Estimate logSD using censored data approaches

Michael A. Gilchrist

21 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.

Evaluate Chunks

M-n v * - v polymode-eval-region-or-chunk - b polymode-eval-buffer - u or ↑ polymode-eval-buffer-from-beg-to-point - d or ↓ polymode-eval-buffer-from-point-to-end

M-n e : Evaluate buffer # Load Libraries

```
library(tidyr)
library(tibble)
library(readr)
library(dplyr)
library(stringr)
library(forcats)
library(ggplot2)
library(optimx)

exportData=TRUE
```

Estimate SD of log(counts) from E-MTAB Data taking into account data censoring

- The lowest count value is 0.1, thus treat all 0 counts (previous empty cells) as coming from a censored distribution at mRNA < 0.1 counts.
- An example of such an approach can be found here.
- An alternative distribution is the 'Zero Modified Log-Normal' which assumes a mixture distribution which is a LogN with an additional probability mass at 0.
  - Zero Modified Log-Normal (ZMLN) is included in the EnvStats package.
  - This package generates CI for the mean, but not the SD. Could bootstrap data to generate CI for the SD
- "One way to try to assess whether a zero-modified lognormal (delta), zero-modified normal, censored normal, or censored lognormal is the best model for the data is to construct both censored and detects-only probability plots (see qqPlotCensored)" - EnvStats page above
- Given that ROC does not have a 0 category, we don't want the ZMLN model.
- Thisblog provides as workflow that we can use w/o any additional packages

Load Data

- Original data file is `E-MTAB-2812-query-results.tpms.tsv`
- Data processed in `../21_Process.Published.Means`

```
load("Input/processed.E-MTAB.data.Rdata")

# get some basic info on what we've loaded
comment(lifeStageCount)
```

```
## [1] "RNASeq counts for each of the\n\t- hermaphrodite non-embryo lifestages: L1 larvae Ce, L2 larvae
## [2] "\tDate: Tue Jul 21 09:57:07 2020\n\tLocation: AcrossTissue/MikesWork/2020/07/20_Process.Publish
```

```
comment(lifeStages)
```

```
## [1] "List of main lifestages to use in regression.\n\tDate: Tue Jul 21 09:57:07 2020\n\tLocation: Ac
```

```
names(lifeStageCount)
```

```
## [1] "WBID"     "geneName" "sex"      "tissue"   "stage"    "count"
```

Analyze Data

- Using code from here
- Note 0 values in dataset were added by myself when processing data. Originally these were blank cells.

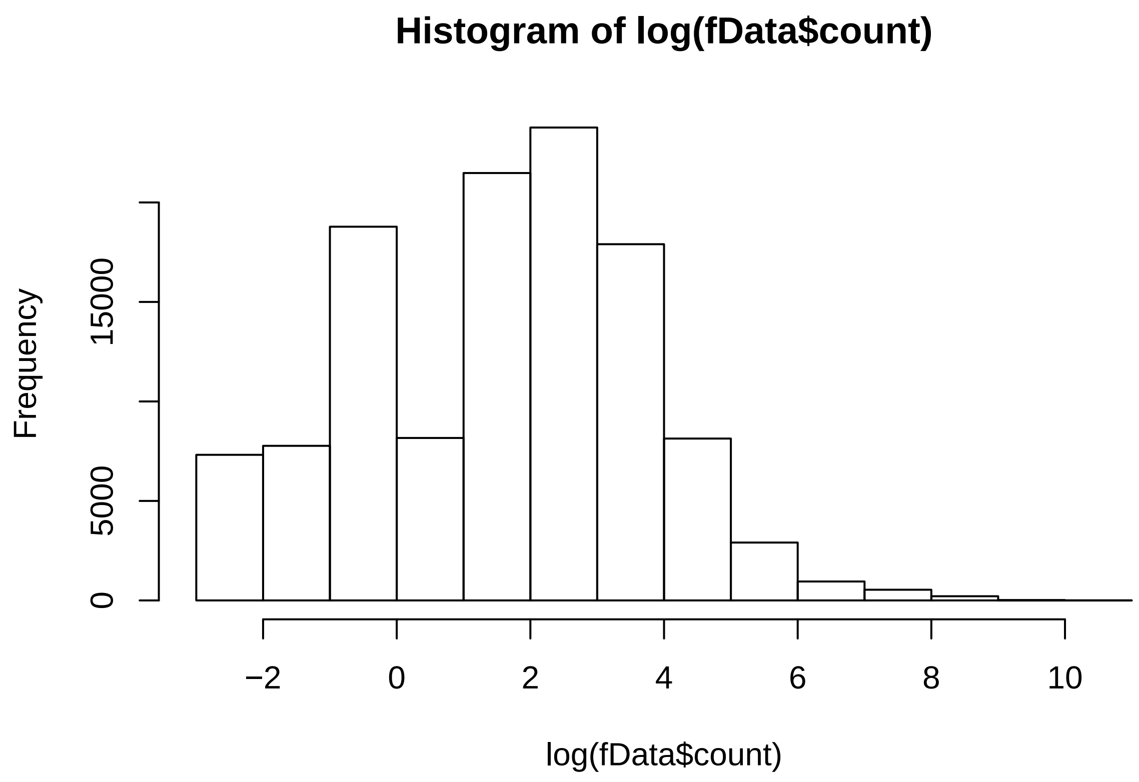
```
## Find lower limits for each stage.
## Exclude the embryo stage because it has a different threshold and was generated using a different pi

#filter data
fData <- lifeStageCount %>% filter(sex=='hermaphrodite' & !(stage=="embryo Ce"))%>%
    select(-c(sex, tissue))%>%
    mutate(count=ifelse(count==0, NA, count))
# get lower limit (which should be 0.1)


#lifeStageCountCensored
lscc <-
    fData %>% group_by(stage) %>%
    mutate(threshold = min(count, na.rm=TRUE)) %>%  ## might want to try using threshold set to 1
    group_by() %>%
    mutate(count = ifelse(count < threshold, threshold, count)) #, threshold=NULL)

save(fData, lscc, file="filtered.and.censored.data.Rdata")
```

Else load data

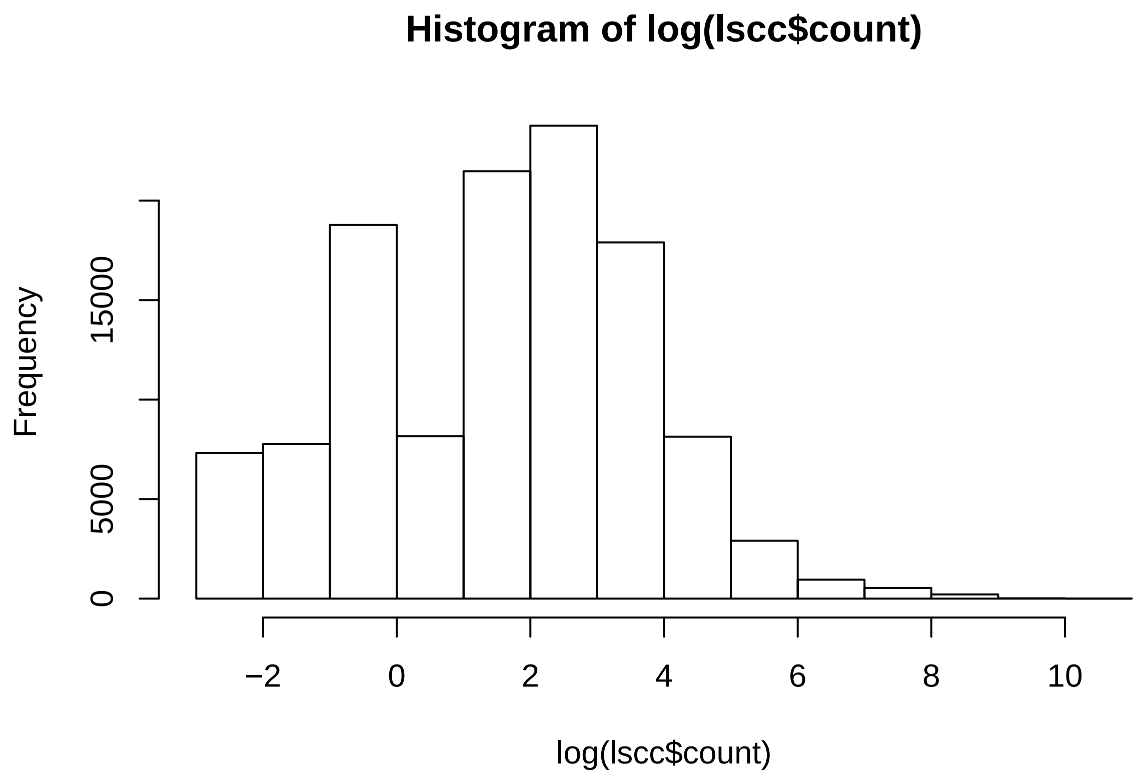
```
load(file="filtered.and.censored.data.Rdata")
```

Plot Data

```
hist(log(fData$count))
```



```
hist(log(lscc$count))
```

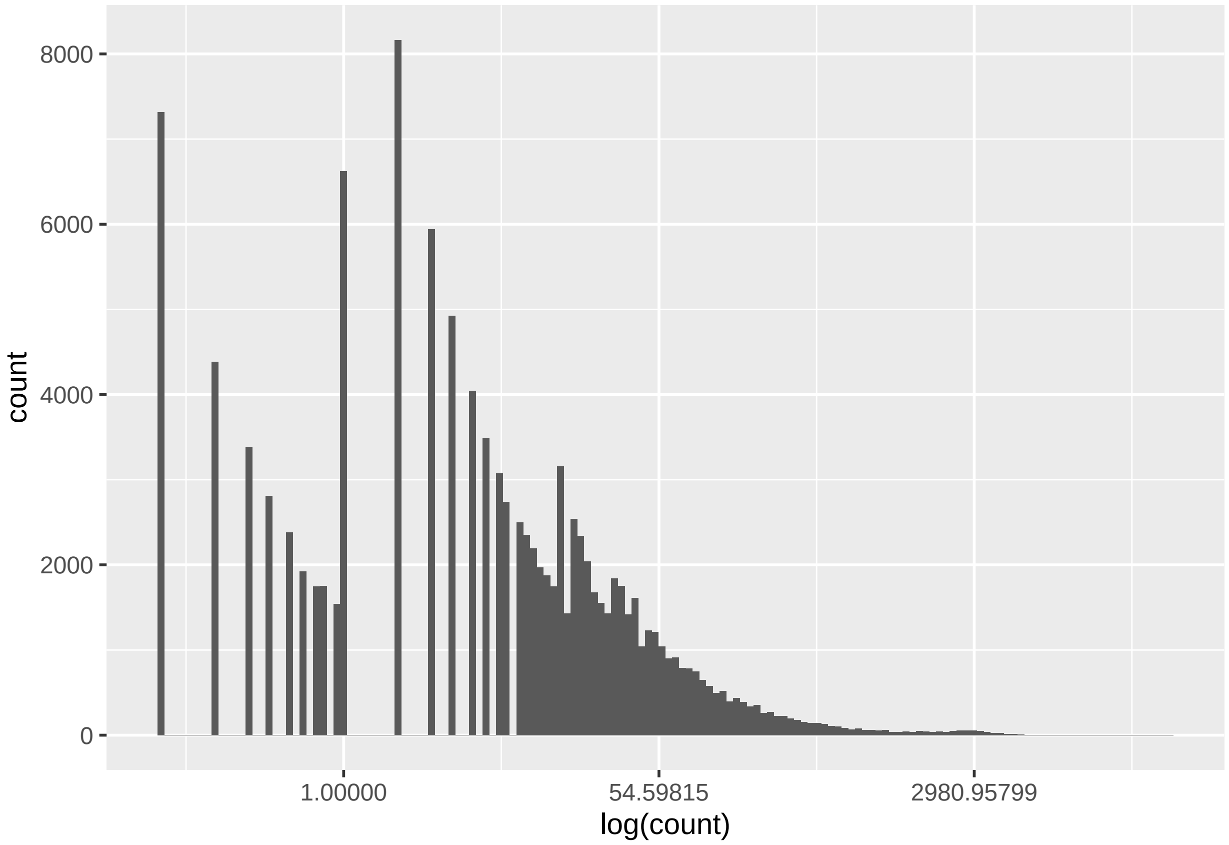


```
mean(fData$count, na.rm=TRUE)
```

```
## [1] 43.4626
```

```r
ggplot(lscc, aes(x = count)) +
  geom_histogram(bins = 150) +
  scale_x_continuous(trans = "log") +
  xlab("log(count)")
```

```
## Warning: Removed 24766 rows containing non-finite values (stat_bin).
```



## Estimate Parameters - `logLikCensoredFun` taken from r-bloggers - Note that `maxLik` - 'must have the parameter vector as the first argument' - can return a single LLik value or a nmeric vector where each component is a LLik value - Is a headache so **don't** use it. Use `optimx` instead

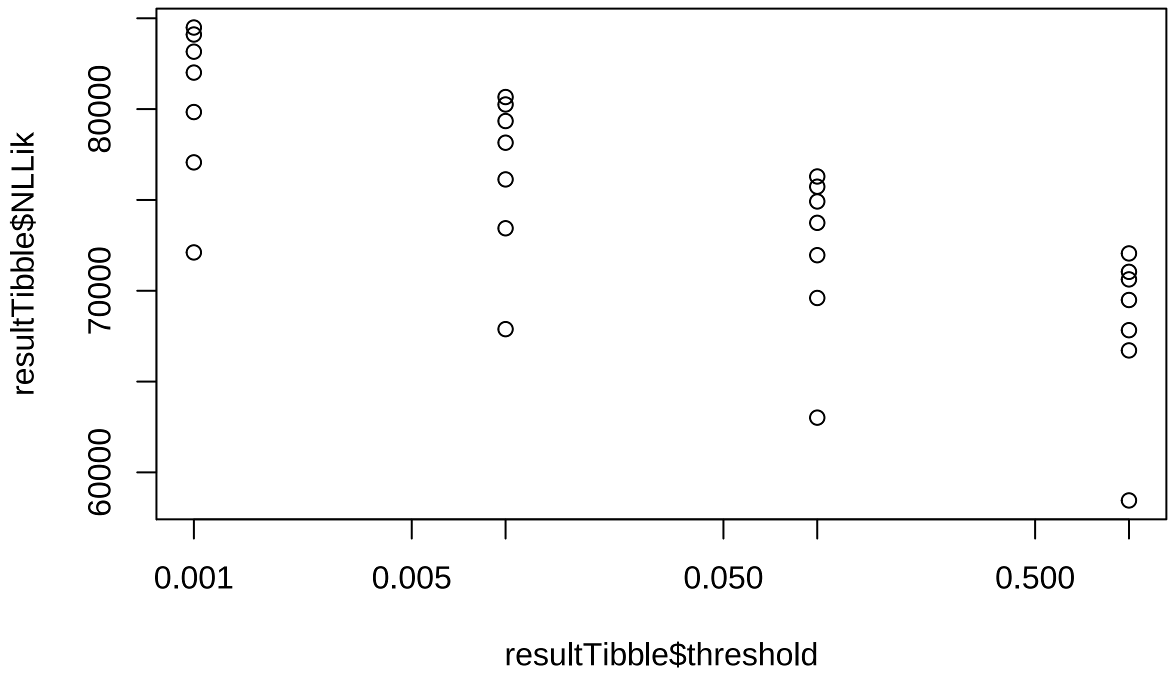
```r
## Create toy dataset
## tmpData  <- lscc %>% filter(stage=="adult Ce") %>% slice_sample(n=10000)

### Define objective function: -LogLik
NLLikCensoredFun <- function(par, count, threshold, print=FALSE){
    meanlog = par["meanlog"]
    sdlog = par["sdlog"]
    #meanlog=par[1]
    #sdlog=par[2]
   if(sdlog < 0) return(NA)
    cdf <-
        plnorm(threshold, meanlog = meanlog, sdlog = sdlog, log.p = TRUE);


    llik <-
        sum(
            ifelse(
                 is.na(count) | count < threshold, #Allow use of either criteria
                 cdf,
                 dlnorm(count, meanlog = meanlog, sdlog = sdlog, log = TRUE))
         )
    if(print) print(paste0("LLik = ", llik, ", meanlog = ", meanlog, ", sdlog = ", sdlog))
    return(-llik)
}


## THIS WORKS!!!
## BUt I'd like to use pipes...
optim(par=c(meanlog=1, sdlog=0.01),
      fn=NLLikCensoredFun, ## function to optimize
      lower=c(-Inf, 1E-10),
      upper=c(1E5, 100),
      method = "L-BFGS-B",
      count=lscc$count,
      threshold=lscc$threshold, ## additional arguments for the function
      print=FALSE
      )
```

```
## $par
##   meanlog     sdlog
## 0.6742661 2.8204880
##
## $value
## [1] 506004.5
##
## $counts
## function gradient
##       27       27
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

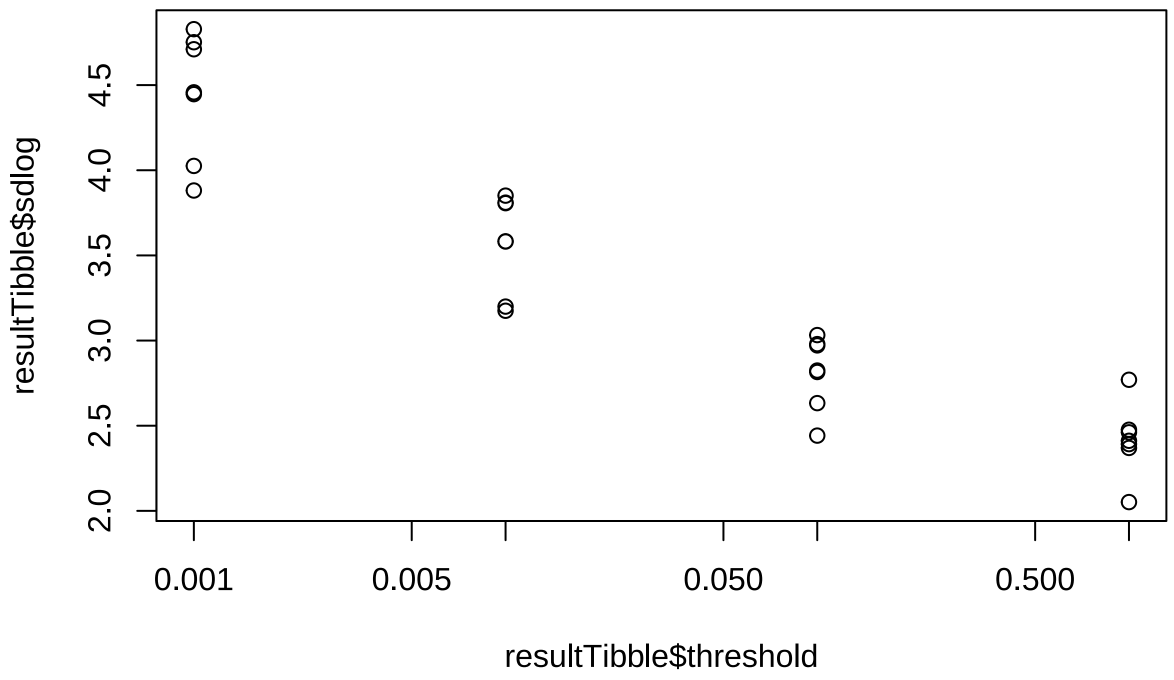
```r
      ##count=pull(tmpData, count), threshold=1, ## additional arguments for the function

## THis works two, but what a PITA
resultTibble <- tibble()
for(threshold in c(0.001, 0.01, 0.1, 1)){
    for(lifestage in lifeStages[-1]){
        ##print(lifestage)
        tmp <- lscc %>% filter(stage==lifestage)
        count <- tmp$count
        fit <-
            optim(
                par=c(meanlog=1, sdlog=2),
                fn=NLLikCensoredFun, ## function to optimize
                lower=c(-Inf, 1E-10), ## c(lower bound, initial value)
                upper=c(1E5, 100), ## c(upper bound, initial value)
                method = "L-BFGS-B",
                count= count,
                threshold=threshold,
                print=FALSE
            )
        results <- tibble(threshold, lifestage, count=length(count), NLLik=fit$value, meanlog=fit$par["
        resultTibble <- bind_rows(resultTibble, results)
        print(results, digits=4)
    }
}
```

```
## # A tibble: 1 x 6
##   threshold lifestage   count  NLLik meanlog sdlog
##       <dbl> <chr>       <int>  <dbl>   <dbl> <dbl>
## 1     0.001 L1 larva Ce 20386 77067.  0.0980  3.88
## # A tibble: 1 x 6
##   threshold lifestage   count  NLLik meanlog sdlog
##       <dbl> <chr>       <int>  <dbl>   <dbl> <dbl>
## 1     0.001 L2 larva Ce 20386 83162.  -0.307  4.75
## # A tibble: 1 x 6
##   threshold lifestage   count  NLLik meanlog sdlog
##       <dbl> <chr>       <int>  <dbl>   <dbl> <dbl>
## 1     0.001 L3 larva Ce 20386 84492.  0.0233  4.45
## # A tibble: 1 x 6
##   threshold lifestage   count  NLLik meanlog sdlog
##       <dbl> <chr>       <int>  <dbl>   <dbl> <dbl>
## 1     0.001 L4 larva Ce 20386 72109.  -0.433  4.03
## # A tibble: 1 x 6
##   threshold lifestage count  NLLik meanlog sdlog
##       <dbl> <chr>     <int>  <dbl>   <dbl> <dbl>
## 1     0.001 adult Ce  20386 79838.  -0.430  4.71
## # A tibble: 1 x 6
##   threshold lifestage      count  NLLik meanlog sdlog
##       <dbl> <chr>          <int>  <dbl>   <dbl> <dbl>
## 1     0.001 dauer larva Ce 20386 82011.  -0.137  4.46
## # A tibble: 1 x 6
##   threshold lifestage           count  NLLik meanlog sdlog
##       <dbl> <chr>               <int>  <dbl>   <dbl> <dbl>
## 1     0.001 post dauer stage Ce 20386 84107.  -0.329  4.83
## # A tibble: 1 x 6
##   threshold lifestage   count  NLLik meanlog sdlog
##       <dbl> <chr>       <int>  <dbl>   <dbl> <dbl>
## 1      0.01 L1 larva Ce 20386 73440.   0.417  3.20
## # A tibble: 1 x 6
##   threshold lifestage   count  NLLik meanlog sdlog
##       <dbl> <chr>       <int>  <dbl>   <dbl> <dbl>
## 1      0.01 L2 larva Ce 20386 79344.   0.204  3.81
## # A tibble: 1 x 6
##   threshold lifestage   count  NLLik meanlog sdlog
##       <dbl> <chr>       <int>  <dbl>   <dbl> <dbl>
## 1      0.01 L3 larva Ce 20386 80665.   0.461  3.58
## # A tibble: 1 x 6
##   threshold lifestage   count  NLLik meanlog sdlog
##       <dbl> <chr>       <int>  <dbl>   <dbl> <dbl>
## 1      0.01 L4 larva Ce 20386 67886. -0.0102  3.18
## # A tibble: 1 x 6
##   threshold lifestage count  NLLik meanlog sdlog
##       <dbl> <chr>     <int>  <dbl>   <dbl> <dbl>
## 1      0.01 adult Ce  20386 76130.  0.0639  3.81
## # A tibble: 1 x 6
##   threshold lifestage      count  NLLik meanlog sdlog
##       <dbl> <chr>          <int>  <dbl>   <dbl> <dbl>
## 1      0.01 dauer larva Ce 20386 78151.   0.314  3.58
## # A tibble: 1 x 6
##   threshold lifestage           count  NLLik meanlog sdlog
##       <dbl> <chr>               <int>  <dbl>   <dbl> <dbl>
## 1      0.01 post dauer stage Ce 20386 80255.   0.204  3.85
## # A tibble: 1 x 6
##   threshold lifestage   count  NLLik meanlog sdlog
##       <dbl> <chr>       <int>  <dbl>   <dbl> <dbl>
## 1       0.1 L1 larva Ce 20386 69602.   0.723  2.63
## # A tibble: 1 x 6
##   threshold lifestage   count  NLLik meanlog sdlog
##       <dbl> <chr>       <int>  <dbl>   <dbl> <dbl>
## 1       0.1 L2 larva Ce 20386 74916.   0.698  2.97
## # A tibble: 1 x 6
##   threshold lifestage   count  NLLik meanlog sdlog
##       <dbl> <chr>       <int>  <dbl>   <dbl> <dbl>
## 1       0.1 L3 larva Ce 20386 76292.   0.884  2.82
## # A tibble: 1 x 6
##   threshold lifestage   count  NLLik meanlog sdlog
##       <dbl> <chr>       <int>  <dbl>   <dbl> <dbl>
## 1       0.1 L4 larva Ce 20386 63015.   0.397  2.44
## # A tibble: 1 x 6
##   threshold lifestage count  NLLik meanlog sdlog
##       <dbl> <chr>     <int>  <dbl>   <dbl> <dbl>
## 1       0.1 adult Ce  20386 71957.   0.537  3.03
## # A tibble: 1 x 6
##   threshold lifestage      count  NLLik meanlog sdlog
##       <dbl> <chr>          <int>  <dbl>   <dbl> <dbl>
## 1       0.1 dauer larva Ce 20386 73739.   0.749  2.82
## # A tibble: 1 x 6
##   threshold lifestage           count  NLLik meanlog sdlog
##       <dbl> <chr>               <int>  <dbl>   <dbl> <dbl>
## 1       0.1 post dauer stage Ce 20386 75728.   0.720  2.98
## # A tibble: 1 x 6
##   threshold lifestage   count  NLLik meanlog sdlog
##       <dbl> <chr>       <int>  <dbl>   <dbl> <dbl>
## 1         1 L1 larva Ce 20386 66715.   0.910  2.41
## # A tibble: 1 x 6
##   threshold lifestage   count  NLLik meanlog sdlog
##       <dbl> <chr>       <int>  <dbl>   <dbl> <dbl>
## 1         1 L2 larva Ce 20386 70625.    1.08  2.48
## # A tibble: 1 x 6
##   threshold lifestage   count  NLLik meanlog sdlog
##       <dbl> <chr>       <int>  <dbl>   <dbl> <dbl>
## 1         1 L3 larva Ce 20386 72055.    1.21  2.37
## # A tibble: 1 x 6
##   threshold lifestage   count  NLLik meanlog sdlog
##       <dbl> <chr>       <int>  <dbl>   <dbl> <dbl>
## 1         1 L4 larva Ce 20386 58456.   0.693  2.05
## # A tibble: 1 x 6
##   threshold lifestage count  NLLik meanlog sdlog
##       <dbl> <chr>     <int>  <dbl>   <dbl> <dbl>
## 1         1 adult Ce  20386 67828.   0.759  2.77
## # A tibble: 1 x 6
##   threshold lifestage      count  NLLik meanlog sdlog
##       <dbl> <chr>          <int>  <dbl>   <dbl> <dbl>
## 1         1 dauer larva Ce 20386 69490.    1.07  2.39
## # A tibble: 1 x 6
##   threshold lifestage            count  NLLik meanlog sdlog
##       <dbl> <chr>                <int>  <dbl>   <dbl> <dbl>
## 1         1 post dauer stage Ce 20386 71041.    1.11  2.46
```

```
plot(x = resultTibble$threshold, y = resultTibble$NLLik, log="x")
```



```
plot(x = resultTibble$threshold, y = resultTibble$sdlog, log="x")
```



```
resultTibble %>% group_by(threshold) %>% summarise(mean(sdlog))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 4 x 2
##   threshold `mean(sdlog)`
##       <dbl>         <dbl>
## 1     0.001          4.44
## 2     0.01           3.57
## 3     0.1            2.81
## 4     1              2.42
```

Example using functions with summarise()

From: https://stackoverflow.com/questions/52718604/passing-a-list-of-arguments-to-a-function-with-quasiquotation

```
sum_fun <- function(.data, .summary_var, .group_vars) {
  summary_var <- enquo(.summary_var)

  .data %>%
    group_by_at(.group_vars) %>%
    summarise(mean = mean(!!summary_var))
}

sum_fun(mtcars, disp, .group_vars = vars(cyl, am))
```