Stomach Contents - Horse Mackerel

2024-06-28

This file will go through the process of generating the selection function for Horse Mackerel. Firstly, I need to check that it PPMR doesnt change

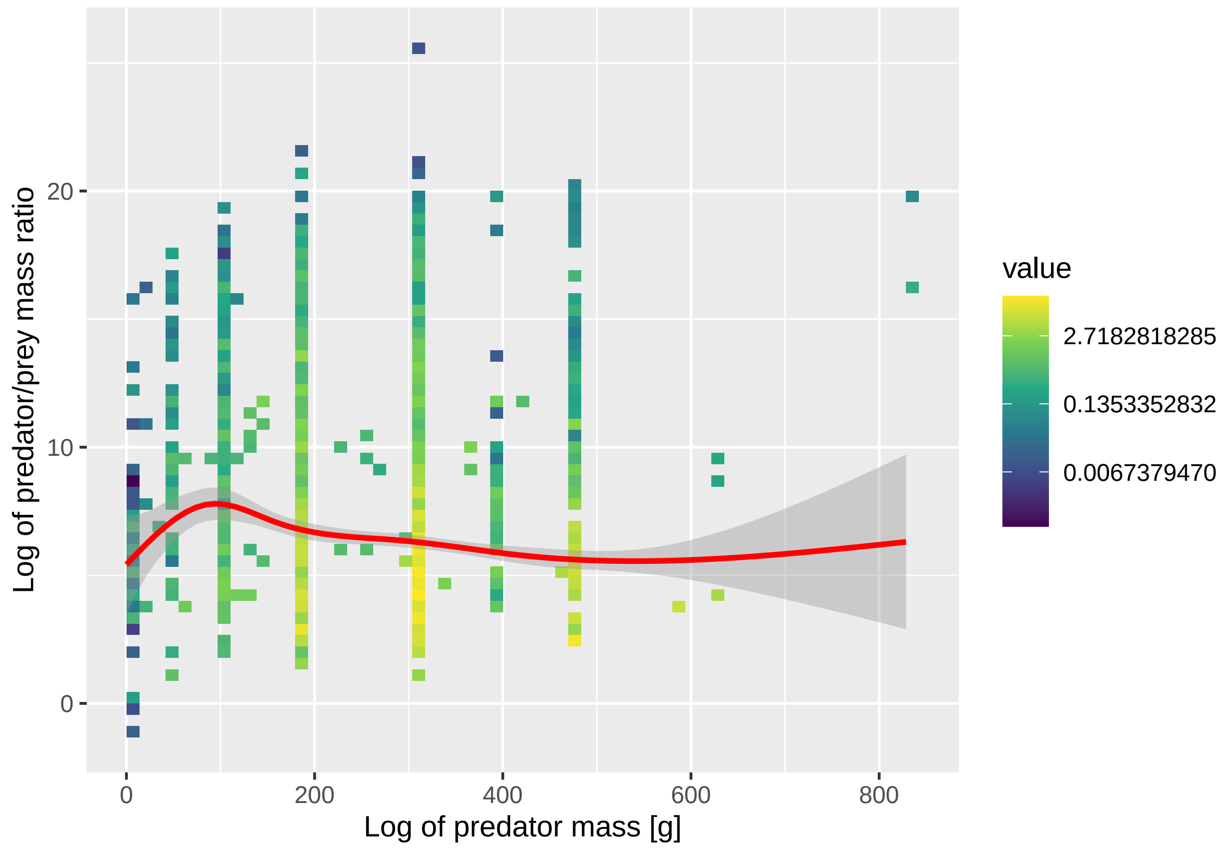
```
library(dplyr)
library(ggplot2)
library(tidyr)
library(bbmle)
library(gridExtra)
#reading in the data
load("C:/Users/lucab/Downloads/stomach_dataset.Rdata")

mack <- stom_df%>%filter(pred_species=="Trachurus trachurus")

mack <- mack%>%select(prey_ind_weight_g, pred_weight_g, nprey_perpred, pred_species,)%>%
rename(wprey=prey_ind_weight_g, wpredator=pred_weight_g, Nprey=nprey_perpred, Species=pred_species)%>%
  mutate(wprey=wprey/Nprey)

#ppmr
mack <- mack%>%mutate(ppmr=(wpredator/wprey))

#plotting the ppmr
```

This doesnt look the best, but it seems to be only the smallest sizes, and the largest that have varying PPMR, it may be that at the largest sizes the only possible prey are outside the normal range. So at the largest sizes of mackerel, their usual prey species do not grow to the same relative sizes, so the PPMR increases. And at the lowest sizes, this could be due to a limitation in sampling, and the fact that we cannot sample extremely small prey, which mackerel would be eating when they are <5g.

I will do the diet contribution now, to see if it is the same. Then, I will plot only the middle ranges, to see if it changes much.

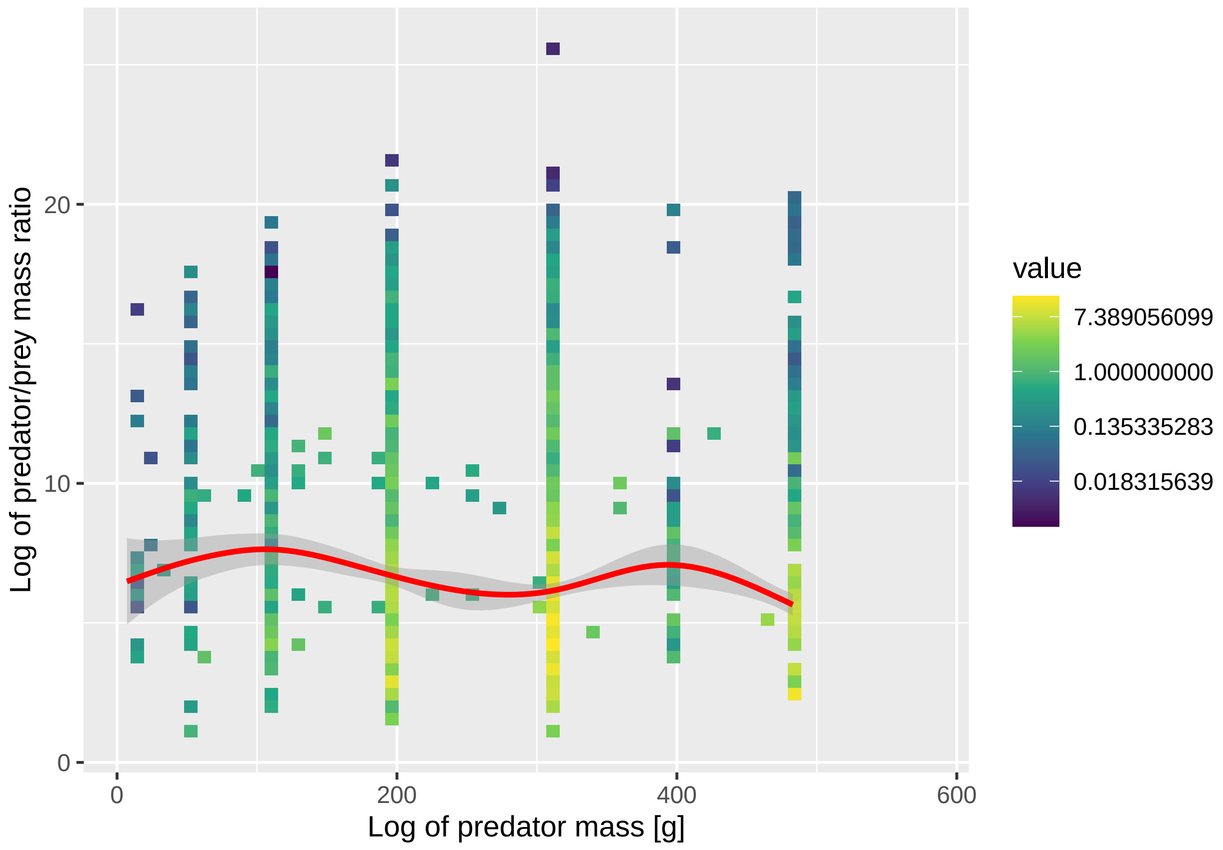
```
#diet contribution
dig <- 2/3

ggplot(mack, aes(x = wpredator, y = log(ppmr))) +
  stat_summary_2d(aes(z = Nprey * wprey^dig), fun = "sum", bins = 60) +
  scale_fill_viridis_c(trans = "log") +
  geom_smooth(aes(weight = Nprey * wprey^dig), colour = "red") +
  xlab("Log of predator mass [g]") +
  ylab("Log of predator/prey mass ratio")
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



```r
 #the same but with a limiation on the predator mass
ggplot(mack, aes(x = wpredator, y = log(ppmr))) +
  stat_summary_2d(aes(z = Nprey * wprey^dig), fun = "sum", bins = 60) +
  scale_fill_viridis_c(trans = "log") +
  geom_smooth(aes(weight = Nprey * wprey^dig), colour = "red") +
  xlab("Log of predator mass [g]") +
  ylab("Log of predator/prey mass ratio")+xlim(5, 580)
```

```
## Warning: Removed 57 rows containing non-finite outside the scale range
## (‘stat_summary2d()‘).

## ‘geom_smooth()‘ using method = ’gam’ and formula = ’y ~ s(x, bs = "cs")’

## Warning: Removed 57 rows containing non-finite outside the scale range
## (‘stat_smooth()‘).

## Warning: Removed 4 rows containing missing values or values outside the scale range
## (‘geom_tile()‘).
```



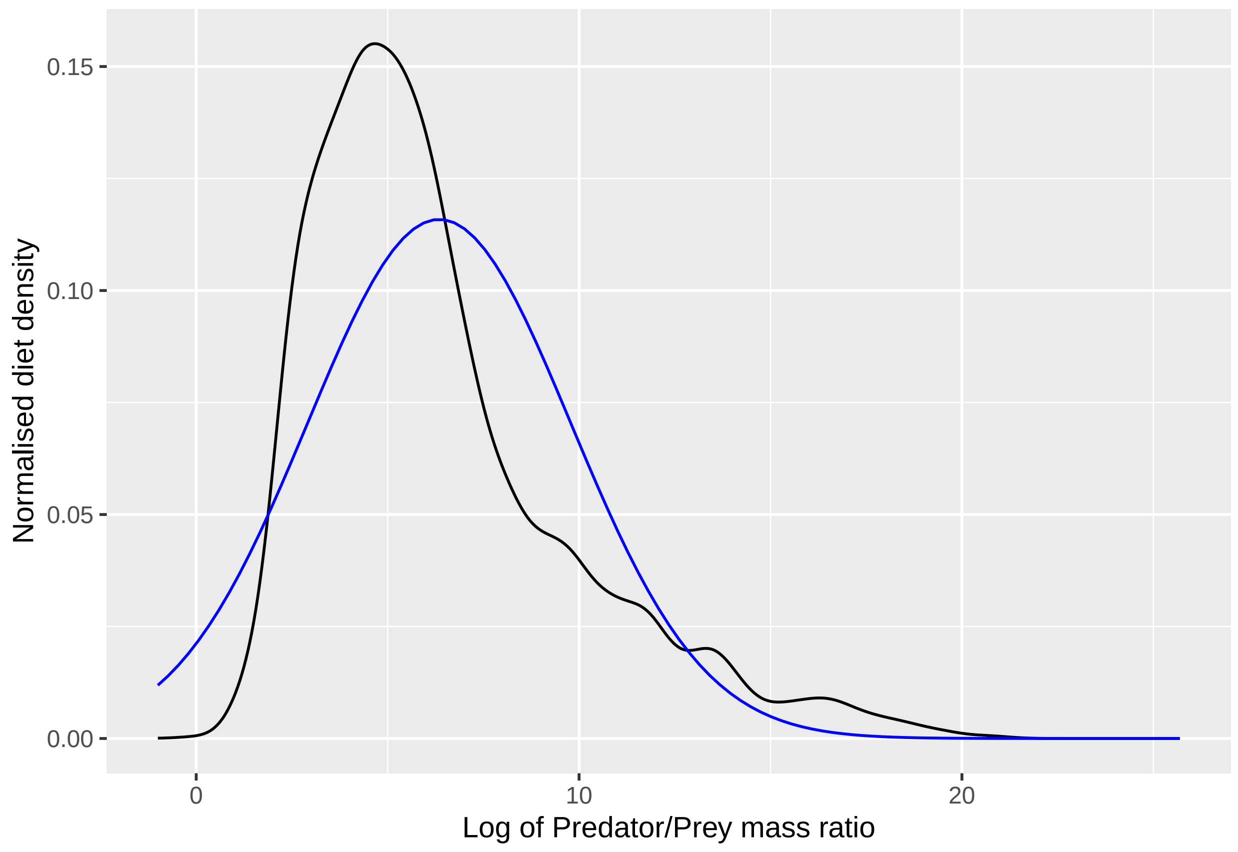
Okay this looks a lot better, but I have had to wiggle the predator mass to include by removing the largest sizes (>600g has a very high PPMR, but then the size range below (580-600g) this has a relatively lower PPMR strangely.) And I have had to make it so that any sizes less than 25g is not included, as the PPMR is much lower. I think this it is reasonable to assume that the sampling methods cannot sample the prey of <25g reliably, as the average PPMR across other size ranges is approximately 3000 (log(3000) = 8), so would correspond to a prey size of 25 / 3000 = 0.008g, which would be hard to sample especially at <25g and would be digested fast inside the mackerel as well.

So therefore, the assumption that PPMR is the same irrespective of predator size is met.

Next, we will check the distribution of PPMR values and try to fit a normal distribution.

```r
# I want to match the format I have to this code
colnames(mack) <- c("w_prey", "w_pred", "n_prey", "pred_species", "ppmr")
#now adding a log ppmr column
colnames(mack) <- make.names(colnames(mack), unique = TRUE)

mack <- mack%>%mutate(log_ppmr=log(ppmr))

stomach <- mack

weighted.sd <- function(x, w) { sqrt(sum(w * (x - weighted.mean(x, w))^2)) }
weight <- stomach$n_prey * stomach$w_prey^dig
weight <- weight / sum(weight)

est_mean <- weighted.mean(stomach$log_ppmr, weight)
est_sd <- weighted.sd(stomach$log_ppmr, weight)
```

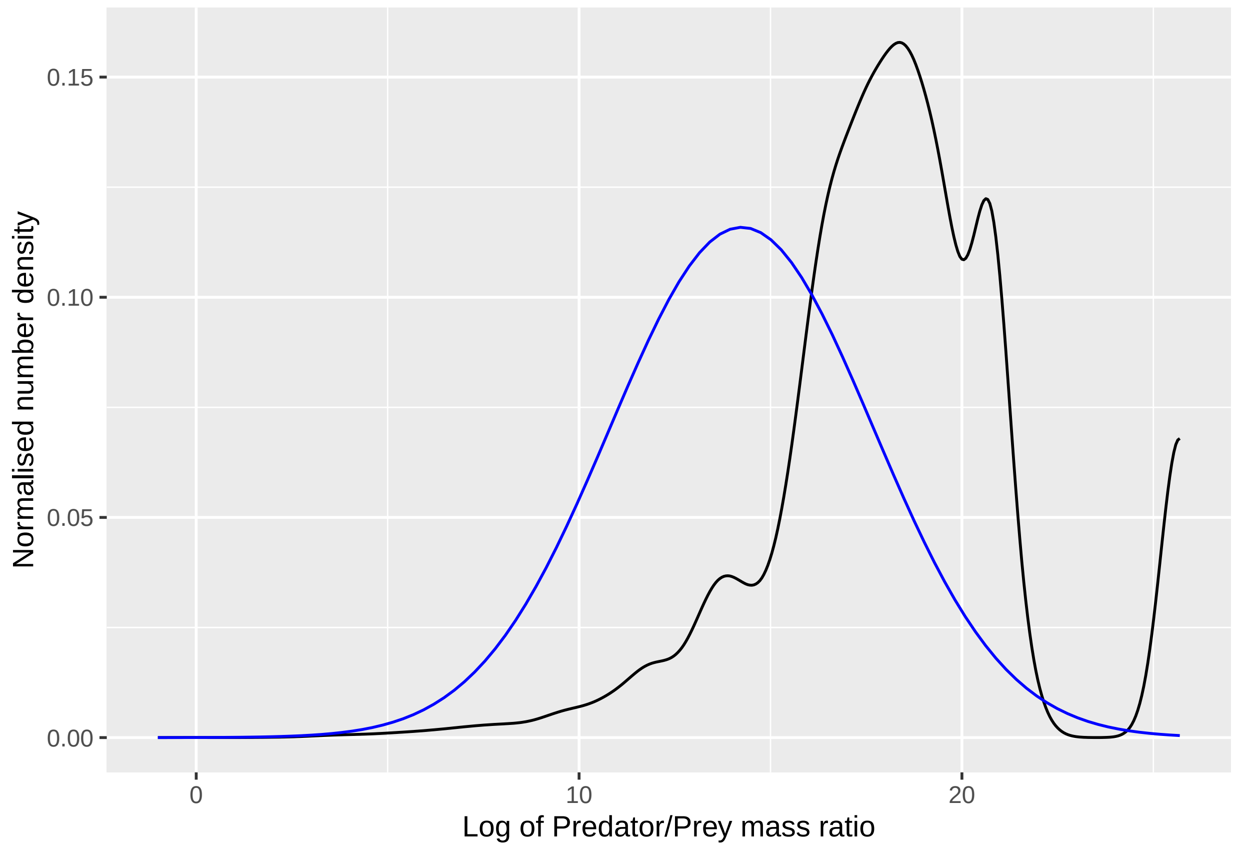
```
ggplot(stomach) +
  geom_density(aes(log_ppmr, weight = n_prey * w_prey^dig), bw = 0.5) +
  stat_function(fun = dnorm, args = list(mean = est_mean, sd = est_sd), colour = "blue") +
  xlab("Log of Predator/Prey mass ratio") + ylab("Normalised diet density")
```



Ok, this isnt a terrible fit, but it seems Horse Mackerel eat multimodal diets, I will do the same but also remove the outliers (at the smallest and largest sizes)

I am now going to check it for the number density as well. Fitting the same sd but with a transformed mean.

```
ggplot(stomach) +
  geom_density(aes(log_ppmr, weight = n_prey), bw = 0.5) +
  stat_function(fun = dnorm, args = list(mean = est_mean + (dig * est_sd^2), sd = est_sd), colour = "blu
  xlab("Log of Predator/Prey mass ratio") + ylab("Normalised number density")
```



This doesnt fit well at all.

So I don't think that the normal distribution fits very well to Horse Mackerel, so we will need to fit another distribution.

Now lets try the code to fit a truncated exponential distribution to the data.

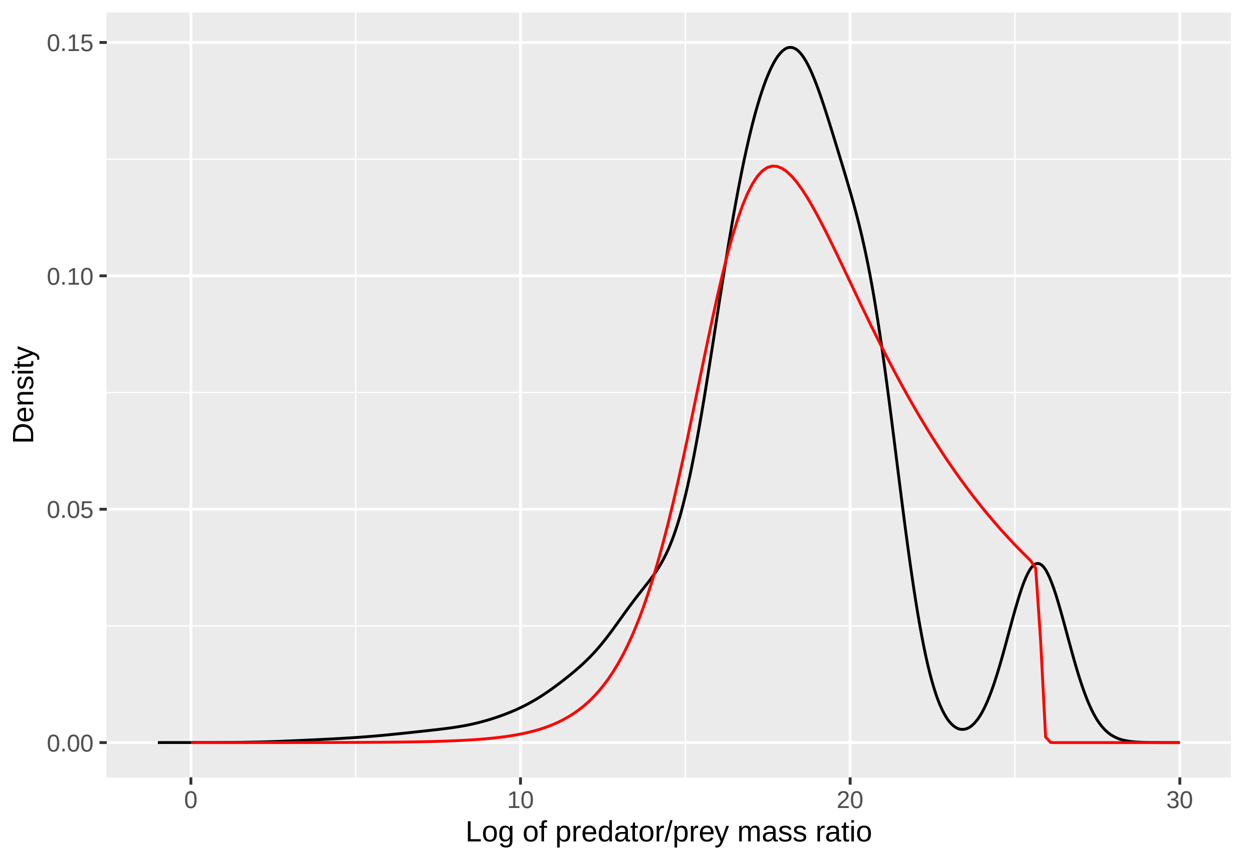
Okay, this is code to define the exponential distribution (taken from the PPMR distribution file.) First, I will do this for the number distribution

```
stomach <- mack
colnames(stomach) <- c("wprey", "w_pred", "Nprey", "pred_species", "ppmr", "l")
stomach <- stomach %>% mutate(weight_numbers = Nprey / sum(Nprey))
#stomach <- stomach%>%mutate(weight_numbers=Nprey*wprey^(2/3))

fl <- function(l, alpha, ll, ul, lr, ur) {
  dl <- ll - l
  dr <- l - lr
  fl <- exp(alpha * l) /
    (1 + exp(ul * dl)) /
    (1 + exp(ur * dr))
  # fl[fl <= 0] <- 0
}

dtexp <- function(l, alpha, ll, ul, lr, ur) {
  d <- fl(l, alpha, ll, ul, lr, ur) /
    integrate(fl, 0, 30, alpha = alpha,
              ll = ll, ul = ul, lr = lr, ur = ur)$value
```

```r
  return(d)
}

mle_texp <- function(df) {
  loglik <- function(alpha, ll, ul, lr, ur) {
    L <- dtexp(stomach$l, alpha, ll, ul, lr, ur)
    - sum(log(L) *  stomach$weight_numbers)
  }
  mle2(loglik, start = list(
    alpha = 0.5,
    ll = min(stomach$l),
    lr = max(stomach$l),
    ul = 5,
    ur = 5))
}

#setting th weights
#stomach <- stomach%>%mutate(weight_numbers=Nprey*wprey^(2/3))
#or for number density
#stomach <- stomach%>%mutate(weight_numbers=Nprey/sum(Nprey))

est <- mle_texp(stomach)
```

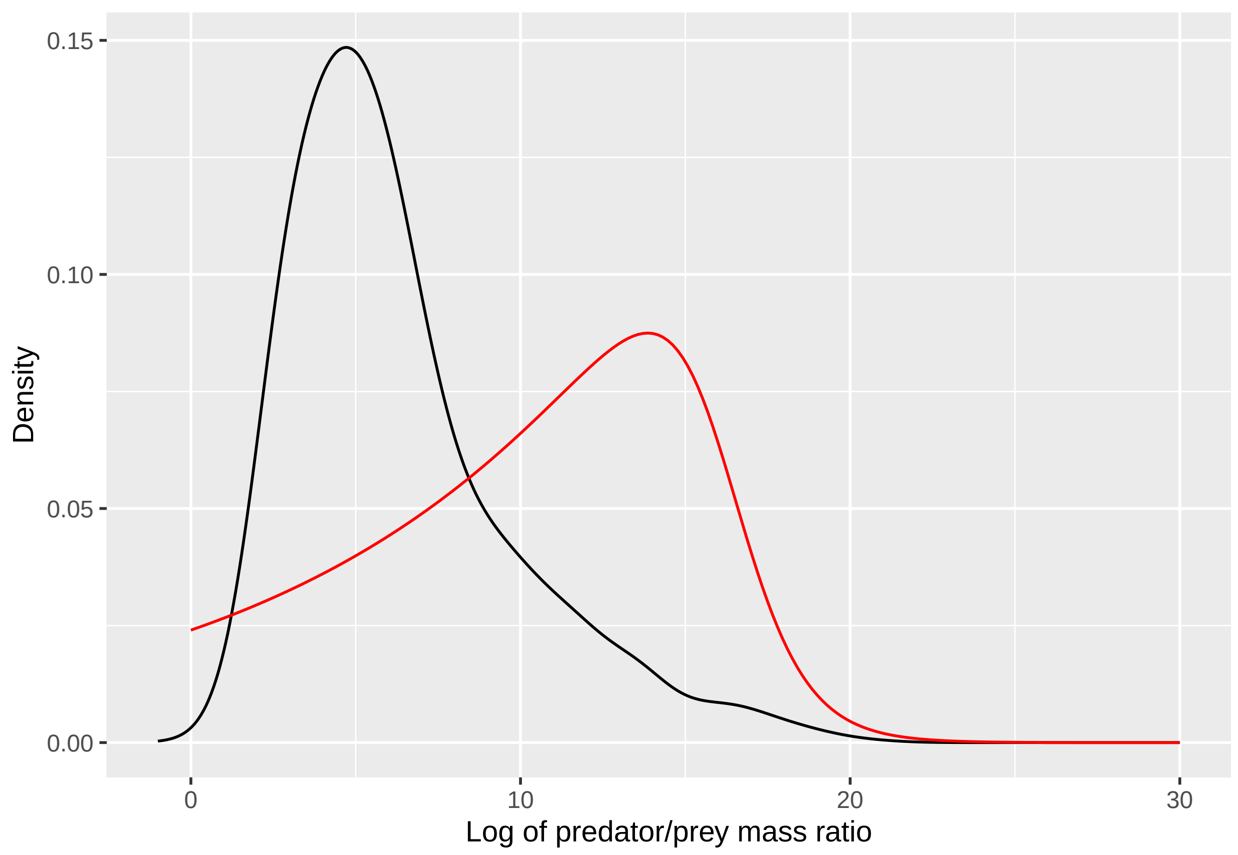
```
## Warning in mle2(loglik, start = list(alpha = 0.5, ll = min(stomach$l), lr =
## max(stomach$l), : convergence failure: code=1 (iteration limit 'maxit' reached)
```

```r
#extracting coefficients
estco <- est@fullcoef

grid = seq(0, 30, length.out = 200)
dist <- dtexp(grid, alpha = estco[1], ll = estco[2], ul = estco[3], lr = estco[4], ur = estco[5])
dist <- data.frame(l=grid, Density=dist)

ggplot(stomach) +
  geom_density(aes(l, weight=weight_numbers))+
 xlab("Log of predator/prey mass ratio") +
  geom_line(aes(l, Density), data = dist, color = "red")
```



Ok, this works, it fits well. Now I have to fit this same distribution to the biomass density, which as defined in the PPMR distributions document, you do alpha-2/3 (I think)

```r
stomach <- stomach%>%mutate(biomass=Nprey*wprey^(2/3))

grid = seq(0, 30, length.out = 200)
#here, the alpha is meant to be -1, but I have to subtract 0.7 to make it work,
#so I am going to run the distribution again
#for the biomass, and see the difference
dist <- dtexp(grid, alpha = (estco[1]-(2/3)), ll = estco[2], ul = estco[3], lr = estco[4], ur = estco[5

dist <- data.frame(l=grid, Density=dist)

ggplot(stomach) +
  geom_density(aes(l, weight=biomass))+
 xlab("Log of predator/prey mass ratio") +
  geom_line(aes(l, Density), data = dist, color = "red")
```



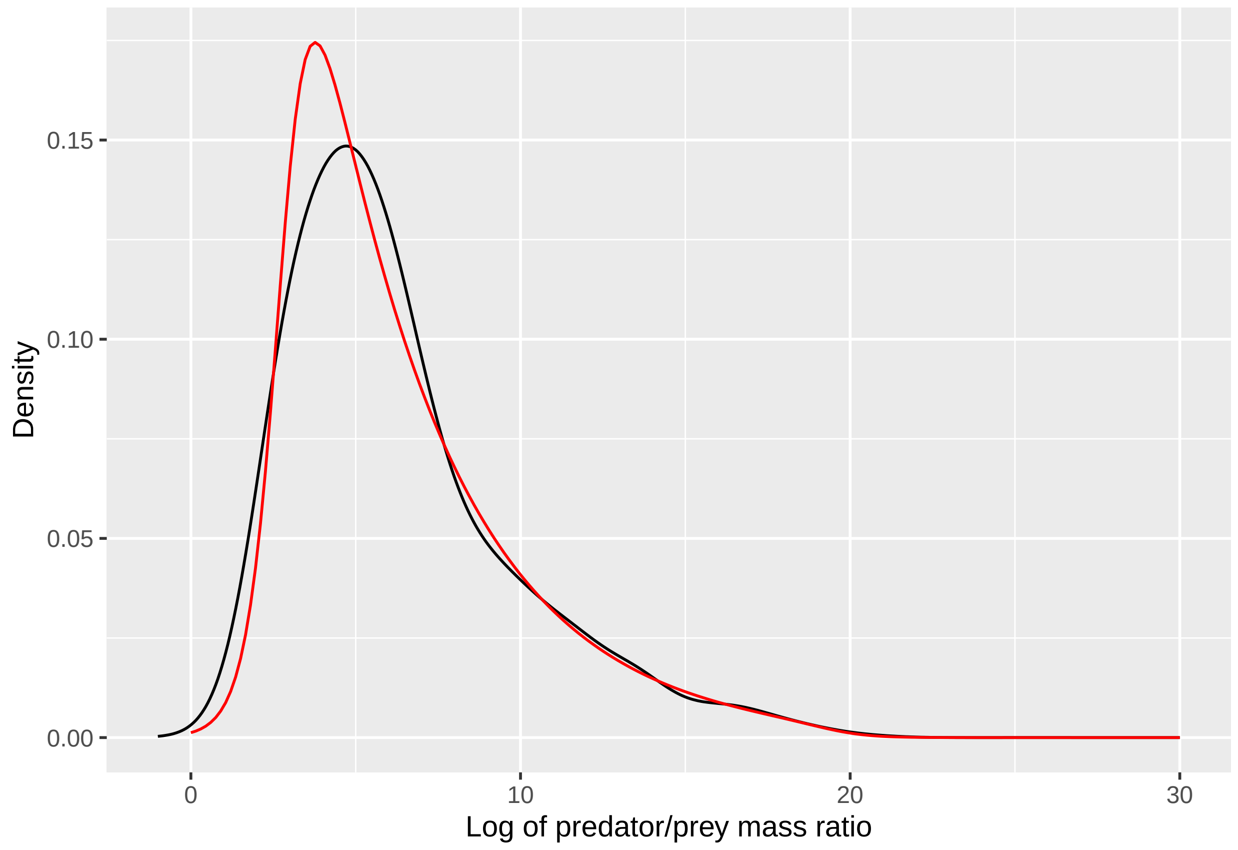
Ok this doesn't fit as well. I will calculate the proper distribution independently and see what the difference is.

It isn't a terrible fit when the alpha is -0.7, which is strange, I am not sure why this might be.

(when trying to use this code to calculate the biomass distribution, I get an error, Error in integrate(fl, 0, 30, alpha = alpha, ll = ll, ul = ul, lr = lr,: non-finite function value), this code works for it though, but provides a slightly different distribution for the number density. The difference in the code is this line , method = "L-BFGS-B", control = list(maxit = 10000) in the MLE function, changing the optimisation algorithm and the maximum number of iterations.)

```
#library(bbmle)
#library(dplyr)
#
##884 is problem line for the number density (idk why)
##stomach <- stomach[-884,]

## Define the function fl with debugging
fl <- function(l, alpha, ll, ul, lr, ur) {
  dl <- ll - l
  dr <- l - lr
  fl_values <- exp(alpha * l) / (1 + exp(ul * dl)) / (1 + exp(ur * dr))

  # Debugging output
  if (any(!is.finite(fl_values))) {
    print("Non-finite fl values found")
    print(fl_values)
```

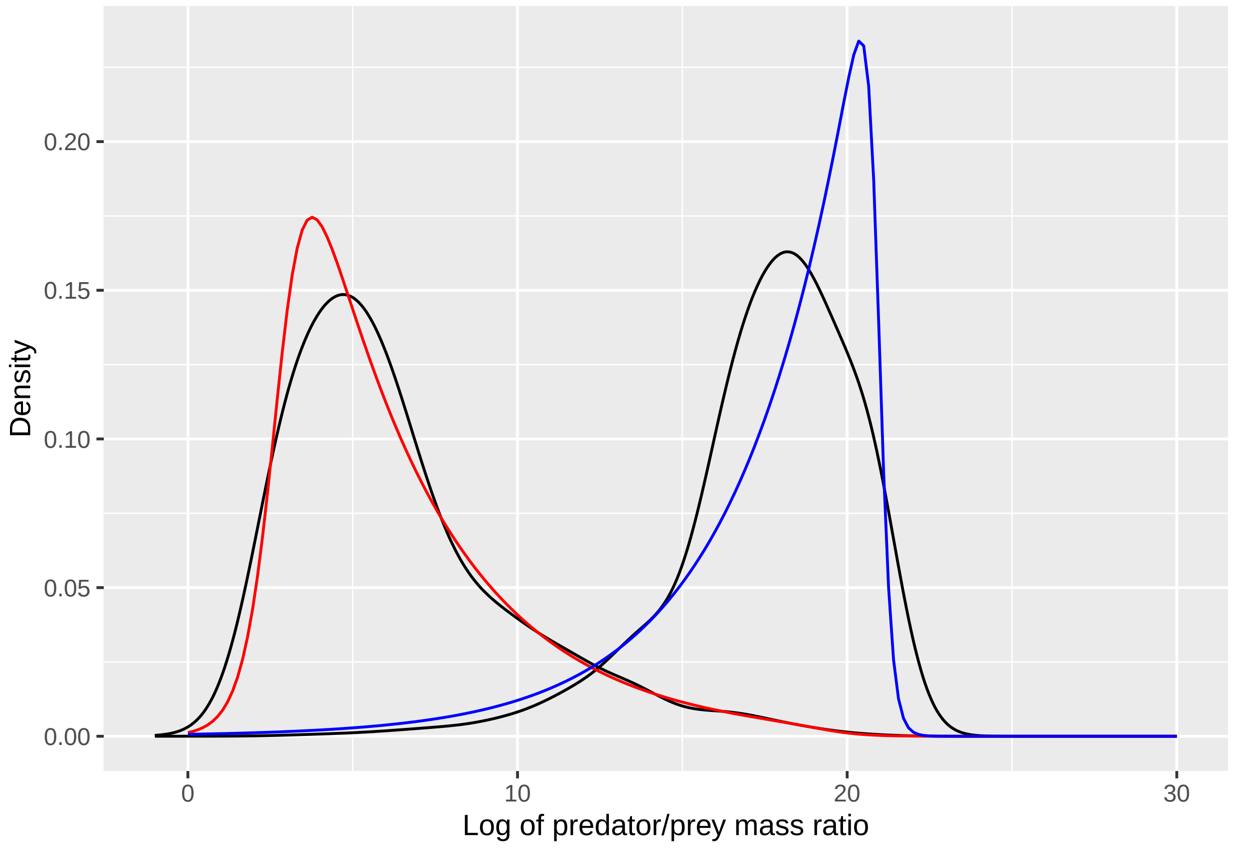
```r
  }

  return(fl_values)
}

## Define the truncated exponential PDF with debugging
dtexp <- function(l, alpha, ll, ul, lr, ur) {
  fl_values <- fl(l, alpha, ll, ul, lr, ur)

  integral_result <- tryCatch(
    integrate(fl, 0, 30, alpha = alpha, ll = ll, ul = ul, lr = lr, ur = ur),
    error = function(e) {
      print("Integration failed")
     print(e)
      return(NULL)
    }
   )

   if (is.null(integral_result)) {
     return(rep(NA, length(l)))
   }

   d <- fl_values / integral_result$value

  # Debugging output
  if (any(!is.finite(d))) {
    print("Non-finite d values found")
    print(d)
  }

  return(d)
}

 #Define the MLE function with debugging
mle_texp <- function(df) {
  loglik <- function(alpha, ll, ul, lr, ur) {
    L <- dtexp(df$l, alpha, ll, ul, lr, ur)

    # Debugging output
    if (any(!is.finite(L) | L <= 0)) {
     print("Non-finite or non-positive L values found")
      print(which(!is.finite(L) | L <= 0))
      return(Inf)
    }

    -sum(log(L) * df$weight_numbers)
  }

  result <- tryCatch(
    mle2(loglik, start = list(
      alpha = 0.5,
      ll = min(df$l),
      lr = max(df$l),
      ul = 5,
      ur = 5
    ), method = "L-BFGS-B", control = list(maxit = 10000)),
    error = function(e) {
      print("MLE fitting failed")
      print(e)
      return(NULL)
    }
  )

  return(result)
}

# Assuming 'stomach' is already defined
# setting the weights
stomach <- stomach %>% mutate(weight_numbers = Nprey * wprey^(2/3))
# or for number density
#stomach <- stomach %>% mutate(weight_numbers = Nprey / sum(Nprey))

# Fit the model
est <- mle_texp(stomach)

biomassco <- est@coef

grid = seq(0, 30, length.out = 200)
#here, the alpha is meant to be -1, but I have to subtract 0.7 to make it work, so I am going to run th
#for the biomass, and see the difference
dist <- dtexp(grid, alpha = (biomassco[1]), ll = biomassco[2], ul = biomassco[3],
              lr = biomassco[4], ur = biomassco[5])

dist <- data.frame(l=grid, Density=dist)

  ggplot(stomach) +
  geom_density(aes(l, weight=weight_numbers))+
 xlab("Log of predator/prey mass ratio") +
  geom_line(aes(l, Density), data = dist, color = "red")
```



Ok, fitting the biomass distribution works well. I will plot the fits of both distributions below, while using the same code for both distributions

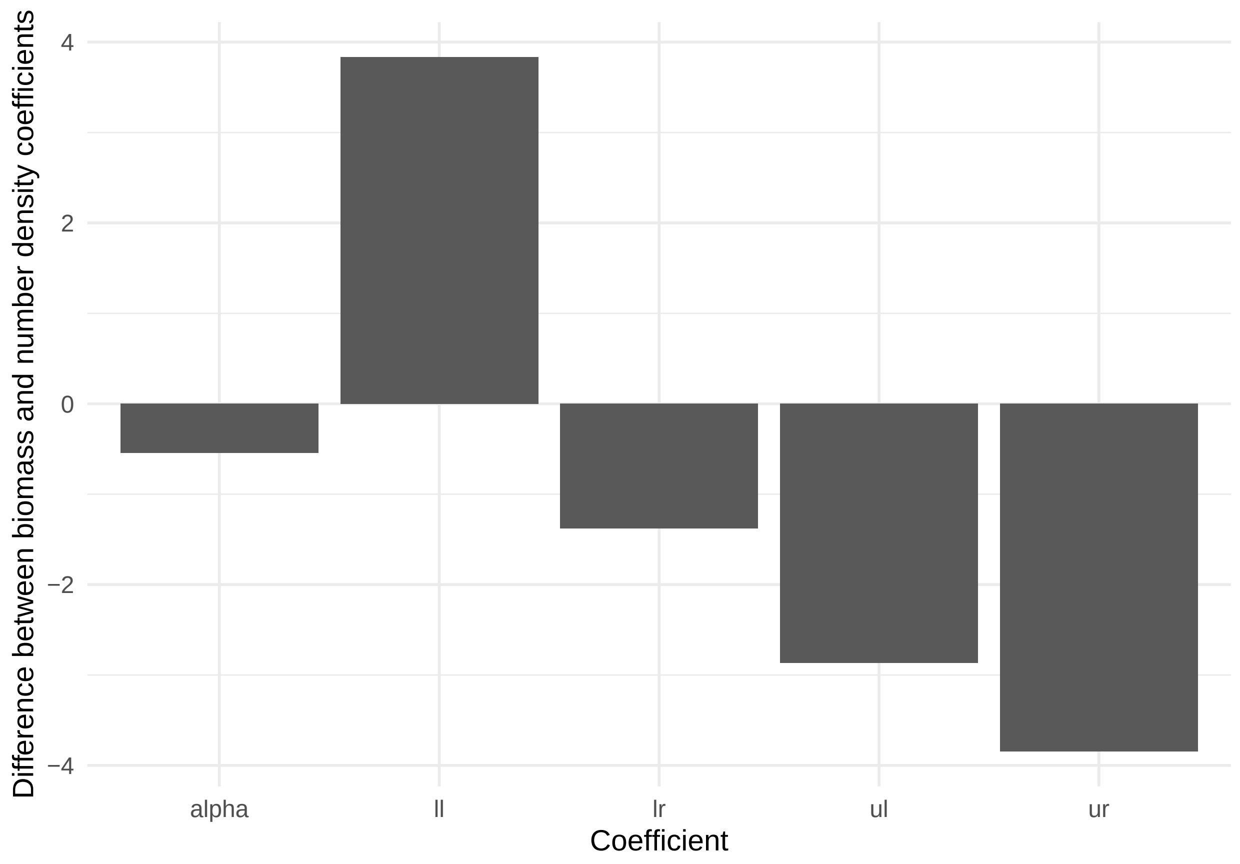
```
#884 is problem line for the number density (idk why)
stomach <- stomach[-884,]

stomach <- stomach %>% mutate(weight_numbers = Nprey * wprey^(2/3))
est <- mle_texp(stomach)
biomassestco <- est@coef

stomach <- stomach %>% mutate(weight_numbers = Nprey / sum(Nprey))
est <- mle_texp(stomach)
numberestco <- est@coef

grid = seq(0, 30, length.out = 200)
dist <- dtexp(grid, alpha = (biomassestco[1]), ll = biomassestco[2], ul = biomassestco[3], lr = biomass
biomassdist <- data.frame(l=grid, Density=dist)

dist <- dtexp(grid, alpha = (numberestco[1]), ll = numberestco[2], ul = numberestco[3], lr = numberestc
numberdist <- data.frame(l=grid, Density=dist)

#now plot these two together

stomach <- stomach %>% mutate(biomass = Nprey * wprey^(2/3))
stomach <- stomach %>% mutate(weight_numbers = Nprey / sum(Nprey))
```

```r
ggplot(stomach) +
  geom_density(aes(l, weight=weight_numbers))+
  geom_density(aes(l, weight=biomass))+
 xlab("Log of predator/prey mass ratio") +
  geom_line(aes(l, Density), data = biomassdist, color = "red")+
  geom_line(aes(l, Density), data = numberdist, color = "blue")
```



So they fit their individual distributions well. but the differences between the coefficients are quite large. I will calculate the difference between the two coefficients

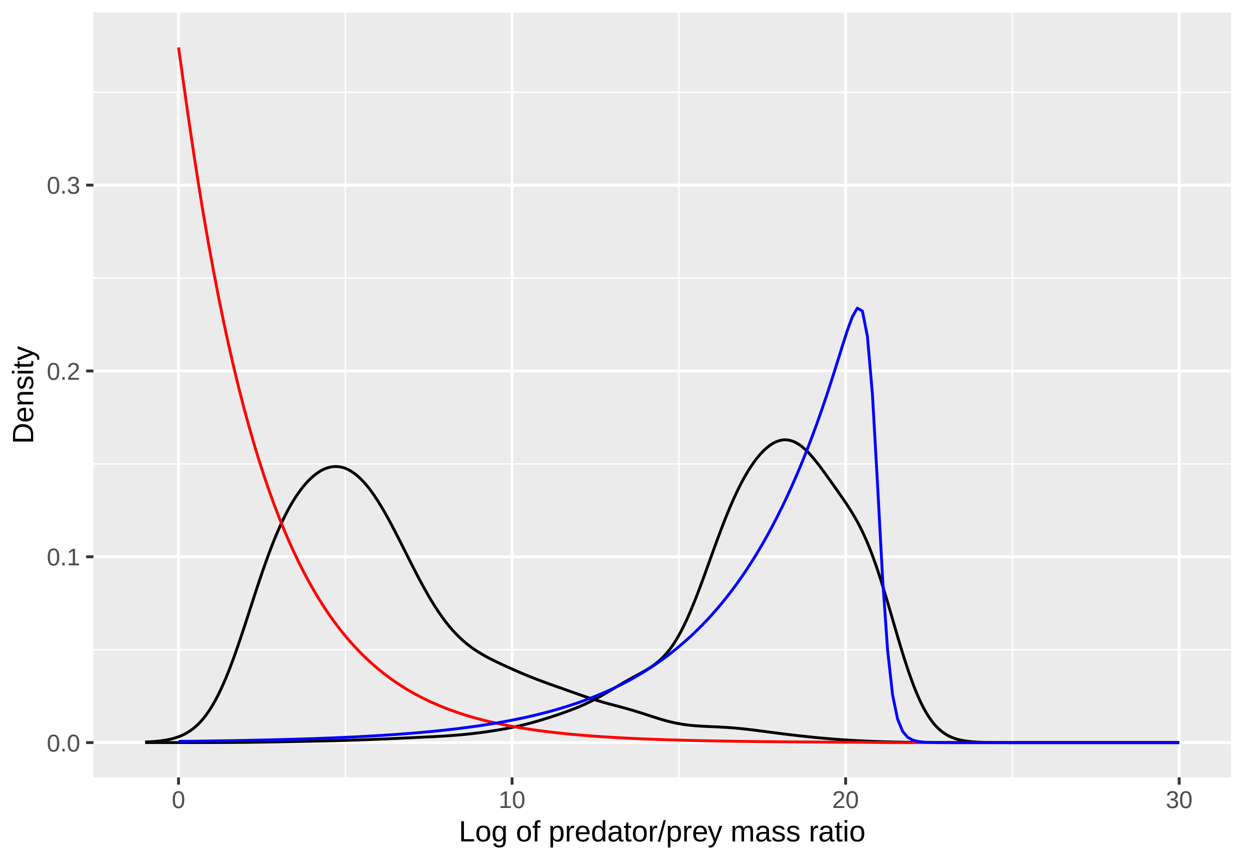
```r
diffco <- biomassestco-numberestco

#now ggplot the diffco
diffco <- data.frame(Coefficient = labels <- c("alpha", "ll", "ul", "lr", "ur"),diffco)

ggplot(diffco, aes(x = factor(Coefficient), y = diffco)) +
  geom_bar(stat = "identity") +
  xlab("Coefficient") +
  ylab("Difference between biomass and number density coefficients") +
  theme_minimal()
```



The difference between the alpha is not 1, it is about 0.625. I am not sure why, as stated in the PPMR document, the difference should be 2/3 (I am also not sure why this is)

Now I will just plot the distribution derived from the number density, and I will try to change the alpha value to calculate the truncated exponential distribution for the biomass density by -2/3.

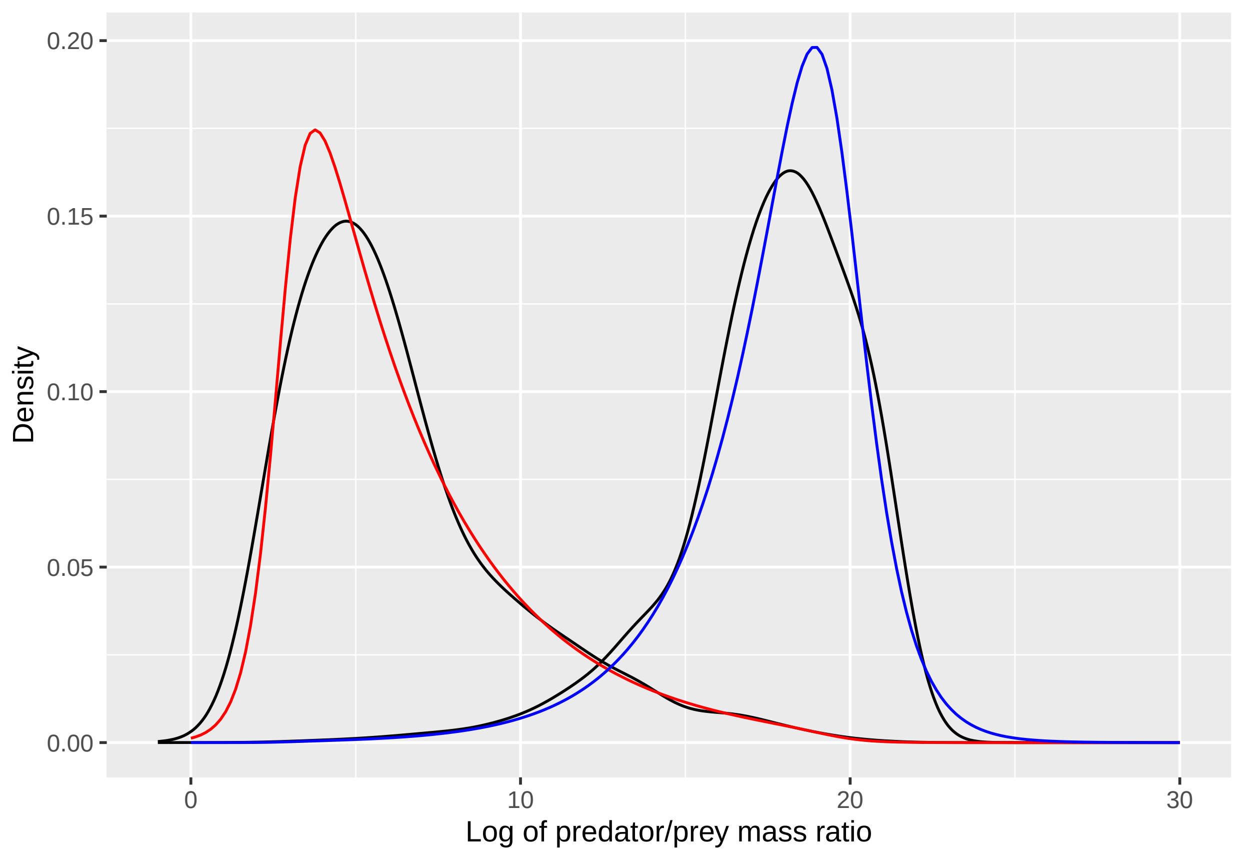
```
dist <- dtexp(grid, alpha = (numberestco[1]), ll = numberestco[2], ul = numberestco[3], lr = numberestc
numberdist <- data.frame(l=grid, Density=dist)
dist <- dtexp(grid, alpha = (numberestco[1]-(2/3)), ll = numberestco[2], ul = numberestco[3], lr = numbe
biomassdist <- data.frame(l=grid, Density=dist)
#now plot these two together

stomach <- stomach %>% mutate(biomass = Nprey * wprey^(2/3))

ggplot(stomach) +
  geom_density(aes(l, weight=weight_numbers), color="black")+
  geom_density(aes(l, weight=biomass))+
 xlab("Log of predator/prey mass ratio") +
  geom_line(aes(l, Density), data = biomassdist, color = "red")+
  geom_line(aes(l, Density), data = numberdist, color = "blue")
```



It fits ok, but it is still not very good for the biomass density (the black line on the left)

I will try to use the density function derived from the biomass density to get a distribution to use for the number density.

```
dist <- dtexp(grid, alpha = (biomassestco[1]), ll = biomassestco[2], ul = biomassestco[3], lr = biomass
biomassdist <- data.frame(l=grid, Density=dist)
dist <- dtexp(grid, alpha = (biomassestco[1]+2/3), ll = biomassestco[2], ul = biomassestco[3], lr = bio
numberdist <- data.frame(l=grid, Density=dist)
#now plot these two together

stomach <- stomach %>% mutate(biomass = Nprey * wprey^(2/3))

ggplot(stomach) +
  geom_density(aes(l, weight=weight_numbers))+
  geom_density(aes(l, weight=biomass))+
 xlab("Log of predator/prey mass ratio") +
  geom_line(aes(l, Density), data = biomassdist, color = "red")+
  geom_line(aes(l, Density), data = numberdist, color = "blue")
```



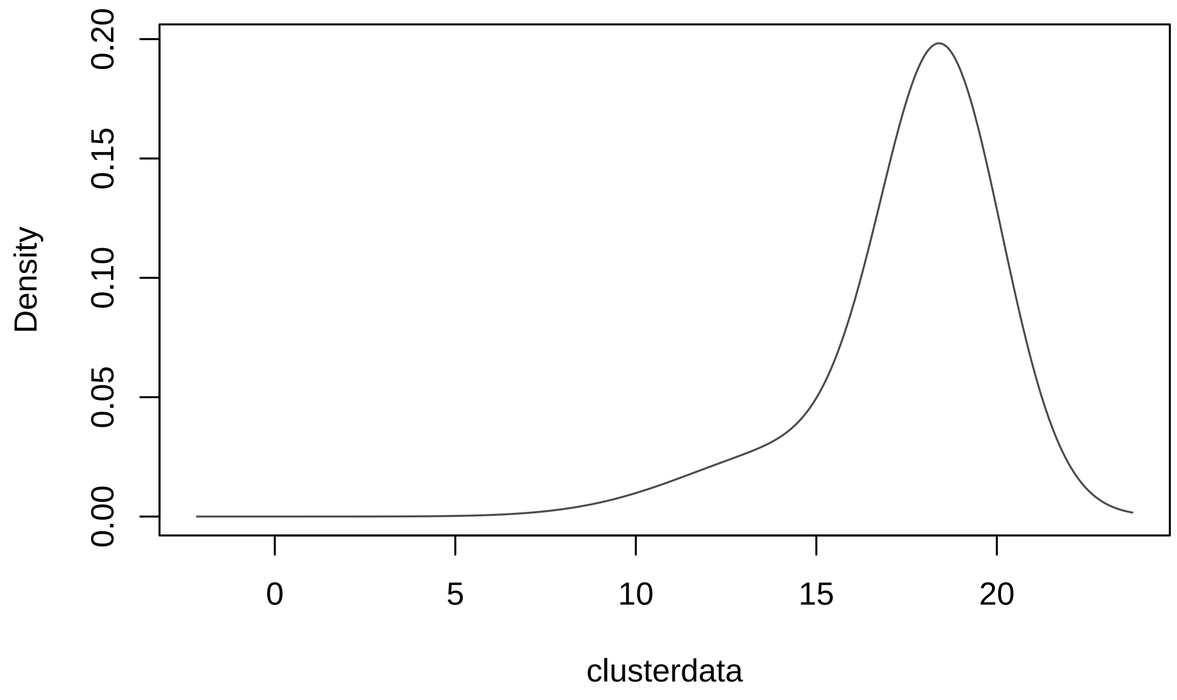
This fits a lot better, but the multi-modality of the diet density is not modeled.

This next section is attempting to fit a mixture gaussian distribution to the data

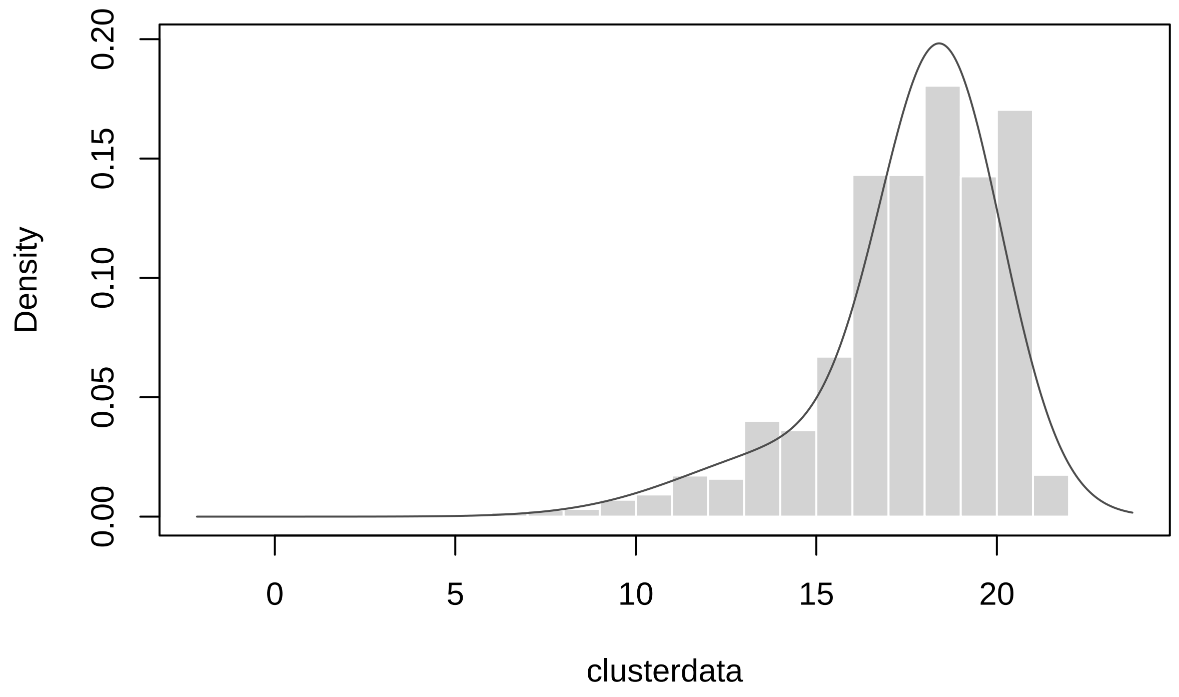
```r
#load packages and also change data format to work (weight it correctly)
library(mclust)
```

```
## Warning: package 'mclust' was built under R version 4.4.1

## Package 'mclust' version 6.1.1
## Type 'citation("mclust")' for citing this R package in publications.
```

```r
repeat_elements <- function(data, weights) {
  # Initialize an empty vector to store repeated elements
  final_vector <- c()

  # Loop through each element in data
  for (i in seq_along(data)) {
    # Get the corresponding weight and round it
    rounded_weight <- round(weights[i])

    # Repeat the current element from data by rounded_weight times
    repeated_values <- rep(data[i], times = rounded_weight)

    # Append the repeated values to the final vector
    final_vector <- c(final_vector, repeated_values)
  }

  return(final_vector)
}

clusterdata <- repeat_elements(stomach$l, stomach$Nprey)

gmm <- densityMclust(clusterdata, G=2)
```



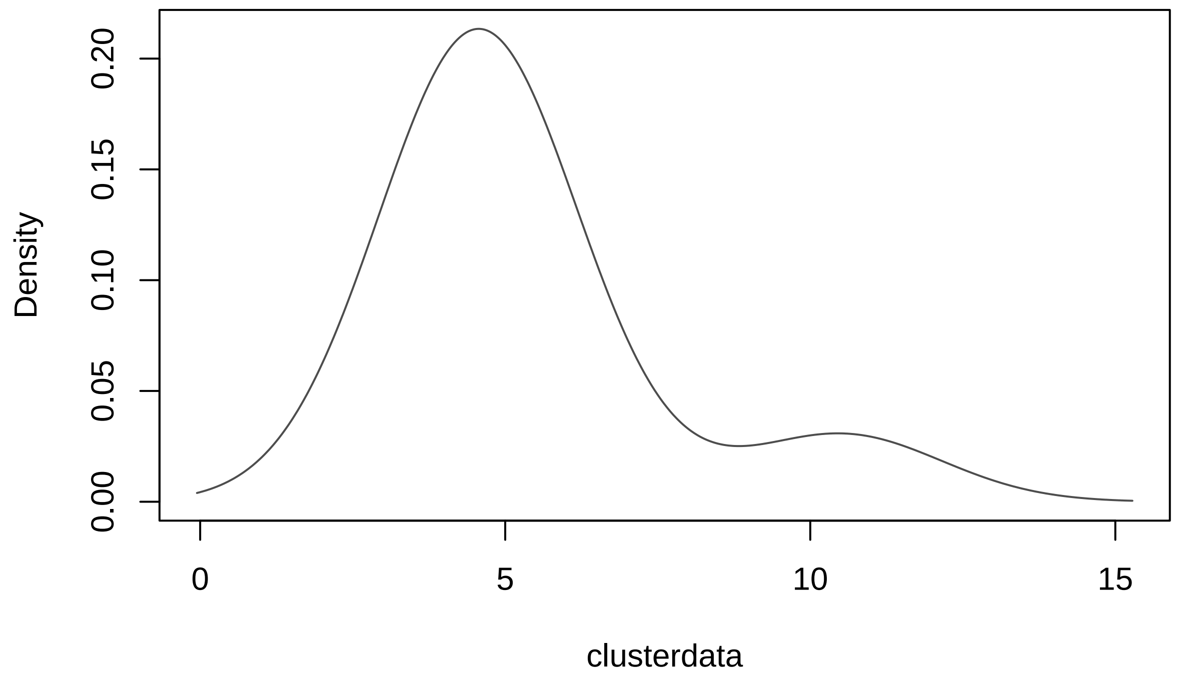
```r
plot(gmm, what = "density", data = clusterdata, breaks = 30)
```



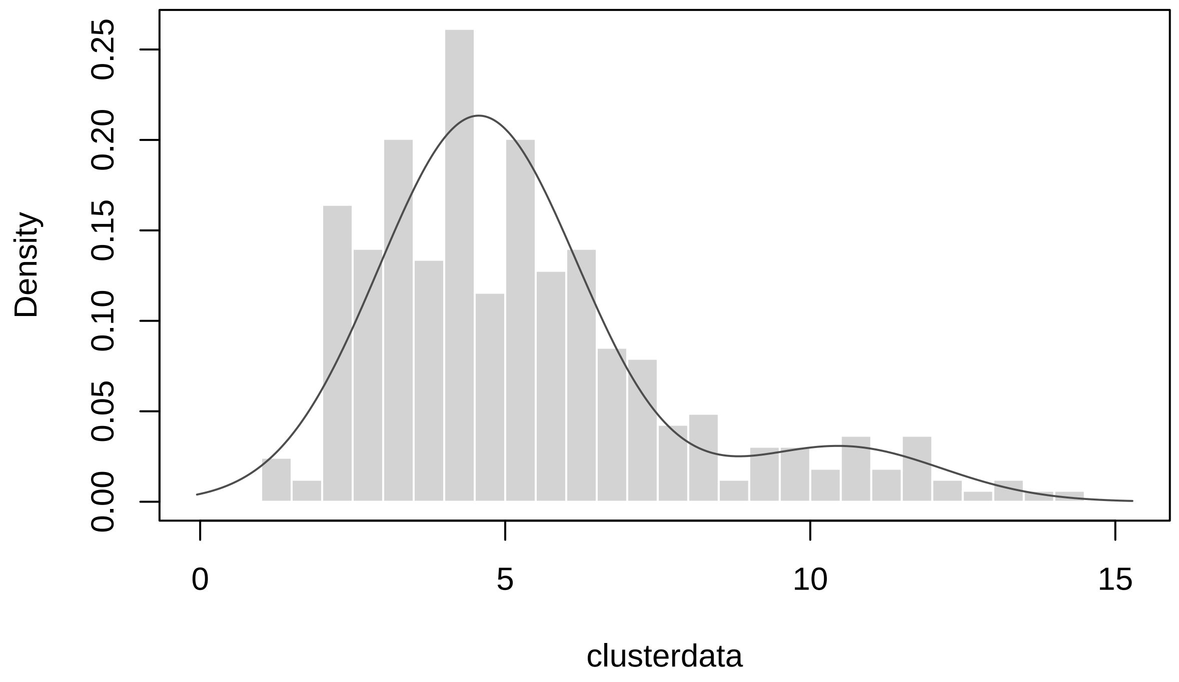
This works for the number density, will do the biomass density now then compare the means, and see what I can find out

Okay, there is a problem, the data needs to be in vector format for densityMclust But this would mean that I would have to weight the vectors by multiplying the value by the weight, but this is isn't an integer when looking at the biomass so it may not work well.

```
clusterdata <- repeat_elements(stomach$l, stomach$Nprey*stomach$wprey^(2/3))

biomassgmm <- densityMclust(clusterdata, G=2)
```



```r
plot(biomassgmm, what = "density", data = clusterdata, breaks = 30)
```

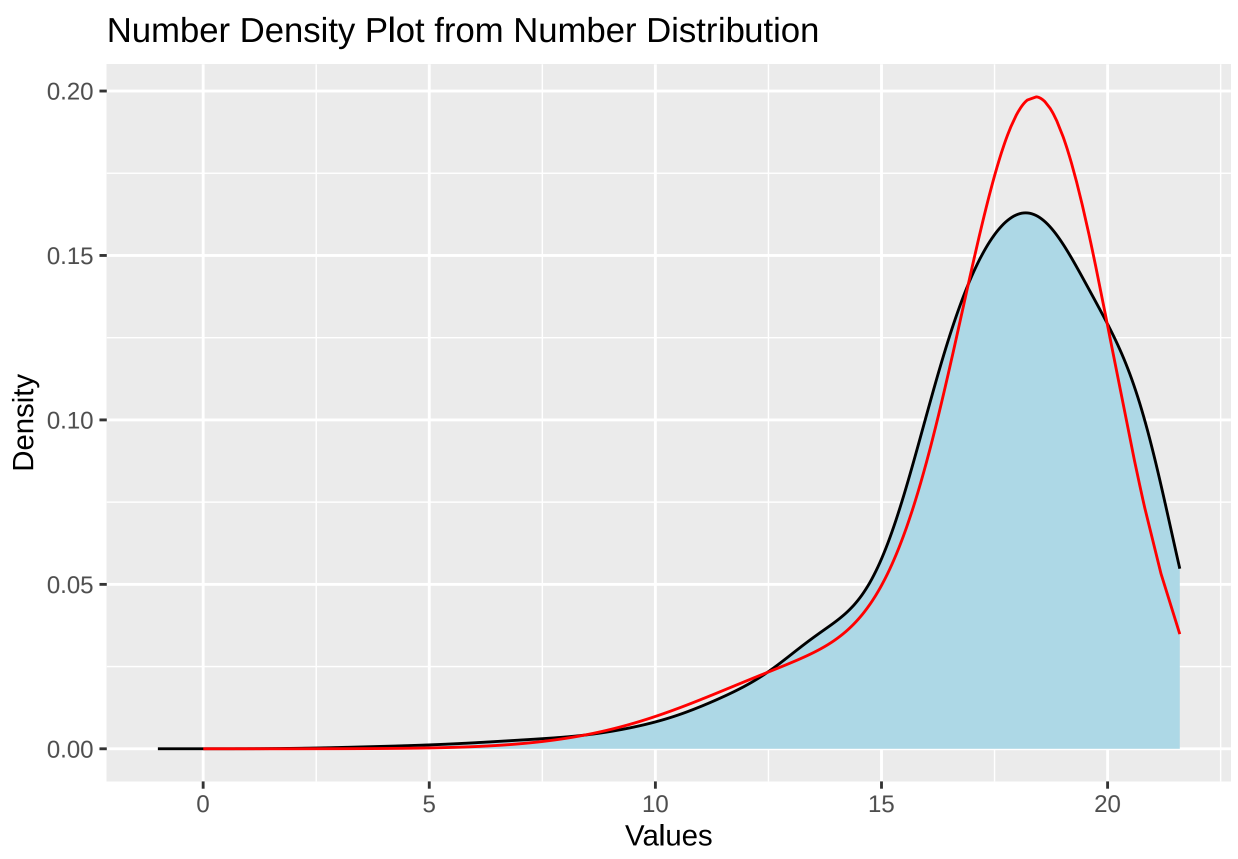


Ok the biomass fits nice too. Next I will try to transform one of the distributions so that we do not need 2 equations.

Will first try to shift the number distribution to the biomass, I am doing this by shifting the mean by the variance (of each of the gaussians) multipled by 2/3.

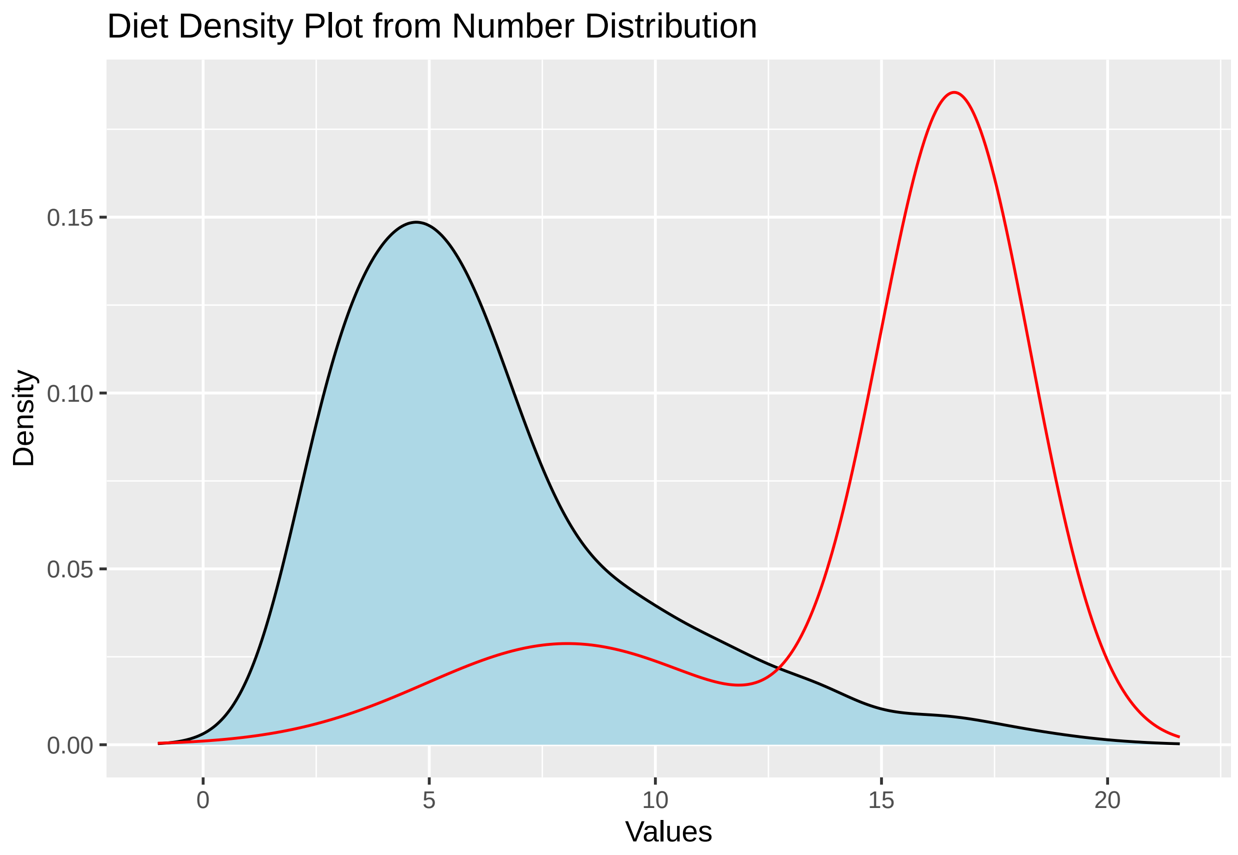
Trying to extract the data now so I can plot on top of other density plots

```
dplot <- data.frame(x=gmm[["data"]], density=gmm[["density"]])

#weightings, necessary later
stomach$weight_numbers <- stomach$Nprey
stomach$weight_biomass <- stomach$Nprey*stomach$wprey^(2/3)


# Plot with ggplot2
(numbfit <- ggplot() +
    geom_density(data=stomach, aes(l, weight=weight_numbers), fill="lightblue")+
  geom_line(data=dplot, aes(x = x, y = density), color="red") +
  labs(x = "Values", y = "Density") +
  ggtitle("Number Density Plot from Number Distribution"))
```



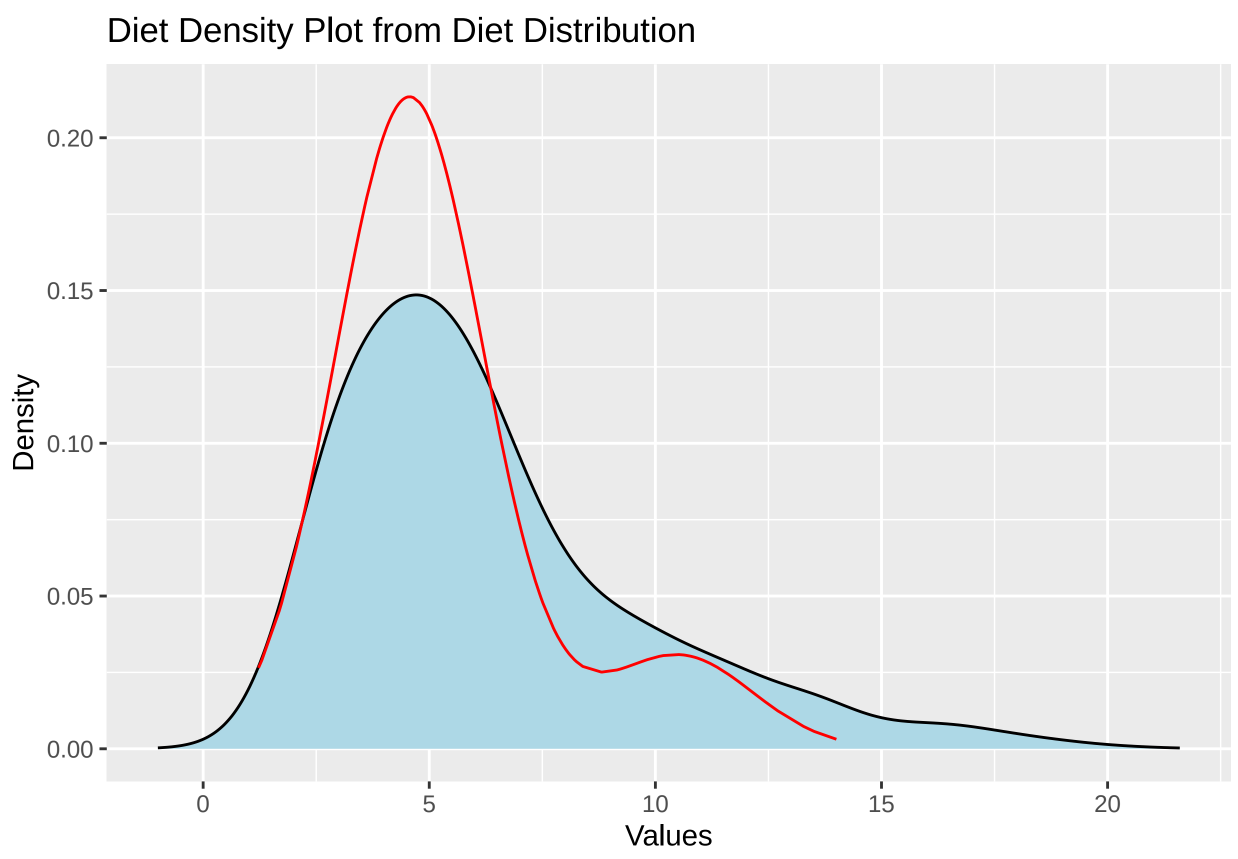
Ok, I am able to extract and plot, now I will try to plot the biomass weighting. As I am transforming the diet PDF to fit the biomass density plots, I should -2/3 of the variance.

```
#add biomass column
stomach$weight_biomass <- stomach$Nprey*stomach$wprey^(2/3)
stomach$weight_biomass <- stomach$weight_biomass/sum(stomach$weight_biomass)

gmm2 <- gmm
gmm2[["parameters"]][["mean"]] <- gmm2[["parameters"]][["mean"]]-(2/3)*gmm2[["parameters"]][["variance"]

#so this next section of code is generating new densities from the new PDF.
x_vals <- seq(min(stomach$l), max(stomach$l), length.out = 1000)

shifted_fit <- gmm2
shifted_pdf <- sapply(x_vals, function(x) sum(shifted_fit$parameters$pro * dnorm(x, mean = shifted_fit$p
plot_data <- data.frame(x = x_vals, shifted_pdf = shifted_pdf)

(numfitbio <- ggplot() +
    geom_density(data=stomach, aes(l, weight=weight_biomass), fill="lightblue")+
  geom_line(data=plot_data, aes(x = x, y = shifted_pdf), color="red") +
  labs(x = "Values", y = "Density") +
  ggtitle("Diet Density Plot from Number Distribution"))
```

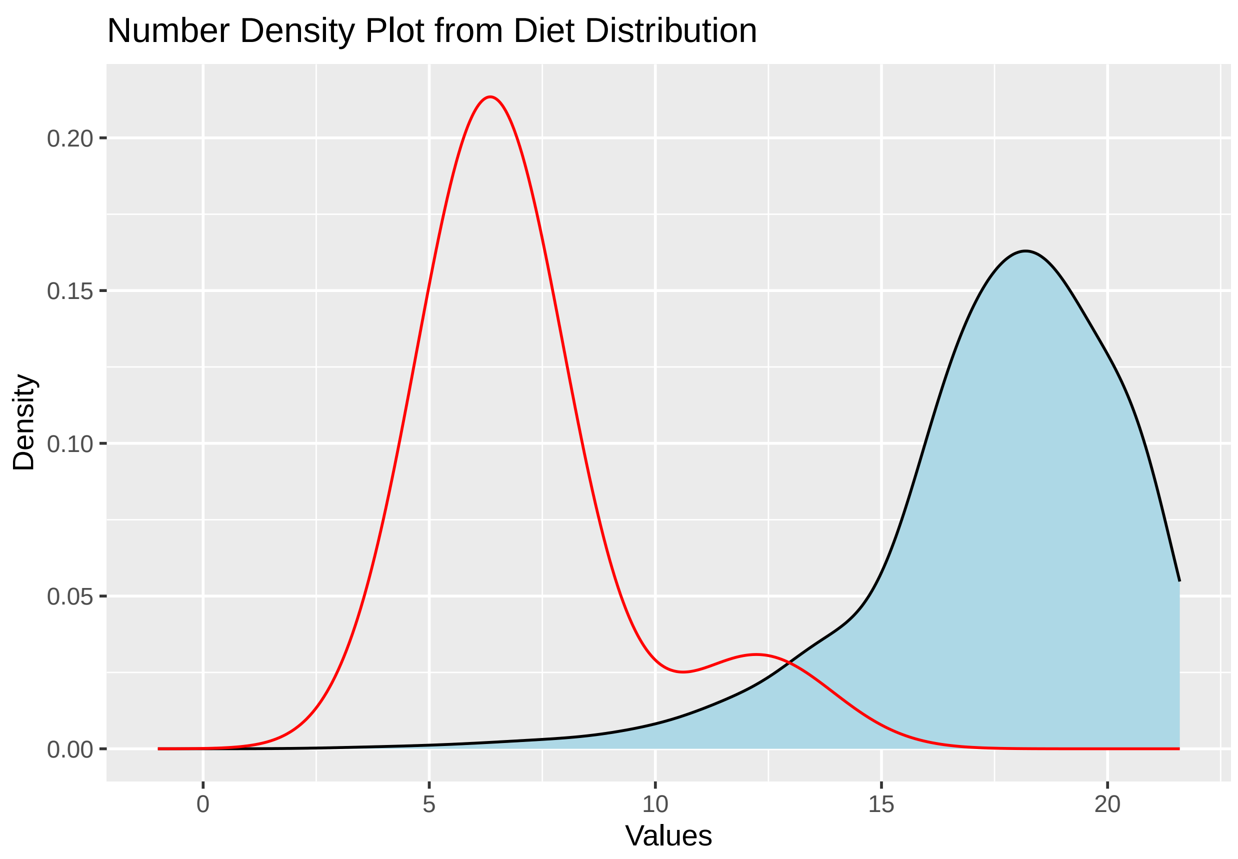


This works, the means have shifted, but its not the best fit, it is not terrible however. Next I will try fit to the biomass distribution first then see if it can be shifted to the number density.

```
#set the correct weighting. and getting the biomass PDF
clusterdata <- repeat_elements(stomach$l, stomach$Nprey*stomach$wprey^(2/3))

dplot <- data.frame(x=biomassgmm[["data"]], density=biomassgmm[["density"]])

#here is the plot of it
(biofit<-ggplot() +
    geom_density(data=stomach, aes(l, weight=weight_biomass), fill="lightblue")+
  geom_line(data=dplot, aes(x = x, y = density), color="red") +
  labs(x = "Values", y = "Density") +
  ggtitle("Diet Density Plot from Diet Distribution"))
```



```
#now going to transform by e^l

shifted_fit <- biomassgmm
shifted_fit[["parameters"]][["mean"]] <- shifted_fit[["parameters"]][["mean"]]+(2/3)*shifted_fit[["paran

#generating the density values
shifted_pdf <- sapply(x_vals, function(x) {
  sum(shifted_fit$parameters$pro * dnorm(x, mean = shifted_fit$parameters$mean, sd = sqrt(shifted_fit$pa
})

#this next code works if you dont change the mean parameter
#shifted_pdf <- shifted_pdf * exp((2/3)*x_vals)
#Normalize the transformed density
#integral_shifted_pdf <- sum(shifted_pdf) * (x_vals[2] - x_vals[1])
#shifted_pdf_normalized <- shifted_pdf / integral_shifted_pdf

# Create a data frame for plotting
plot_data <- data.frame(x = x_vals, shifted_pdf = shifted_pdf)

(biofitnum <- ggplot() +
    geom_density(data=stomach, aes(l, weight=weight_numbers), fill="lightblue")+
  geom_line(data=plot_data, aes(x = x_vals, y = shifted_pdf), color="red") +
  labs(x = "Values", y = "Density") +
  ggtitle("Number Density Plot from Diet Distribution"))
```



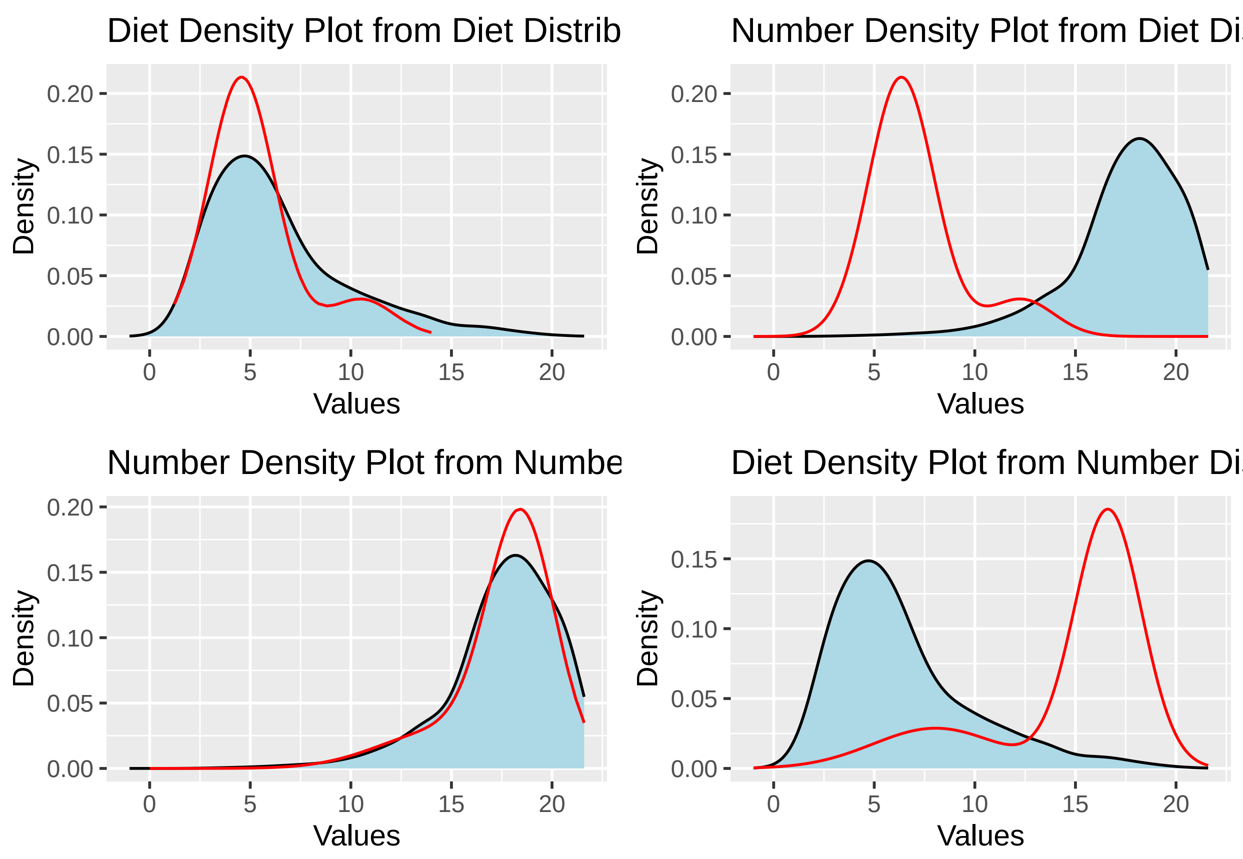
This fit is quite good. So the fit to the diet distribution also works for the number distribution.

Now I am just going to plot both fits side by side to observe

```r
library(cowplot)
```

```
## Warning: package 'cowplot' was built under R version 4.4.1
```

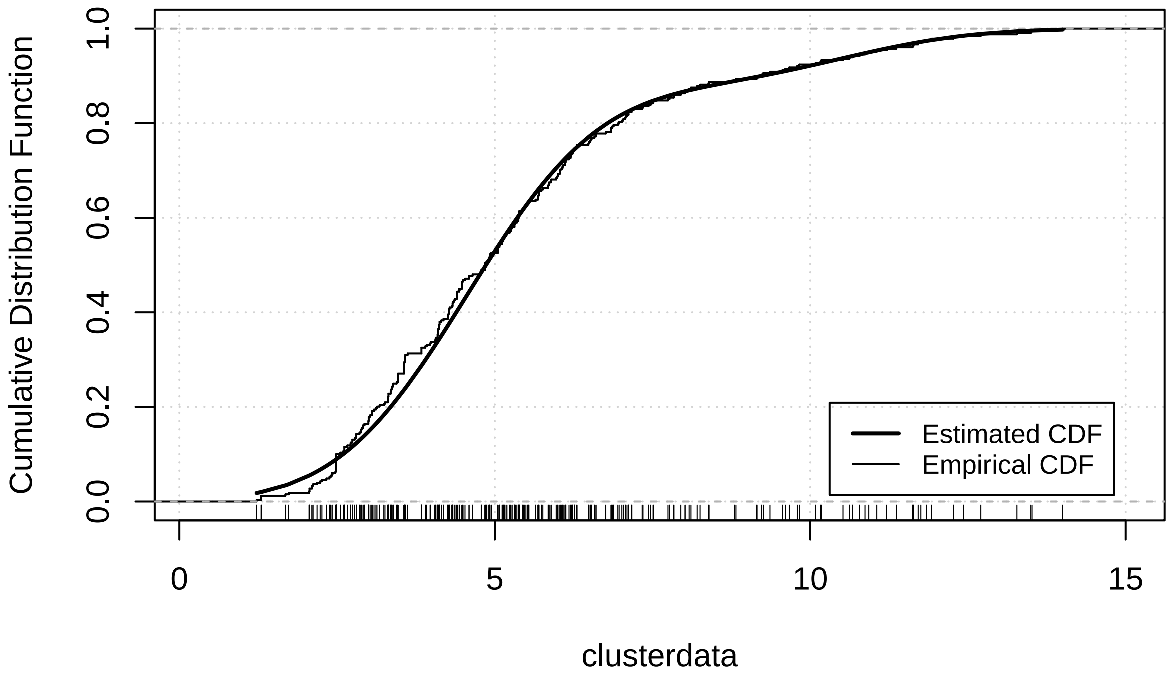
```r
plot_grid(biofit,biofitnum,numbfit,numfitbio)
```

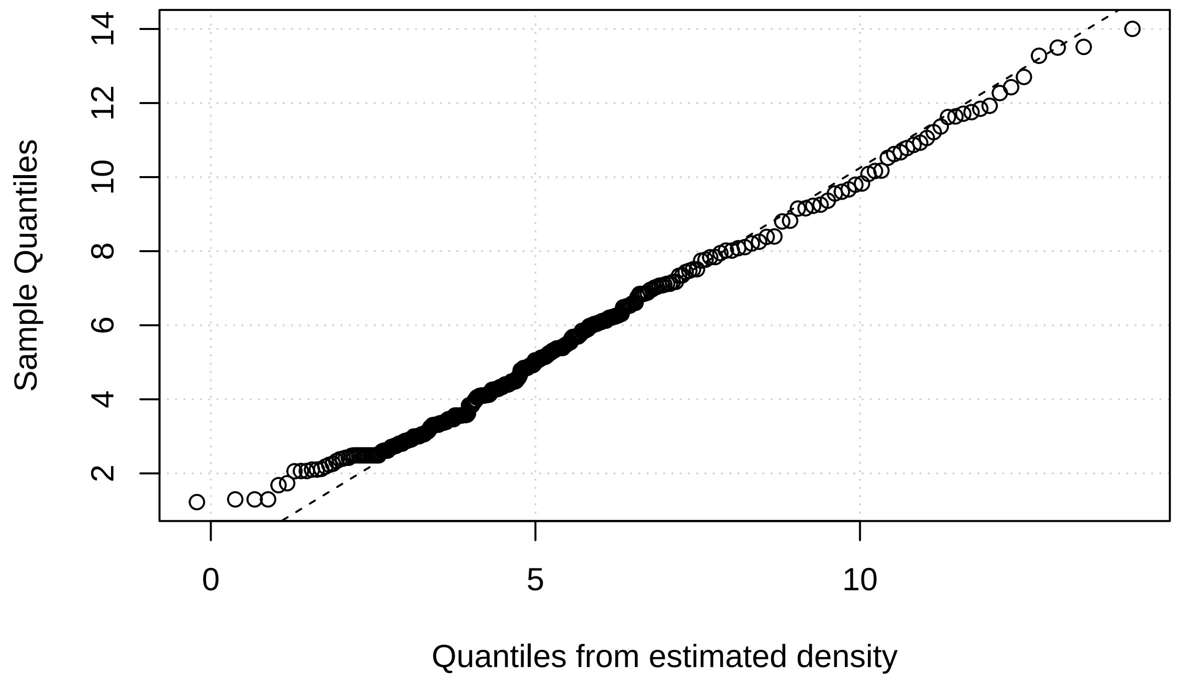


So, if this code is correct (I hope so), the exponential distribution and the mixture gaussian model distribution are both good, so I am not sure which one to use. I am next going to run some tests to see which is actually the better fit of the data.

This is for the GMM biomass - a cumulative distribution function.

```
plot(biomassgmm, what = "diagnostic")
```


Cumulative Distribution Function
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
clusterdata
Estimated CDF
Empirical CDF

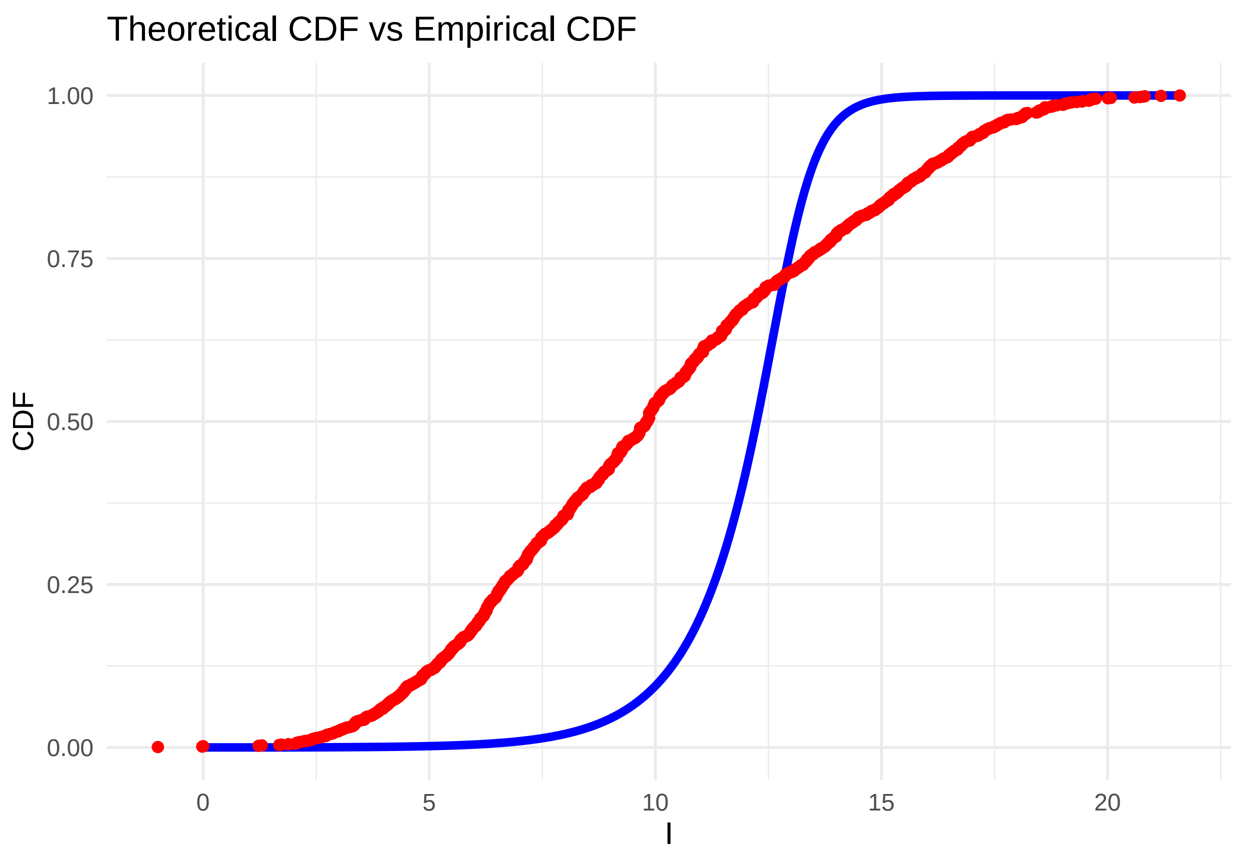


Now I will try to fit one for the truncated exponential distribution

```
#est@fullcoef
#     alpha          ll          ul          lr          ur
# 0.7571769 -1.3754708  4.9093020 12.9416283  2.7145160


cdf_dtexp <- function(l, alpha, ll, ul, lr, ur) {
  integrate(dtexp, lower = 0, upper = l, alpha = alpha, ll = ll, ul = ul, lr = lr, ur = ur)$value
}

alpha <- 0.7571769  # Example parameter
ll <- 1.3754708     # Lower limit for left truncation effect
ul <- 4.9093020     # Upper limit for left truncation effect
lr <- 12.9416283     # Lower limit for right truncation effect
ur <- 2.7145160     # Upper limit for right truncation effect
l_vals <- seq(0, max(stomach$l), length.out = 1000)

# Compute the CDF values
cdf_vals <- sapply(l_vals, cdf_dtexp, alpha = alpha, ll = ll, ul = ul, lr = lr, ur = ur)

cdf_data <- data.frame(l = l_vals, CDF = cdf_vals)

# Sample data points (replace with your actual data)
data_points <- stomach$l
```

```r
# Compute empirical CDF
ecdf_data <- ecdf(data_points)

# Create a data frame for the empirical CDF
empirical_cdf_data <- data.frame(l = sort(data_points), ECDF = ecdf_data(sort(data_points)))

# Plot both CDFs using ggplot2
ggplot() +
  geom_line(data = cdf_data, aes(x = l, y = CDF), color = "blue", size = 1.5) +
  geom_point(data = empirical_cdf_data, aes(x = l, y = ECDF), color = "red", size = 1.5) +
  labs(title = "Theoretical CDF vs Empirical CDF",
       x = "l",
       y = "CDF") +
  theme_minimal()
```

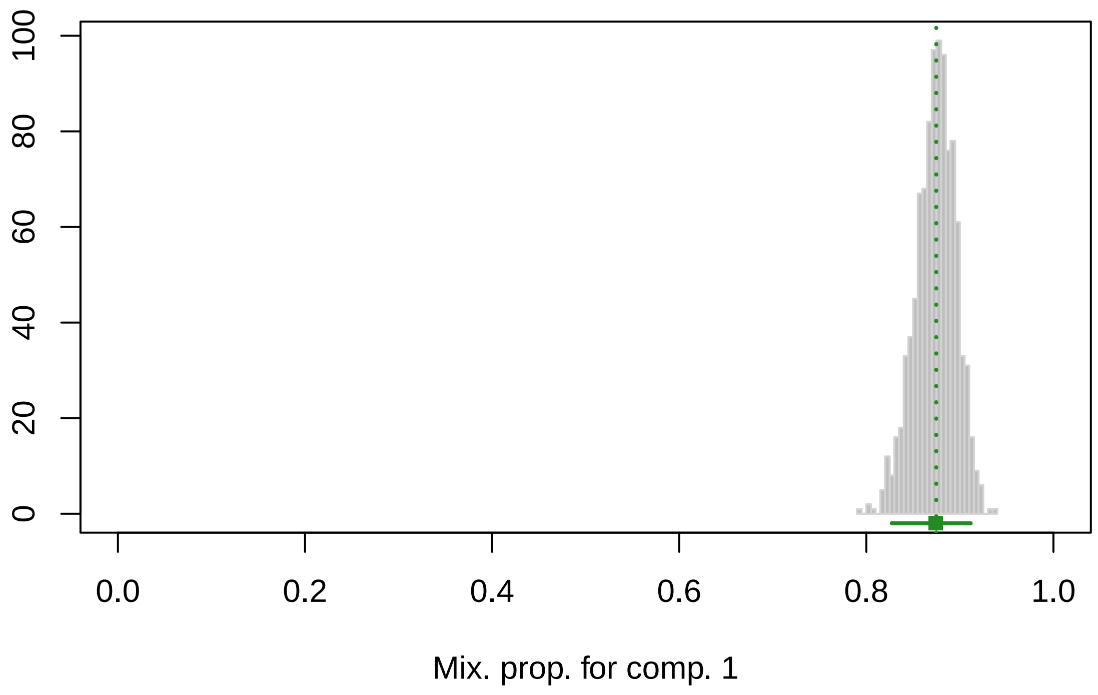
```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

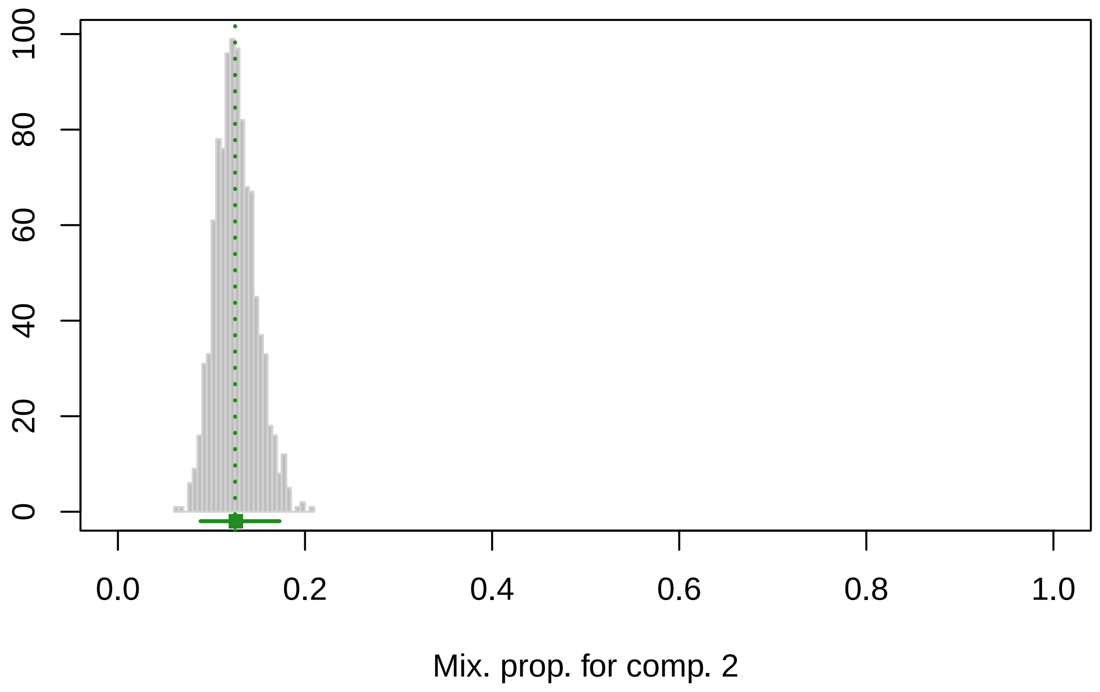


This doesnt seem right, but I also do not know how to produce a CDF plot, so it probably isn’t.

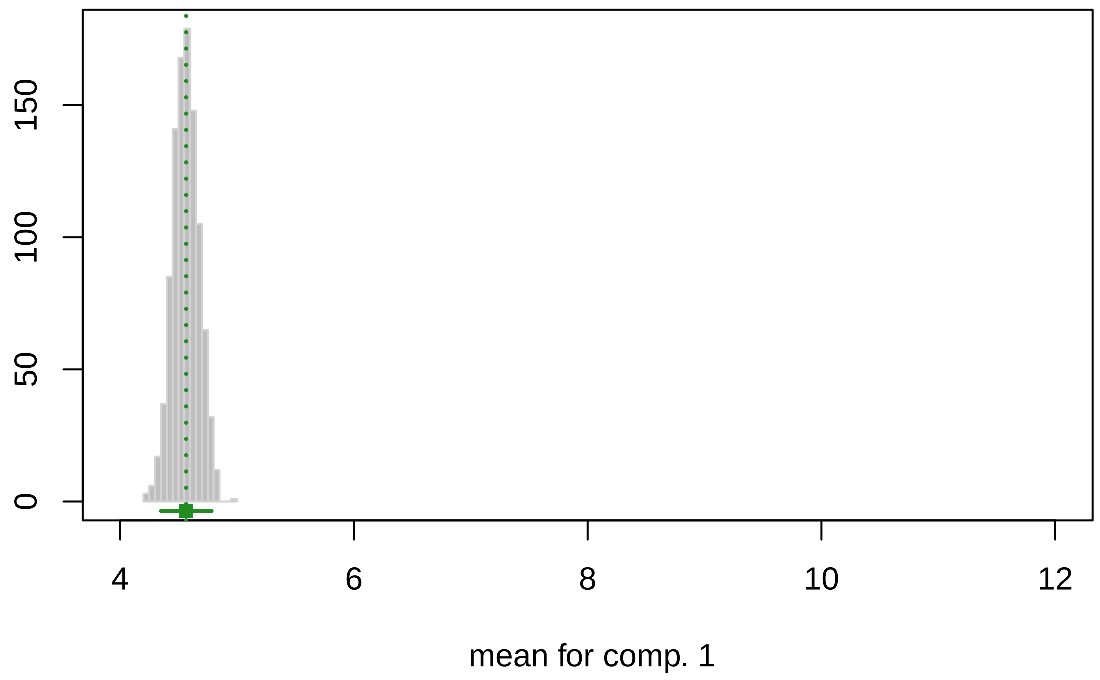
This is checking the fit of the GMM - It looks good.

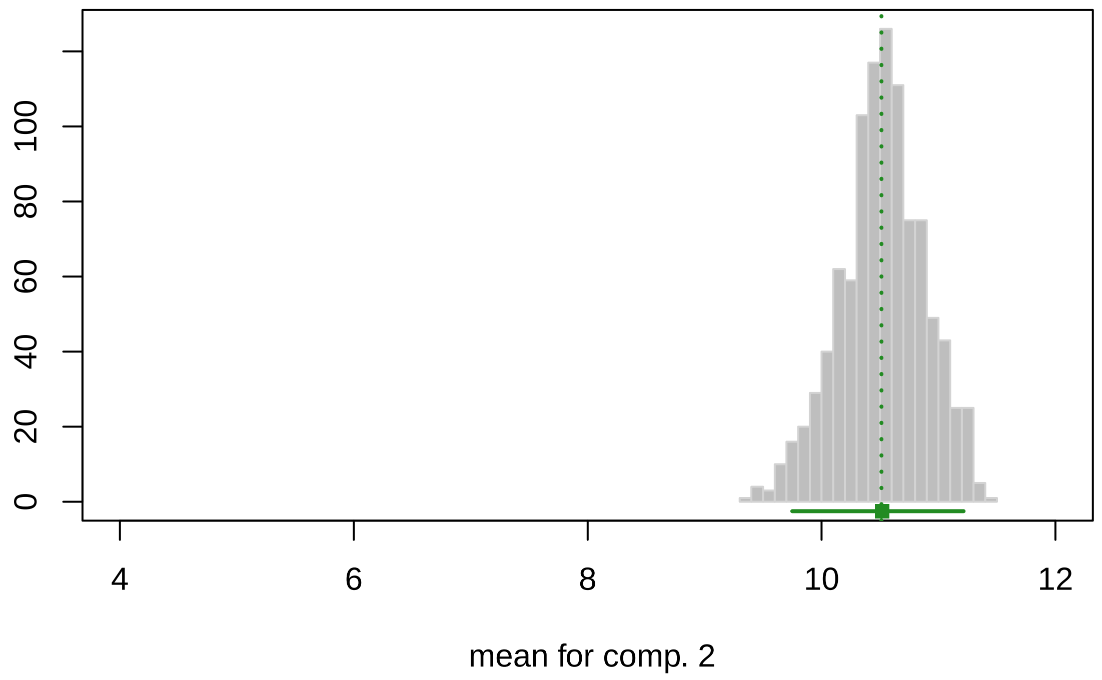
```r
boot4 <- MclustBootstrap(biomassgmm, nboot = 999, type = "bs")
plot(boot4, what = "pro")
```





```r
plot(boot4, what = "mean")
```


0
50
100
150
4
6
8
10
12
mean for comp. 1



```r
summary(boot4)
```

```
## ----------------------------------------------------------
## Resampling standard errors
## ----------------------------------------------------------
## Model                      = E
## Num. of mixture components = 2
## Replications               = 999
## Type                       = nonparametric bootstrap
##
## Mixing probabilities:
##          1          2
## 0.02127925 0.02127925
##
## Means:
##         1         2
## 0.1099810 0.3624122
##
## Variances:
##         1         2
## 0.2123943 0.2123943
```