: 'tidyr'

## The following object is masked from 'package:S4Vectors':
##
##     expand
```

```
library(tibble)
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

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

## The following object is masked from 'package:XVector':
##
##     slice

## The following objects are masked from 'package:IRanges':
##
##     collapse, desc, intersect, setdiff, slice, union

## The following objects are masked from 'package:S4Vectors':
##
##     first, intersect, rename, setdiff, setequal, union

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

## The following objects are masked from 'package:stats':
##
```

```
##     filter, lag
```

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

```
library(stringr)
library(forcats)
library(ggplot2)
library(knitr)
library(ggpubr)
library(ggpmisc)
library(purrr)
```

```
##
## Attaching package: 'purrr'
```

```
## The following object is masked from 'package:XVector':
##
##     compact
```

```
## The following object is masked from 'package:IRanges':
##
##     reduce
```

```
creationInfo <- paste0("\tDate: ", date(), "\n\tLocation: ", sub(".*/AcrossTissue", "AcrossTissue", getw

exportData=FALSE ## Flag for running save(), write.csv() and other output commands
```

Load and Shape Data

ROC

Get gene names

Extract from FASTA file

- There are a bunch of anomolous genes that have a phi of 1 but an elevated sd relative to the rest of genes with that value Turns out they have a width which is not a multiple of 3.
- Note that a gene may have some isoforms that are not multiples of 3, so we don't want to exclude all isoforms of a gene.
- Data that hasn't been filtered for these anomolous genes has the prefix 'unfiltered'.
- Filtered data does not have a prefix.
- ROC analyses should be rerun with these genes filtered out.

```
## Load original FASTA file
## Using Biostrings function which is not a standard df
unfilteredSeqData <- readDNAStringSet("Input/c_elegan.fasta")

## names are really long descriptions.
## NEed to extract relevant part
unfilteredSeqDesc  <- names(unfilteredSeqData)
unfilteredSeqLength <- width(unfilteredSeqData)
unfilteredSeqGeneNames <- sub(".*\\[gene=([^]]+).*", "\\1", unfilteredSeqDesc)
```

```
## verify there's a match for each entry
if(sum(is.na(unfilteredSeqGeneNames)) ==0) print("Every entry matches")
```

```
## [1] "Every entry matches"
```

```
## Verify that all 'names' are unique.
if(length(unfilteredSeqGeneNames) != length(unique(unfilteredSeqGeneNames))) print("Some geneNames appea
```

```
## [1] "Some geneNames appear twice due to isoforms"
```

```
## Filter out whose length is not a multiple of 3
## Keep name of filtered data simple
unfilteredFastaInfo <- tibble(info=names(unfilteredSeqData), geneName=unfilteredSeqGeneNames, length=as

## Create a vector flagging isoforms with proper gene lengths
properLengthIsoformsFlag <- ((unfilteredFastaInfo$length %% 3) == 0)

## Check for MtDNA genes
## Results indicate there are none. Good!

## note seqData uses 'names' while fastaInfo uses 'info'
seqData <- unfilteredSeqData[properLengthIsoformsFlag, ]
fastaInfo <- unfilteredFastaInfo[properLengthIsoformsFlag, ]
```

Import Phi Values from ROC Output

```
## detailed information on phi: posterior mean, posterior mean of log10(phi), etc
## StdError really StdDev of posterior
unfilteredPhiPosteriorInfo <- readr::read_csv("Input/phi.posterior.unlabeled.csv") %>% dplyr::rename(phi

## Not yet filtered for genes that have anomolous lengths (i.e. length mod 3 !=0)
unfilteredPhiData  <- as_tibble(bind_cols(geneName=unfilteredSeqGeneNames, unfilteredPhiPosteriorInfo, )

##
dim(unfilteredPhiData)
plot(unfilteredPhiData$phi, unfilteredPhiData$sd,
     main="Plot Includes 'anomolous' isoforms",
     sub = "anomolous = gene lengths in nts that are not multiples of 3"
     )
phiData <- unfilteredPhiData[properLengthIsoformsFlag, ]

## verify you've filtered correctly
plot(phiData$phi, phiData$sd, main="Plot excludes 'anomolous' isoforms",
     sub = "anomolous = gene lengths in nts that are not multiples of 3"
     )

anomolousLengthIsoformsFlag <- !properLengthIsoformsFlag
anomolousGeneInfo <- unfilteredFastaInfo[anomolousLengthIsoformsFlag, ]
anomolousPhiData  <-unfilteredPhiData[anomolousLengthIsoformsFlag, ]
plot(anomolousPhiData$phi, anomolousPhiData$sd, main="Isoforms Whose Lengths are Not Multiples of Three

## Problem: Isoforms of the same gene exist in the fasta file (and thus phi estimates), but are not par
## Solution: Combine separate estimates using mean or median values of phi

summaryStatsPhiData  <- phiData %>% group_by(geneName) %>% summarize(mean_phi = mean(phi), mean_sd = mea
```

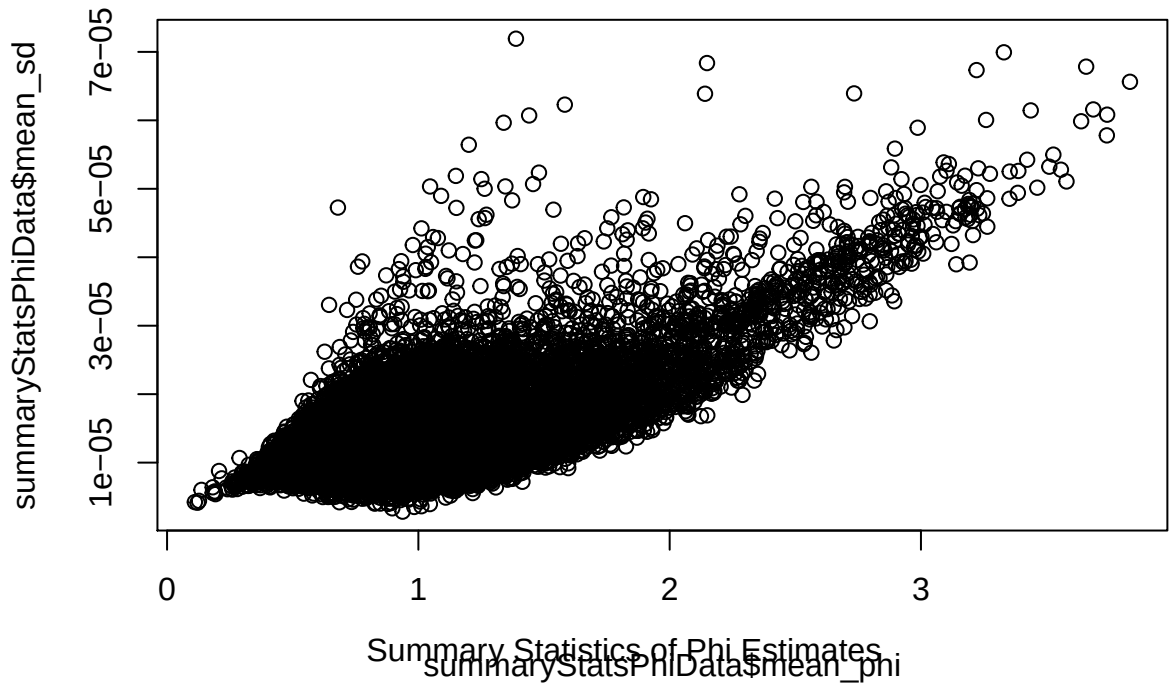
```
comment(summaryStatsPhiData) <- "summary stats for means and sd of phi for a geneName's multiple isofor
dim(summaryStatsPhiData)

write.csv(unfilteredPhiData, file="Output/labeled.unfiltered.phi.data.csv", quote=FALSE, row.names=FALS
write.csv(anomolousGeneInfo, "Output/anomolousGeneInfo.csv", quote=FALSE, row.names=FALSE)
write.csv(summaryStatsPhiData, file="Output/summaryStatsPhiData.csv", quote=FALSE, row.names=FALSE)
save(phiData, summaryStatsPhiData, seqData, file = "Output/processed.ROC.data.Rdata")
```

Otherwise import the data

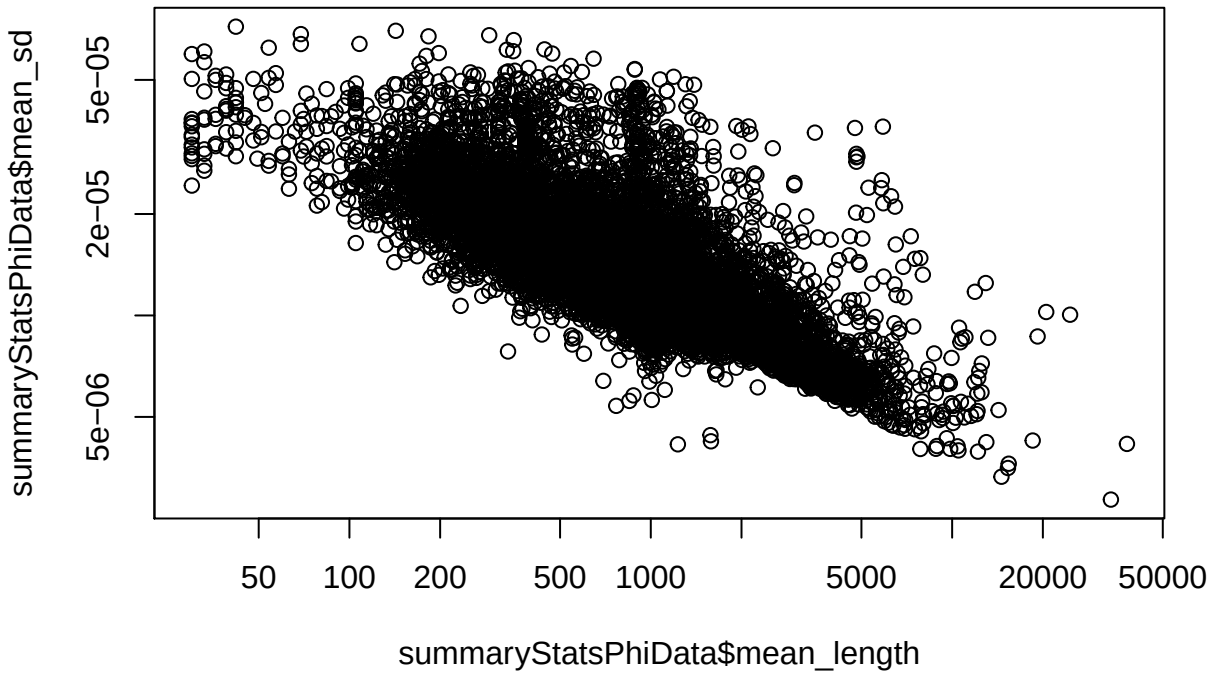
```
load("Output/processed.ROC.data.Rdata")
```

Plot Results

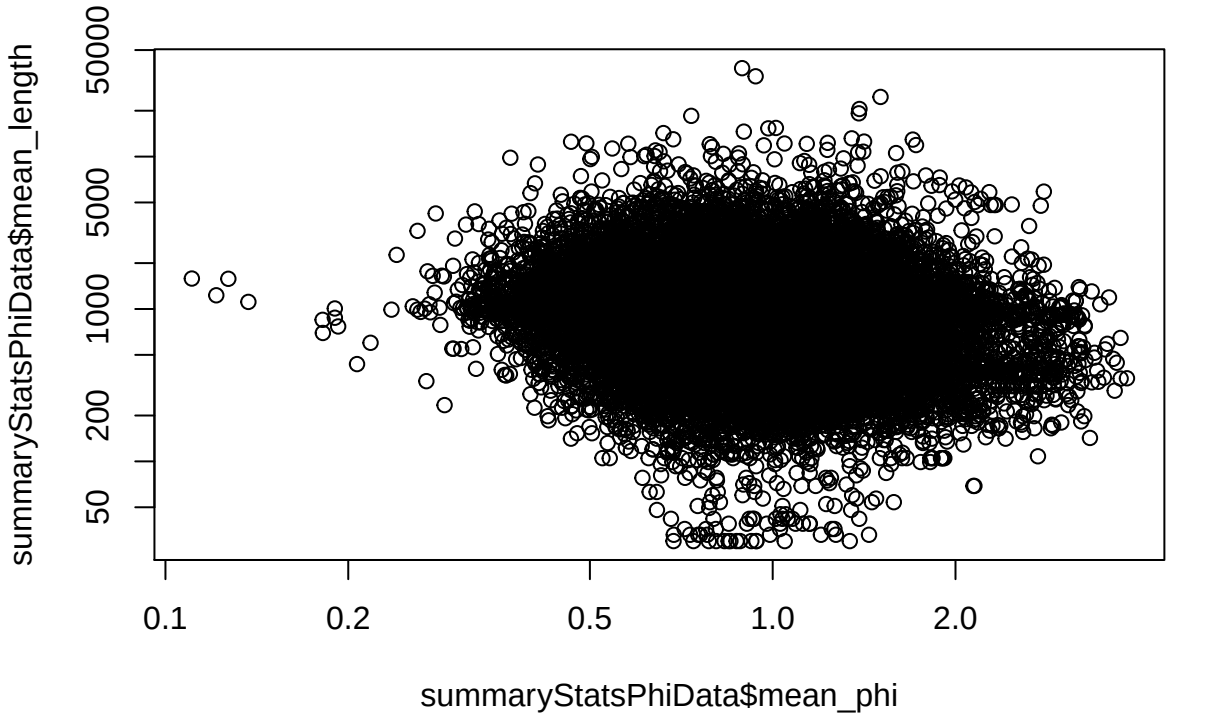
```
#par(mfrow=c(2,2))
plot(summaryStatsPhiData$mean_phi, summaryStatsPhiData$mean_sd)
mtext("Summary Statistics of Phi Estimates", side = 3, line = -21, outer = TRUE)
```



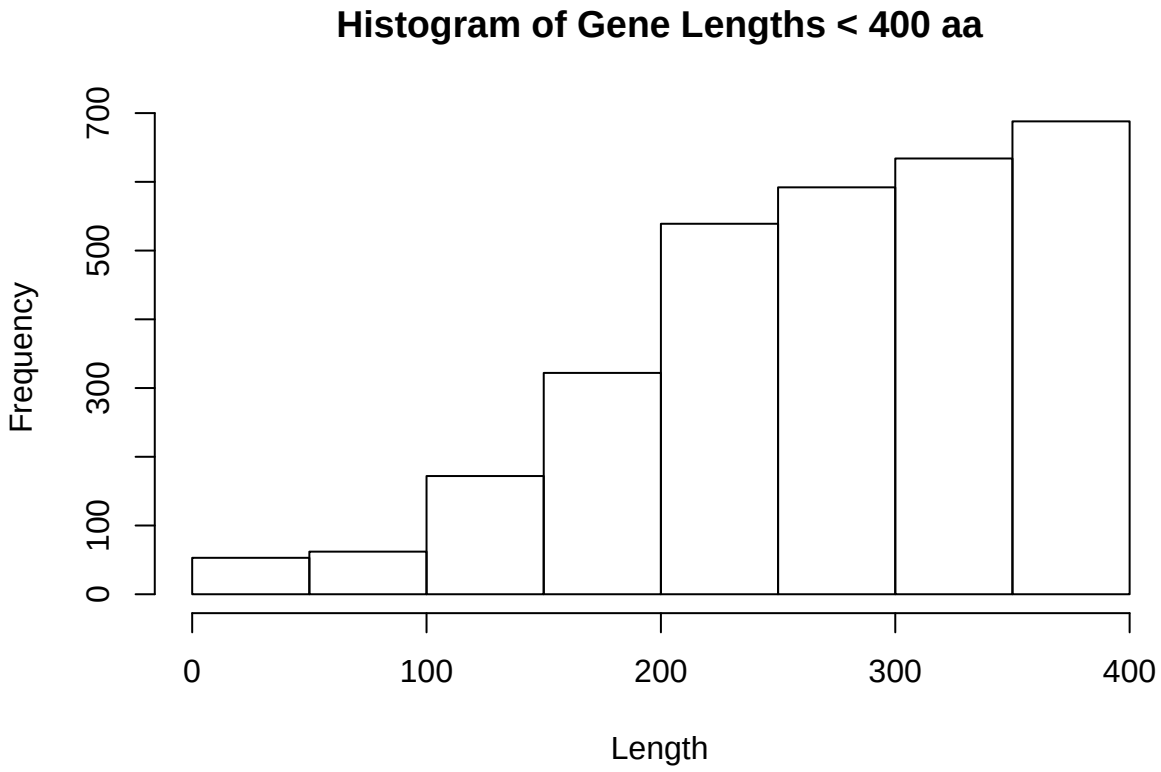

```
plot(summaryStatsPhiData$mean_length, summaryStatsPhiData$mean_sd, log="xy")
```



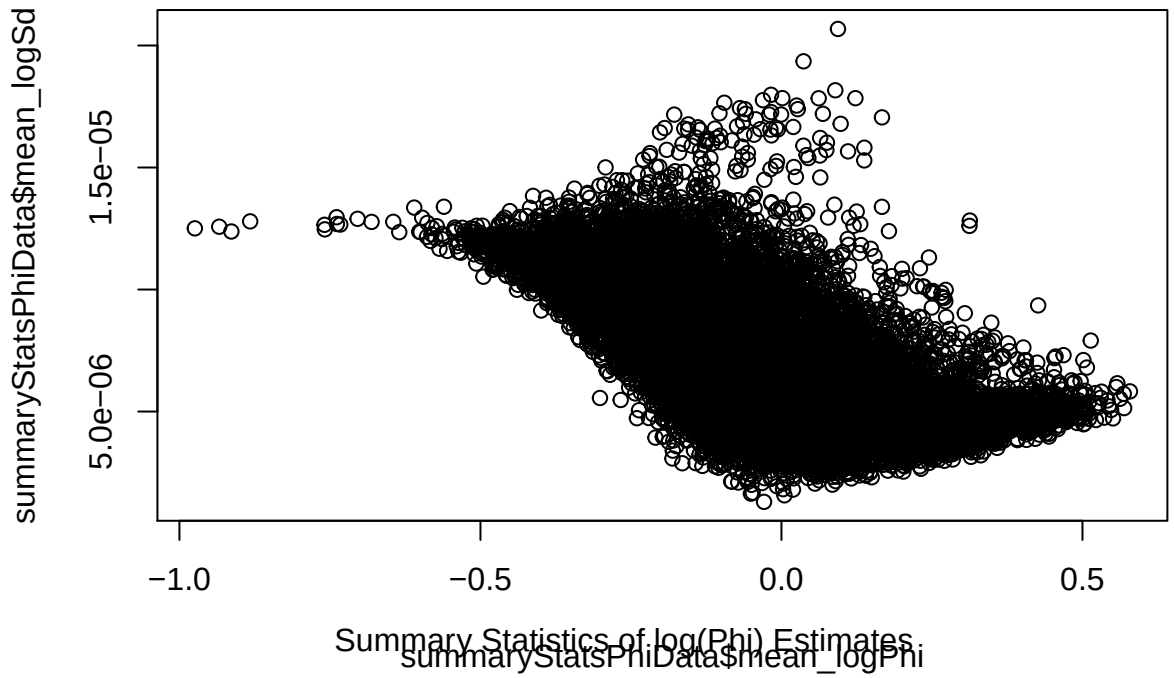

```
plot(summaryStatsPhiData$mean_phi, summaryStatsPhiData$mean_length, log="xy")
```



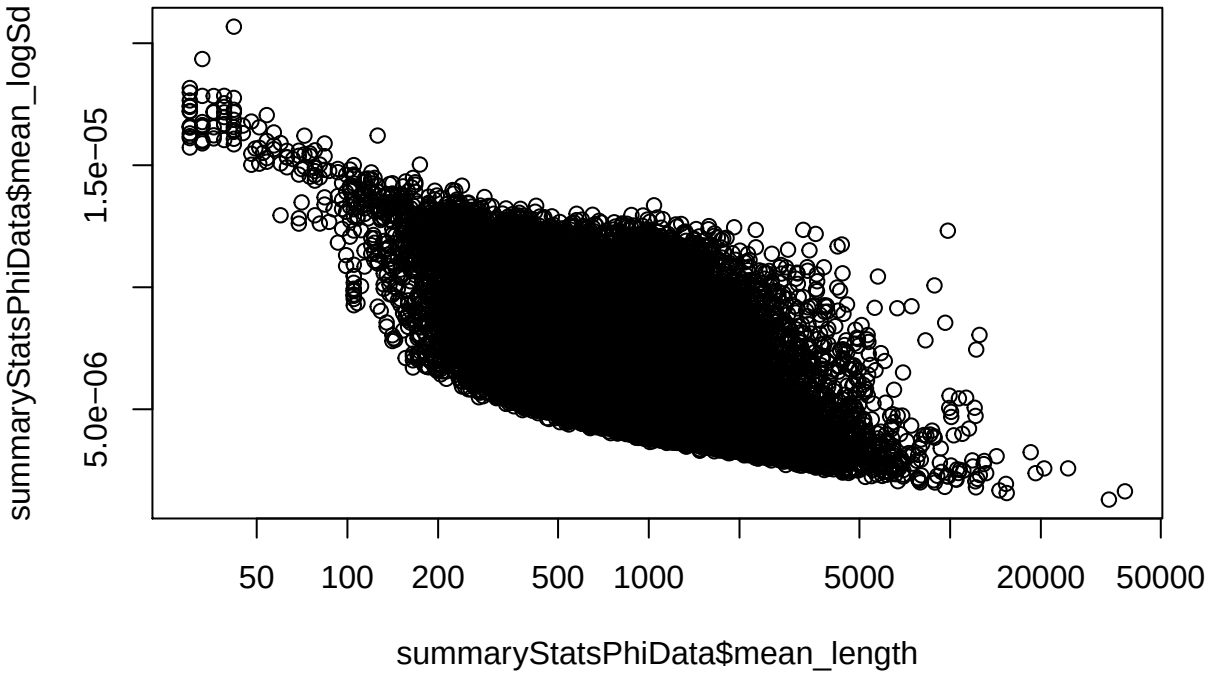

```
## Create histogram of short genes
tmp <- summaryStatsPhiData$mean_length[ summaryStatsPhiData$mean_length < 400]
hist(tmp, xlim=c(0, max(tmp)), main="Histogram of Gene Lengths < 400 aa", xlab="Length" )
```



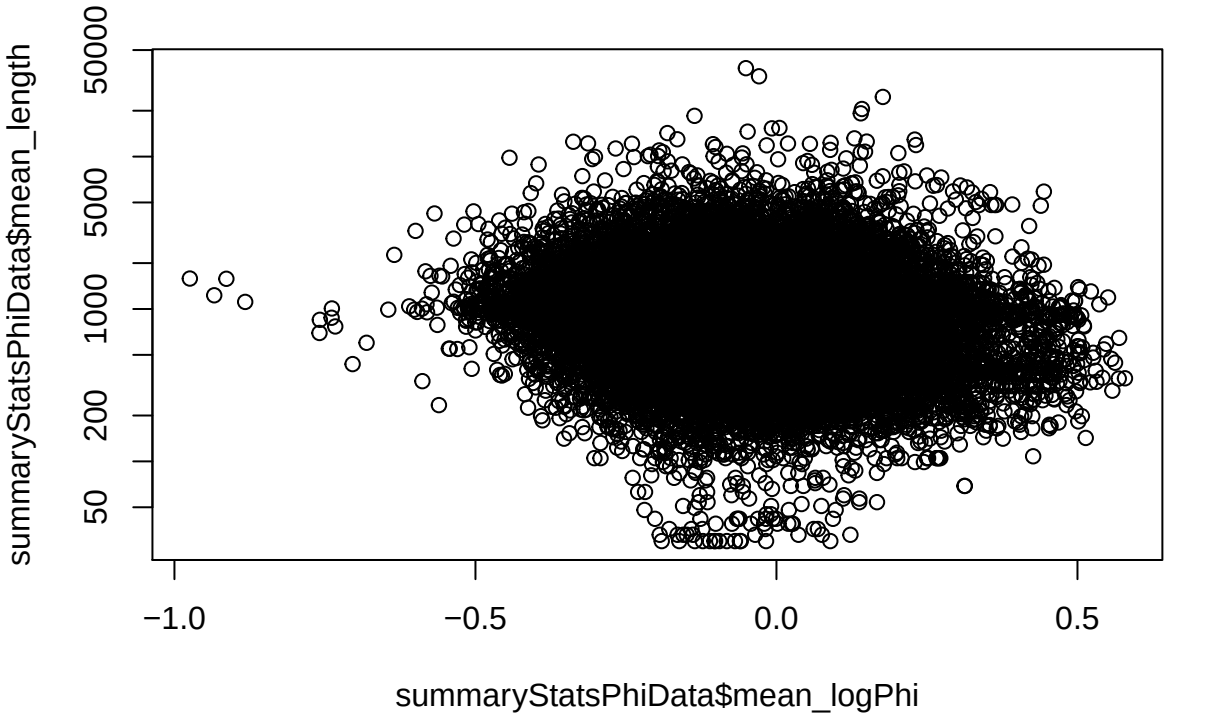

```
## log metrics
#par(mfrow=c(2,2))
plot(summaryStatsPhiData$mean_logPhi, summaryStatsPhiData$mean_logSd)
mtext("Summary Statistics of log(Phi) Estimates", side = 3, line = -21, outer = TRUE)
```



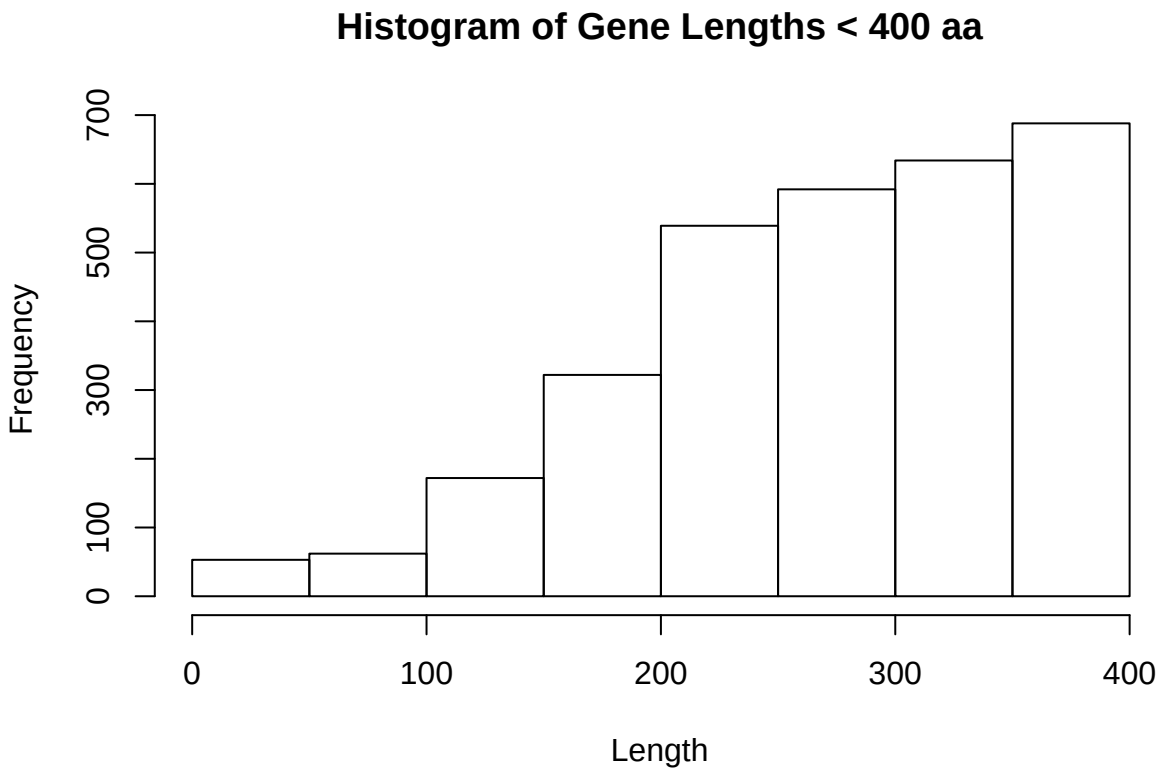

```
plot(summaryStatsPhiData$mean_length, summaryStatsPhiData$mean_logSd, log="x")
```




```
plot(summaryStatsPhiData$mean_logPhi, summaryStatsPhiData$mean_length, log="y")
```




```
## Create histogram of short genes
tmp <- summaryStatsPhiData$mean_length[ summaryStatsPhiData$mean_length < 400]
hist(tmp, xlim=c(0, max(tmp)), main="Histogram of Gene Lengths < 400 aa", xlab="Length" )
```



Import Phi Values Using Lu's File – BROKEN! NOT USED

- Don't use Lu's file

```
## NOTE: rocNames is corrupt.  It has replaced some gene names with date formats (e.g. apr-1 has been c
rocNamesBroken <- read_csv("Input/lu.phi.mean.by.names.csv", col_names = c("geneName","phi2") )
rocNamesBroken[rocNamesBroken$geneName=='1-Apr',]
## THere are 17 corrupted names
length(unfilteredSeqGeneNames) - sum(rocNamesBroken==unfilteredSeqGeneNames)

## Need to run 'Get estimates from ROC' code below to evaluate following line
phiBrokenDataCheck  <- bind_cols(rocNamesBroken, unfilteredPhiPosteriorInfo)
## Verify that phi values line up between the two datasets
plot(phiBrokenDataCheck$phi2, phiDataBrokenCheck$phi)
```

Examine Anomolous Genes

- Originally observed as cluster of genes with phi~1 and elevated SDs
- Upon examination we see they are actually genes with n mod 3 !=0.
- Don't need to evaluate this code any more as of 12 Jul 2020

```
length(anomolousGeneNames$geneName)
length(unique(anomolousGeneNames$geneName))
anomolousInfo <- filter(unfilteredFastaInfo, geneName %in% anomolousGeneNames$geneName)
regularInfo <- filter(unfilteredFastaInfo, !(geneName %in% anomolousGeneNames$geneName))
```

E-MTAB

Load Data

```
emtabFile <- "Input/E-MTAB-2812-query-results.tpms.tsv"
flatData  <- readr::read_tsv(emtabFile, skip=4) %>%
    dplyr::rename(WBID = `Gene ID`, geneName=`Gene Name` ) ## WBID is the WormBaseID
```

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

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

```
## use pivot_longer command (not gather which is depricated)
tmpData <- flatData %>% pivot_longer(-c(WBID, geneName), names_to = "long_description", values_to = "co


## Now separate entry in descriptor column into separate column entries and filter out tissue specific
tmpFastaGeneName <- unique(fastaInfo$geneName)
countData  <- separate(tmpData, long_description, into=c("sex", "tissue", "stage"), sep=", ") %>% filte
    filter(geneName %in% tmpFastaGeneName) %>%
    replace_na(list(count = 0)) ## replace NAs with 0.
## Before doing so the min value in the tibble was 0.1

comment(countData) <- paste0("Data from ", emtabFile, "in tidy format. Empty cells filled with '0', gen
```

Create stageCounts

- Data includes a new set of measurements for sex=‘hermaphrodite’ stage=‘embryo Ce’. This data is created from the mean values of all sex = ‘hermaphrodite’ and stage = ‘* embryo Ce’ data

```
## Get hermphrodite embryo stage names
## NB: male, 'embryo Ce' lacks a space at the start
embryoStageName  <- unique(countData$stage[grepl(' embryo Ce', countData$stage )])

separateEmbryoStageCount  <-  filter(countData, stage %in% embryoStageName)

##
## Make things simple and focus on just the 'main' lifestage measurements
## note that the stage "embryo Ce" is specific to sex = male
## Thus need to combine embryo measurements some how
##
## Only include data columns well find elsewhere
## See separate chunk for looking at moments of data
embryoStageCount <- separateEmbryoStageCount %>%
 group_by(WBID) %>%
    summarise(geneName, sex, tissue, stage="embryo Ce", count =mean(count), .groups="drop") %>%
    unique()
```

```
## embryoStageCount <- select(tmpEmbryoStageCount, -c(count, sd, tissue) ) %>% group_by()

nonEmbryoStageCount <- filter(countData, !(stage %in% embryoStageName)) %>% select(-tissue)

stageCount <- bind_rows(embryoStageCount, nonEmbryoStageCount)
comment(stageCount) <-  paste0("Data from ", emtabFile, "in tidy format. Empty cells filled with '0'\n"
```

Plot Hermaphrodite Embryo Data

- Embryonic data from various development points.
- Use it this data to
  - Create a 'embryo Ce' stage for the analysis. Done using the mean across these various stages.
  - Examine how the variation in the measurements increases with its value. (Figure **??**)
  - It clearly increase linearly on a log scale. Increasing the order of the polynomial helps some.
  - Model error seems a bit off at higher values. Perhaps it's the error in the x variable tempering things?
  - Note 832 points are excluded due to 0 values.

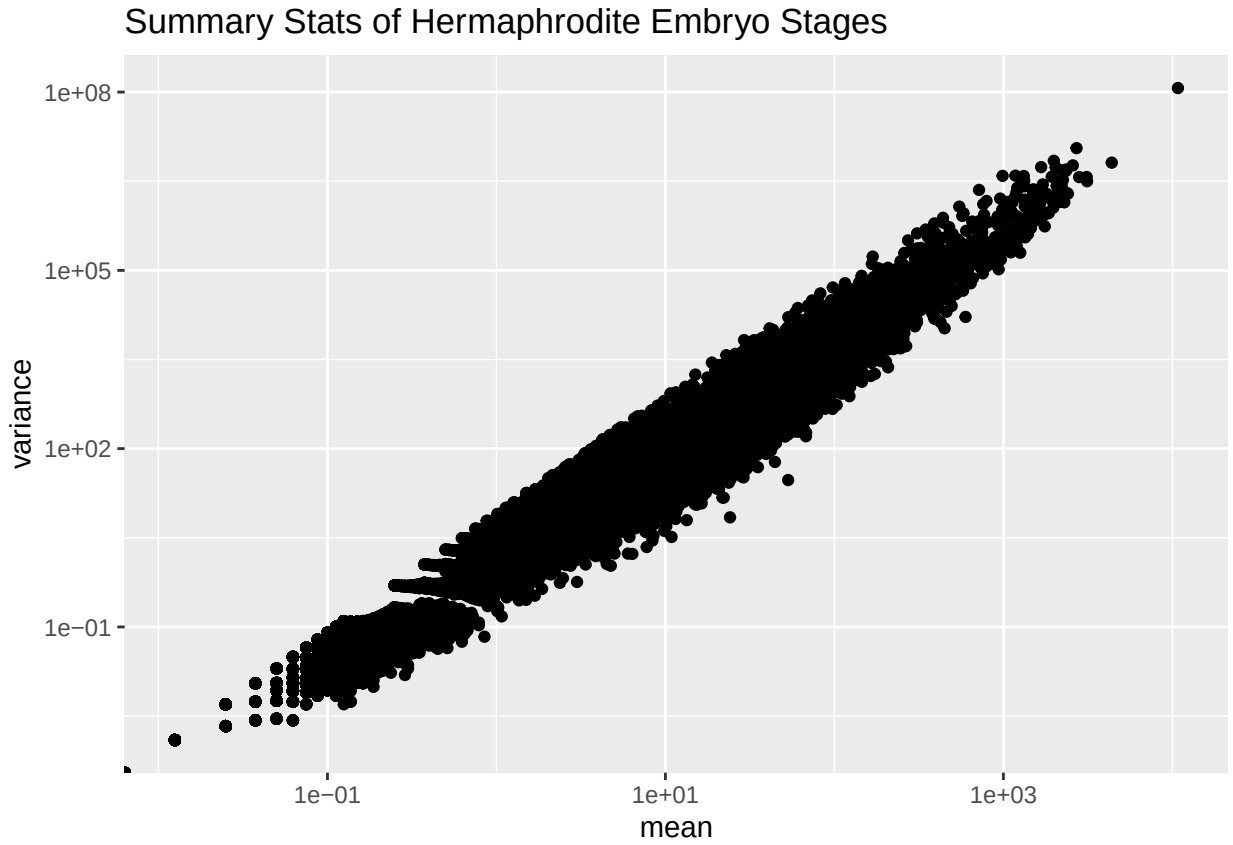
```
## Be sure to use count = func(count) last in summarise().
## Otherwise it will redefine count to a single value which will screw up the other functions
## Use mean() rather than median for summarizing count
## It seems less sensitive to 0 values
embryoStageCountMoments <- separateEmbryoStageCount %>%
    group_by(WBID) %>%
    summarise(mean=mean(count), inverseMean = 1/mean(count), median=median(count), sd=sd(count), logMea

embryoStageCountMoments
```

```
## # A tibble: 20,386 x 11
##    WBID     mean inverseMean median      sd logMean  logSd logVar     n
##    <chr>   <dbl>       <dbl>  <dbl>   <dbl>   <dbl>  <dbl>  <dbl> <int>
##  1 WBGe~ 2.85e+1     0.0351    29.5 1.05e+1   3.35   2.35    4.70     8
##  2 WBGe~ 4.14e+1     0.0242    24.5 4.98e+1   3.72   3.91    7.82     8
##  3 WBGe~ 3.26e+1     0.0307    24   3.07e+1   3.49   3.43    6.85     8
##  4 WBGe~ 1.57e+2     0.00637  108.  1.43e+2   5.06   4.96    9.92     8
##  5 WBGe~ 3.04e+0     0.329      3   2.56e+0   1.11   0.941   1.88     8
##  6 WBGe~ 6.72e+0     0.149      1.5 1.08e+1   1.91   2.38    4.76     8
##  7 WBGe~ 1.11e+0     0.899      0.2 1.86e+0   0.107  0.622   1.24     8
##  8 WBGe~ 1.25e-2    80          0   3.54e-2  -4.38  -3.34   -6.68     8
##  9 WBGe~ 1.55e+0     0.645      0.3 2.46e+0   0.438  0.900   1.80     8
## 10 WBGe~ 2.26e+1     0.0442    18.5 1.30e+1   3.12   2.57    5.13     8
## # ... with 20,376 more rows, and 2 more variables: `min(count)` <dbl>,
## #   `max(count)` <dbl>
```

```
embryoStageCountMoments %>%
    #filter(mean > 1) %>%
    ggplot() + geom_point(aes(mean, sd^2)) +
    scale_x_log10() +
    scale_y_log10() +
    labs(title="Summary Stats of Hermaphrodite Embryo Stages",
         x="mean", y="variance")
```

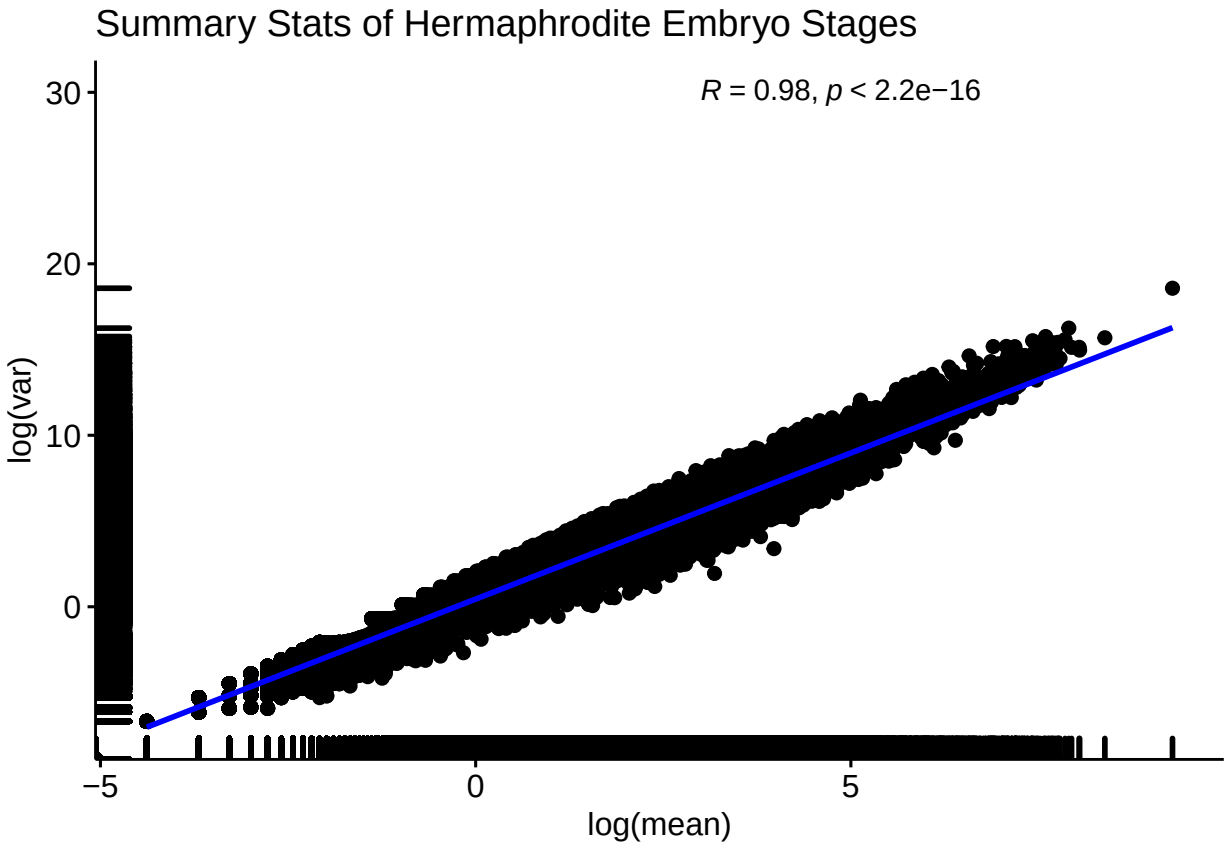
```
## Warning: Transformation introduced infinite values in continuous x-axis
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```




```
## Make a scatter plot fitting a linear regression using ggpubr routines
 ggscatter(embryoStageCountMoments, x = "logMean", y = "logVar",
           add = "reg.line", ## Add regression line
           conf.int = TRUE,   ## Add confidence interval
           add.params = list(color = "blue",
                             fill = "lightgray"),
           rug = TRUE,  ## add marginal
           title="Summary Stats of Hermaphrodite Embryo Stages",
           xlab="log(mean)", ylab="log(var)"
 )+
     stat_cor(method = "pearson", label.x = 3, label.y = 30)  ## Add correlation coefficient
```

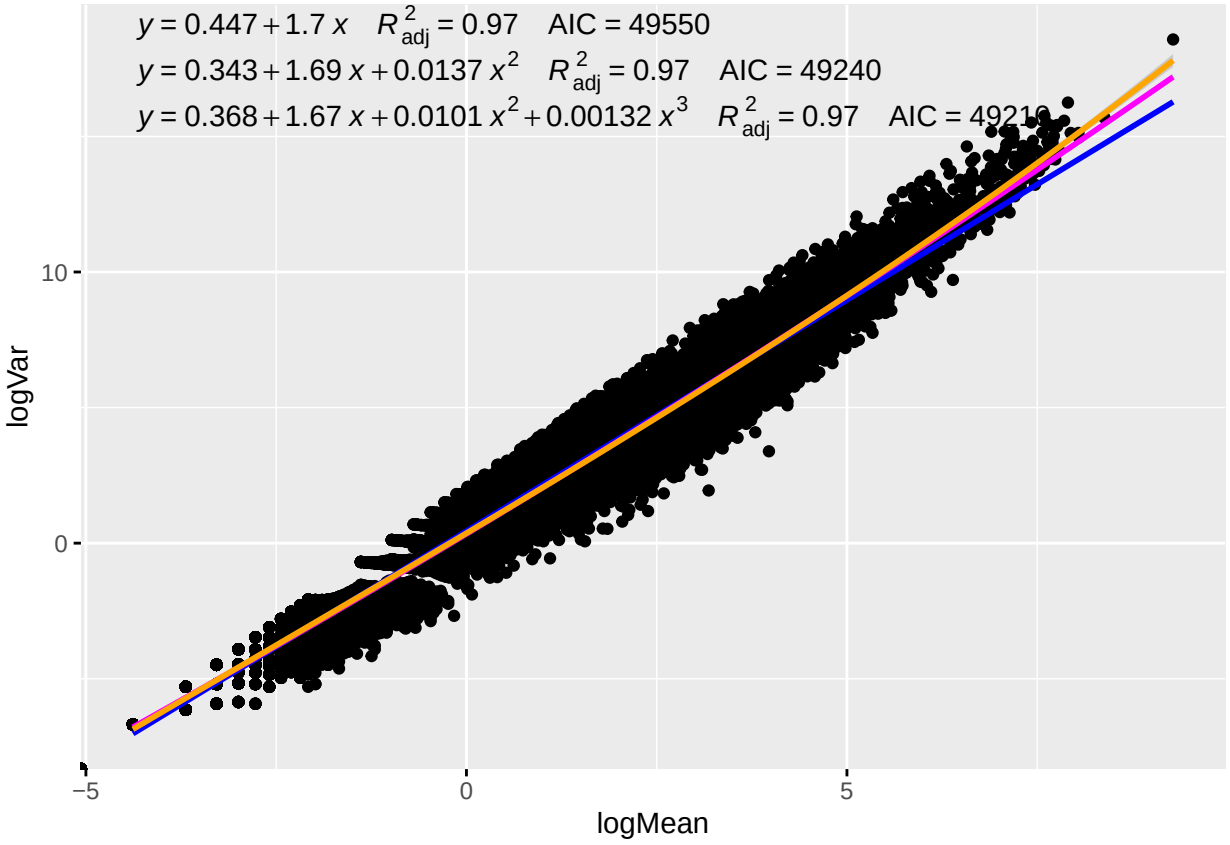
```
## `geom_smooth()` using formula 'y ~ x'
## Warning: Removed 823 rows containing non-finite values (stat_smooth).
## Warning: Removed 823 rows containing non-finite values (stat_cor).
```




```
## From http://www.sthda.com/english/articles/32-r-graphics-essentials/131-plot-two-continuous-variables
## Polynomial regression. Sow equation and adjusted R2

## Try higher order functions
formula1 <- y ~ poly(x, 1, raw = TRUE)
formula2 <- y ~ poly(x, 2, raw = TRUE)
formula3 <- y ~ poly(x, 3, raw = TRUE)

myColors = c("blue", "magenta", "orange", "green")


p <- ggplot(embryoStageCountMoments,
            aes(logMean, logVar)) +  ## could add `color = group
    geom_point() +
    geom_smooth(method = "lm", formula = formula1, color = myColors[1]) +
    stat_poly_eq(
    aes(label =  paste(..eq.label.., ..adj.rr.label.., ..AIC.label.., sep = "~~~~")),
    formula = formula1, label.y= 1, parse = TRUE
    )+
    geom_smooth(method = "lm", formula = formula2, color = myColors[2]) +
    stat_poly_eq(
    aes(label =  paste(..eq.label.., ..adj.rr.label.., ..AIC.label.., sep = "~~~~")),
    formula = formula2, label.y = 0.94, parse = TRUE
    )+
    geom_smooth(method = "lm", formula = formula3, color = myColors[3]) +
    stat_poly_eq(
    aes(label =  paste(..eq.label.., ..adj.rr.label.., ..AIC.label.., sep = "~~~~")),
```

```
    formula = formula3, label.y = 0.88, parse = TRUE
    )

ggpar(p, palette = "jco")
```

```
## Warning: Removed 823 rows containing non-finite values (stat_smooth).
## Warning: Removed 823 rows containing non-finite values (stat_poly_eq).
## Warning: Removed 823 rows containing non-finite values (stat_smooth).
## Warning: Removed 823 rows containing non-finite values (stat_poly_eq).
## Warning: Removed 823 rows containing non-finite values (stat_smooth).
## Warning: Removed 823 rows containing non-finite values (stat_poly_eq).
```



Broken functions.

```
## Try to generalize and fail
myFormula=list(
    y ~ poly(x, 1, raw = TRUE),
    y ~ poly(x, 2, raw = TRUE),
    y ~ poly(x, 3, raw = TRUE)
)



p <- ggplot(embryoStageCountMoments,
```

```
            aes(logMean, logVar)) +  ## could add `color = group
    geom_point() +
    geom_smooth(method = "lm", formula = myFormula, color = myColors) +
    stat_poly_eq(
    aes(label =  paste(..eq.label.., ..adj.rr.label.., sep = "~~~~")),
    formula = myFormula, parse = TRUE
    )

ggpar(p, palette = "jco")

p <- ggplot(embryoStageCountMoments,
            aes(logMean, logVar)) +  ## could add `color = group
    geom_point() +
    geom_smooth(method = "lm", formula = myFormula[1], color = myColors[1]) +
    stat_poly_eq(
    aes(label =  paste(..eq.label.., ..adj.rr.label.., sep = "~~~~")),
    formula = myFormula[1], parse = TRUE
    ) +
    geom_smooth(method = "lm", formula = myFormula[2], color = myColors[2]) +
    stat_poly_eq(
    formula = myFormula[2], parse = TRUE
    )


ggpar(p, palette = "jco")

print("Data suggests log(var) = c log(mean) across biological replicates")
```

Create Counts for mainLifeStages

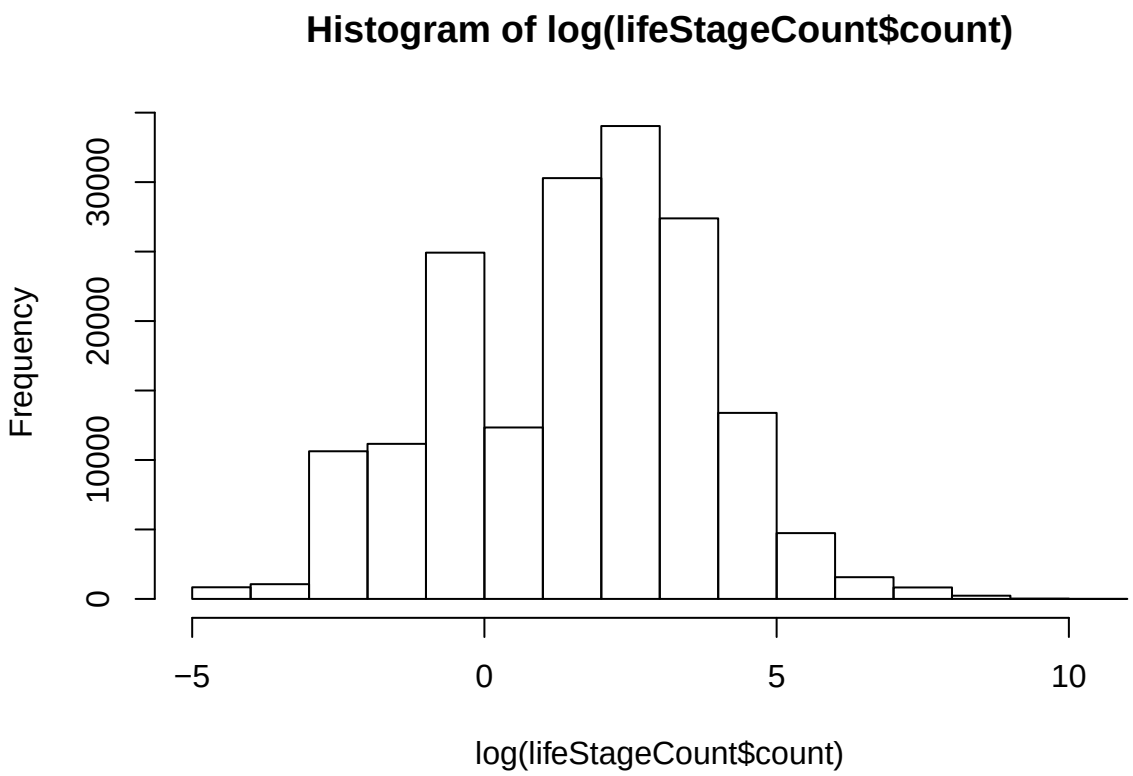
```
## This will include both sex = c('male', 'hermaphrodite')\
## Really only lacks 'newly molted young adult hermaphrodite Ce'
lifeStages <- c("embryo Ce", "L1 larva Ce", "L2 larva Ce", "L3 larva Ce", "L4 larva Ce", "adult Ce", "da
comment(lifeStages)=paste0("List of main lifestages to use in regression.\n", creationInfo)

## Note that 'count' are standardized means of biological and technical replicates
## 'organism' scale data for different lifestages
## Includes data for 'sex' =  male and hermphrodite
lifeStageCount <- filter(stageCount, (stage %in% lifeStages))
comment(lifeStageCount) <- c("RNASeq counts for each of the\n\t- hermaphrodite non-embryo lifestages: Li


lifeStageCountMoments <- lifeStageCount %>% group_by(sex, stage)%>% summarize(stage_sd = sd(count), stag
```

```
## `summarise()` regrouping output by 'sex' (override with `.groups` argument)
```

```
summary(lifeStageCount$count)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     0.00     0.30     4.00    36.94    19.00 36316.00
```

```
## hist(lifeStageCount$count)
hist(log(lifeStageCount$count))
```




```
geneNameAndWormBaseID <- unique(select(lifeStageCount, c(geneName, WBID)))
comment(geneNameAndWormBaseID) <- paste("Mapping between WormBase IDs and geneName in FASTA file")

## Export values
## Don't use write_csv because that only works with dataframes,not tibbles
## write.csv converts the tibble to a dataframe
```

Create database linking geneName and WormBaseID

Evaluate these lines manually outside of knitr

```
save(lifeStageCount, lifeStages, file = "Output/processed.E-MTAB.data.Rdata")
## Export data using 'geneName' name or WormBaseID
write.csv(geneNameAndWormBaseID, file="Output/geneName.and.WBID.csv", quote=FALSE, row.names=FALSE)
```

Combine empirical E-MTAB and ROC data

- This used to work, but doesn't now. Not worth fixing right now.

```
## Combine phi

phiNames <- summaryStatsPhiData$geneName

namesToWBID <- lapply(phiNames, function(x) {index = which(geneNameAndWormBaseID$geneName == x); ifelse


unmatchedPhiNames <- phiNames[is.na(namesToWBID)]
geneName <- geneNameAndWormBaseID$geneName
```

```
WBID <- geneNameAndWormBaseID$WBID




tidyJoinedData <- bind_rows(lifeStageCount, tidyFilteredAndWeightedPhiData) %>%
    mutate_if(is.character, as.factor)

joinedData <- pivot_wider(tidyJoinedData, names_from=stage, values_from=c(count, weight))
```