Body Condition of Mother and Calf Humpback Whales (*Megoptera novaeangiliae*) Over Temporal Scales

Lewis Evans, Sophie Paradis, Maya Otsu, and Alyssa Veneri

2023-12-12

```r
tinytex::install_tinytex(force = TRUE)
```

```
## tlmgr --repository http://www.preining.info/tlgpg/ install tlgpg

## tlmgr option repository 'https://mirrors.mit.edu/CTAN/systems/texlive/tlnet'

## tlmgr update --list
```

```r
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
```

```r
library(tidyr)
library(ggplot2)
library(forcats)
```

Research Questions

1. How does the mother and calf body condition index vary by year?
2. How does the mother and calf body condition index vary seasonally?

Data Wrangling

Below is the code used to optimize and clean the dataset used to answer the research questions. ## Install packages and organise sheets to be filled

```r
# template to be filled
temp<-read.csv("lewis_temp.csv")
# data from master sheet to be transferred
data<-read.csv("Lewis_Raw_Sheet.csv")
# Pulling corrected altitude from master database to analysis data sheet
nm <- c("Corrected_height_m",   "Pixel_length")
temp[nm] <- lapply(nm, function(x) data[[x]][match(temp$Filename,data$Filename)])
View(temp)
#write new data back to original temp csv file.
write.csv(temp,"lewis_temp.csv")
```

preparing to calculate body volumes for each row, need to clean data first:

```r
data<-read.csv("Lewis_Raw_Sheet.csv")
head(data,20)
```

```
##    Region Year DOY Grade_pass Altitude_Source         Filename     Role
## 1      HI 2019  15          N           S_Alt 2019.01.15.03.01   Mother
## 2      HI 2019  15          Y           S_Alt 2019.01.15.03.02     Calf
## 3      HI 2019  15          N           S_Alt 2019.01.15.06.02   Mother
## 4      HI 2019  15          N           S_Alt 2019.01.15.08.01     Calf
## 5      HI 2019  16          N           S_Alt 2019.01.16.19.01   Mother
## 6      HI 2019  16          N           S_Alt 2019.01.16.19.02     Calf
## 7      HI 2019  16          Y           S_Alt 2019.01.16.20.01   Mother
## 8      HI 2019  16          Y           S_Alt 2019.01.16.20.02     Calf
## 9      HI 2019  16          Y           S_Alt 2019.01.16.28.03   Mother
## 10     HI 2019  16          Y           S_Alt 2019.01.16.28.04 Yearling
## 11     HI 2019  16          N           S_Alt 2019.01.16.29.01   Mother
## 12     HI 2019  16          N           S_Alt 2019.01.16.29.02     Calf
## 13     HI 2019  16          Y           S_Alt 2019.01.16.31.01   Mother
## 14     HI 2019  16          Y           S_Alt 2019.01.16.31.02     Calf
## 15     HI 2019  18          Y           S_Alt 2019.01.18.37.01   Mother
## 16     HI 2019  18          Y           S_Alt 2019.01.18.37.02     Calf
## 17     HI 2019  18          Y           S_Alt 2019.01.18.39.01   Mother
## 18     HI 2019  18          Y           S_Alt 2019.01.18.39.02     Calf
## 19     HI 2019  18          Y           S_Alt 2019.01.18.46.01   Mother
## 20     HI 2019  18          Y           S_Alt 2019.01.18.47.01     Calf
##    Corrected_height_m Pixel_length Pixel_Width_5 Pixel_Width_10 Pixel_Width_15
## 1         32.97236219    0.3491373      9.214691       14.08607       16.75829
## 2         29.66091393    0.1469100      9.168017       11.96253       14.89135
## 3          35.1236886    0.3252343      9.235541       13.33296       15.24165
## 4          33.1213199    0.1727700      8.592942       11.40709       13.43479
## 5         28.41806378    0.4404700      7.601259       12.22899       13.32856
## 6          28.3481908    0.1854800      8.151804       10.96770       12.79753
## 7         26.92526647    0.4740500      7.264967       11.78561       13.53775
## 8         27.23761989    0.1965600      7.527777       10.60096       12.77157
## 9         44.31016642    0.2596200      7.488021       11.46235       12.95918
## 10        40.52863803    0.2199700      7.357397       11.52921       13.42690
## 11        35.51454667    0.3499500      7.934658       11.76965       11.13231
## 12        35.44183216    0.1430300      9.196329       12.17870       13.53997
## 13        33.51786646    0.4236100      7.551070       11.54780       13.30373
## 14        32.12062813    0.1519900      6.906498       10.29645       13.14793
## 15        29.82653577    0.4328000      7.773092       11.93179       13.77891
## 16        30.91753566    0.1587100      7.774773       10.95052       12.63967
## 17        28.95496055    0.3999100      6.978168       10.54792       12.86602
## 18        30.95948877    0.1614000      7.535857       11.08413       12.26405
## 19        29.72669054    0.4367700      7.161791       10.72130       12.92166
## 20        33.40264928    0.1454500      9.667448       11.36653       13.47669
##    Pixel_Width_20 Pixel_Width_25 Pixel_Width_30 Pixel_Width_35 Pixel_Width_40
## 1        17.60272       18.97778       21.82566       21.32035       23.46663
## 2        19.27598       19.39924       20.14465       20.47707       20.60363
## 3        16.46303       20.91649       21.71981       22.57257       24.47837
## 4        17.99478       19.00087       20.61068       20.94254       20.53144
## 5        14.26289       17.53144       16.76094       17.62161       19.22495
## 6        17.40439       17.49475       17.82366       18.11683       18.59816
## 7        13.92794       17.20895       16.92749       17.47834       19.05466
## 8        17.10737       18.06163       18.82385       19.20491       18.99115
## 9        13.83876       17.44389       17.51684       17.76539       17.58225
## 10       15.19218       18.65696       19.46731       20.66172       21.92376
## 11       13.82885       17.48153       17.35476       17.63721       18.73034
## 12       17.00266       18.71652       20.14160       20.31278       19.31423
## 13       13.88924       16.89439       18.24686       18.37969       19.58700
## 14       16.11030       17.49982       18.51418       18.20086       17.98907
## 15       14.33013       17.92958       17.48693       17.90626       19.60360
## 16       17.10741       16.43856       18.56772       19.05737       18.97150
## 17       13.83772       17.00028       16.70581       16.42969       17.36708
## 18       16.50976       17.12640       18.65631       18.95578       19.53071
## 19       14.03786       17.13754       17.11963       17.11527       18.27278
## 20       18.04932       17.90279       19.60061       20.06176       19.21272
##    Pixel_Width_45 Pixel_Width_50 Pixel_Width_55 Pixel_Width_60 Pixel_Width_65
## 1        23.48878       23.24622       22.05670       20.48338       17.15620
## 2        20.52011       19.38956       17.30192       14.44496       11.64315
## 3        24.76568       23.72228       22.25214       19.60454       16.97563
## 4        20.42106       19.39355       18.11501       16.06948       13.12512
## 5        19.39205       19.03363       17.85561       16.04386       14.20165
## 6        18.36427       17.15453       15.43386       13.45949       11.13915
## 7        19.37626       19.05142       17.65981       16.34621       14.01598
## 8        18.83388       18.33518       17.08391       14.32996       12.05849
## 9        17.69819       17.55115       16.09979       14.18050       12.92683
## 10       22.04131       21.78012       19.30108       17.82342       16.05820
## 11       19.43507       18.97326       17.56442       15.76950       13.55491
## 12       18.70602       17.53451       16.28507       14.67773       13.02331
## 13       20.00920       20.23003       19.68546       18.35524       16.11702
## 14       17.63002       16.79251       15.70816       13.79602       11.28902
## 15       20.32834       20.06035       18.40953       16.57174       13.94073
## 16       18.21947       16.51763       14.11057       11.70644       10.15790
## 17       17.87276       17.38125       16.32448       14.47927       12.26478
## 18       18.80565       18.42439       15.99676       13.62290       11.30035
## 19       18.78152       18.47390       16.82046       15.25812       13.12270
## 20       19.19343       18.78801       16.40225       13.77020       11.13886
##    Pixel_Width_70 Pixel_Width_75 Pixel_Width_80 Pixel_Width_85 Pixel_Width_90
## 1       13.368451       9.313393       5.996423       4.145080       6.428957
## 2        9.400436       7.252516       5.628185       4.630932       7.018416
## 3       13.265143       9.806319       6.205202       3.880260       3.781384
## 4       10.408040       8.035108       5.839893       5.002380       7.944888
## 5      11.511003       7.885681       5.245174       3.469279       3.075581
## 6       9.107958       6.044240       3.718700       2.812207       7.884673
## 7      11.400634       8.498591       5.396030       3.718112       2.601100
## 8       9.445162       7.078139       4.690090       3.899297       7.493124
## 9      10.534293       8.485563       5.250565       3.108060       2.805712
## 10     12.431880       9.600389       6.004028       2.826976       4.630283
## 11     10.928039       7.677108       5.489817       3.713897       2.502679
## 12     10.196921       7.472869       4.762985       3.779840       9.420200
## 13     13.071048       9.824728       6.404522       3.051423       2.151714
## 14      9.349803       7.899898       6.515875       5.011698       6.906907
## 15     10.340296       7.439381       5.763816       3.246921       3.712072
## 16      8.412008       7.032644       5.787230       4.361447       5.207700
## 17     10.318540       7.615860       5.193292       3.288621       2.651242
## 18      8.461549       7.079675       5.254443       4.514248       5.923629
## 19     10.391854       7.994633       5.271805       3.200238       2.183869
## 20      9.179532       8.096080       5.616791       5.176974       7.508265
##    Pixel_Width_95
## 1       26.440150
## 2       19.082293
## 3       12.272840
## 4       18.171346
## 5       11.559919
## 6       17.642882
## 7       10.445093
## 8       19.441146
## 9       12.048172
## 10      17.471110
## 11      10.486013
## 12      18.765224
## 13      11.537238
## 14      19.155079
## 15       7.513404
## 16      16.646037
## 17      11.754325
## 18      18.803778
## 19      10.114084
## 20      18.330342
```

```r
dim(data)
```

```
## [1] 2686   28
```

```r
#remove rows containg NA value
data <- data %>% drop_na()
dim(data)
```

```
## [1] 2671   28
```

```r
# Only keep rows containing S_Alt altitude source
data<-subset(data, Altitude_Source == 'S_Alt')
dim(data)
```

```
## [1] 2446   28
```

```
#Remove any measuremetns that didnt pass grading (N)
data<-subset(data, Grade_pass != 'N')
dim(data)
```

```
## [1] 2166   28
```

```
#Check value class and change to numeric
class(data$Corrected_height_m)
```

```
## [1] "character"
```

```
data$Corrected_height_m <- as.numeric(data$Corrected_height_m)
class(data$Pixel_length)
```

```
## [1] "numeric"
```

Calculate Total length of whale and length of 5% intervals

```
# Calculate TL
data$TL <- data$Corrected_height_m*data$Pixel_length

#check range to see for anomolies
range(data$TL)
```

```
## [1]  3.753996 14.787516
```

```
#equation to change 5% intervals from pixel length to absolute length
data<-data %>%  mutate(across(c("Pixel_Width_5","Pixel_Width_10","Pixel_Width_15","Pixel_Width_20","Pixe
View(data)
```

isolating 5 % intervals

```
colnames(data)[10:28] <- c("Width.5.proc.m","Width.10.proc.m","Width.15.proc.m","Width.20.proc.m","Widt
```

#Renaming TL and Role.

```
data <- data %>%
  rename("Total.length.m" = "TL",
         "Rep.class" = "Role")
```

reorganisng/subsetting and reprganisng columns of interest. changename to BM

```
BM<-as.data.frame(data[,c("Filename","Total.length.m","Rep.class","Width.5.proc.m","Width.10.proc.m","Wi

#confirming widths are numeric
class(BM$Width.5.proc.m)
```

```
## [1] "numeric"
```

choosing 5% intervals from 5%-85% of the whale.

```
HW.ratios<-data.frame(
  Measurement.site=seq(5,85,5),
  All.HW.ratio=c(0.89,0.78,0.90,0.97,0.91,0.92,0.93,0.92,0.94,1.00,1.08,1.19,1.34,1.51,1.75,2.03,2.26))

for(i in 1:length(BM[,1])){                                                                         #Ru
  temp.widths<-BM[i,c(which(colnames(BM)=="Width.5.proc.m"):which(colnames(BM)=="Width.85.proc.m"))]  #
  temp.heights<-temp.widths*HW.ratios$All.HW.ratio                    #...calculate the height of whale
  colnames(temp.heights)<-c("Height.5.proc.m","Height.10.proc.m","Height.15.proc.m","Height.20.proc.m",
  ifelse(i==1,temp.output<-temp.heights,temp.output<-rbind(temp.output,temp.heights))                 #
}
BM<-cbind(BM,temp.output)
```

Set width and height at 0 and 100%BL from rostrum to be 0

```
BM$Width.0.proc.m<-0
BM$Height.0.proc.m<-0
BM$Width.100.proc.m<-0
BM$Height.100.proc.m<-0
```

Calculate the body width and height at 90 and 95%BL from rostrum based on linear interpolation between 85 and 100%BL from rostrum

```
BM$Width.90.proc.m<-BM$Width.85.proc.m-(1*(BM$Width.85.proc.m/3))
BM$Width.95.proc.m<-BM$Width.85.proc.m-(2*(BM$Width.85.proc.m/3))
BM$Height.90.proc.m<-BM$Height.85.proc.m-(1*(BM$Height.85.proc.m/3))
BM$Height.95.proc.m<-BM$Height.85.proc.m-(2*(BM$Height.85.proc.m/3))
```

Re-order data frame (necessary for the next step)

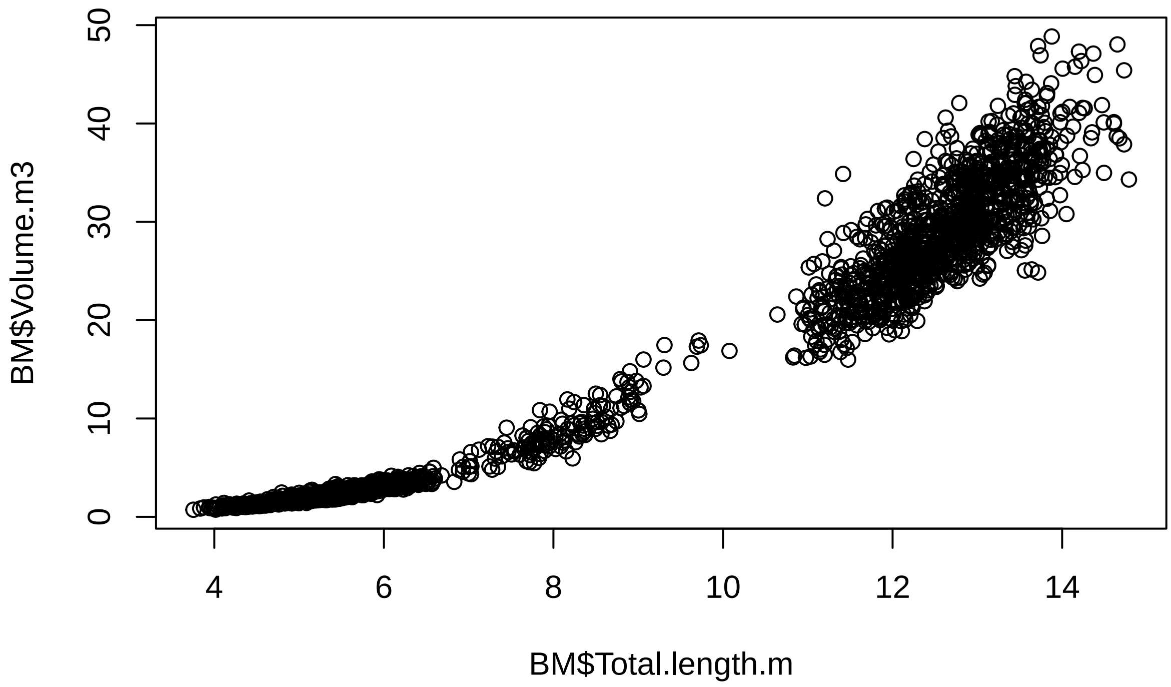
```
BM<-BM[,c(which(colnames(BM)=="Width.0.proc.m"),which(colnames(BM)=="Width.5.proc.m"):which(colnames(BM
Width.col.start<-which(colnames(BM)=="Width.0.proc.m")        #Extracts the column number of the starti
Height.col.start<-which(colnames(BM)=="Height.0.proc.m")      #Extracts the column number of the starti
```

Calculate the body volume of the whales

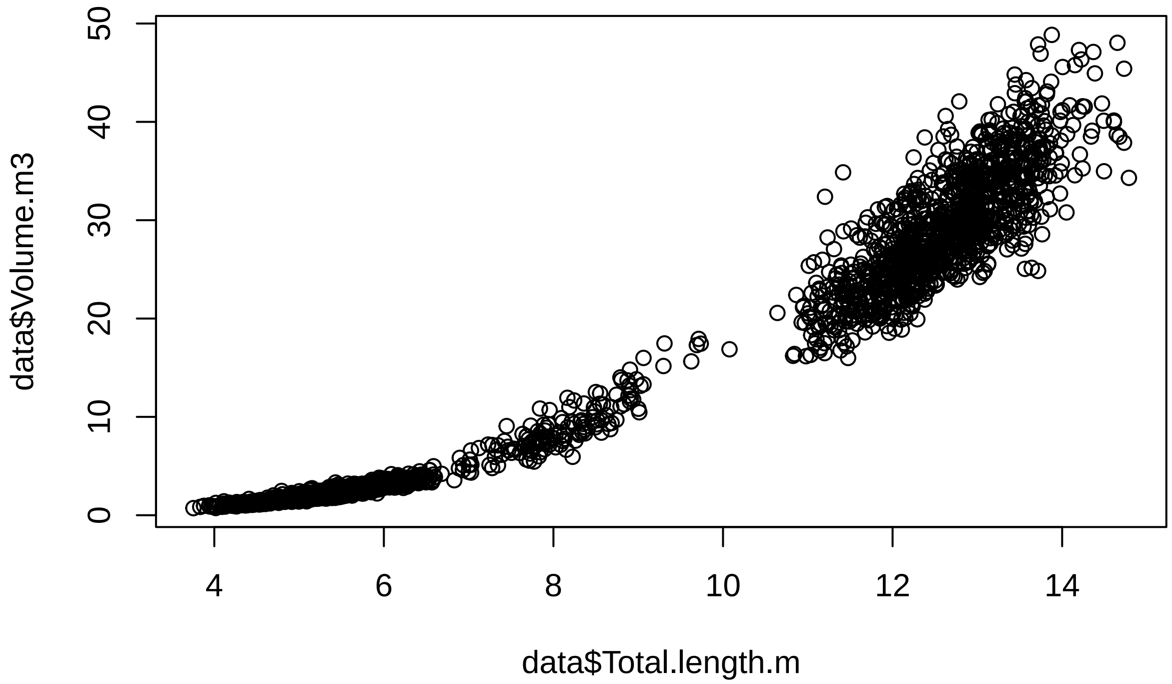
```
BM$Volume.m3<-NA                            #Creates an empty storage vector for body volume
for(y in 1:length(BM[,1])){                 #Runs a loop for every individual in the data frame
  for(k in 1:(length(seq(0,100,5))-1)){     #Runs a loop for every body segment (volume between two mea
    f.ellipse<-function(x){                   #Formula to calculate the volume of an ellipse
      (BM[y,Width.col.start+(k-1)]+((BM[y,(Width.col.start+k)]-BM[y,Width.col.start+(k-1)])*x))/2*(BM[y
    }
    Volume.temp<-integrate(f.ellipse,lower=0,upper=1)$value*BM$Total.length.m[y]*0.05   #Multiplies the
    ifelse(k==1,Store1<-Volume.temp,Store1<-c(Store1,Volume.temp))                          #Stores the out
  }
  BM$Volume.m3[y]<-sum(Store1)                                                               #Calculates the
}
```

check data

```
plot(BM$Total.length.m,BM$Volume.m3)
```



```
#Check data makes sense/that there arent many outliers
data$Volume.m3 <- NA
nm <- c("Volume.m3")
data[nm] <- lapply(nm, function(x) BM[[x]][match(data$Filename,BM$Filename)])
View(data)
plot(data$Total.length.m,data$Volume.m3)
```



```
#write the CSV file
write.csv(data, "Lewis_Raw_Sheet.csv")
```

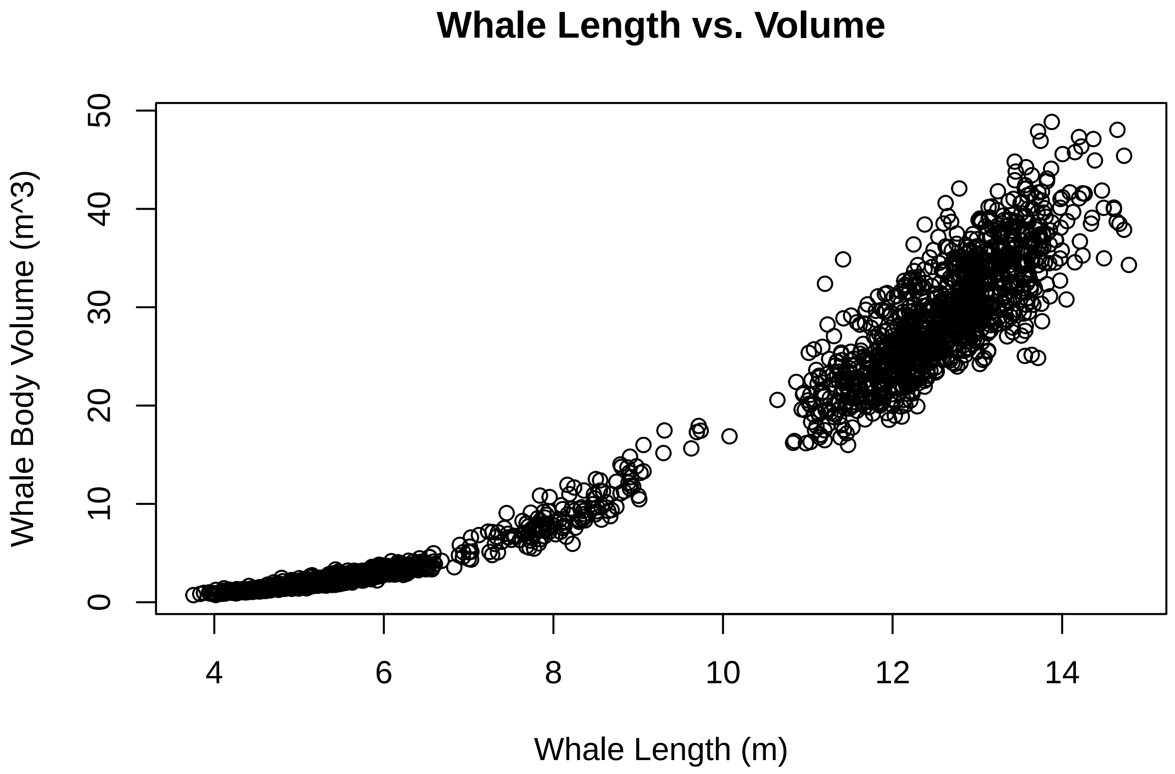
Data Analysis

Below is the code and plots used to analyze our data with the corresponding statistical analysis.

```
###Create regression###
whales <- read.csv("Lewis_Raw_Sheet.csv")
model <- lm(Volume.m3 ~ Total.length.m, data = whales)
model
```

```
##
## Call:
## lm(formula = Volume.m3 ~ Total.length.m, data = whales)
##
## Coefficients:
##    (Intercept)  Total.length.m
##        -18.322           3.771
```

```
whale_plot <- plot(whales$Total.length.m, whales$Volume.m3, main = "Whale Length vs. Volume",
     xlab = "Whale Length (m)", ylab = "Whale Body Volume (m^3)")
```



```r
summary_result <- summary(model)
summary_result
```

```
##
## Call:
## lm(formula = Volume.m3 ~ Total.length.m, data = whales)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.9548 -1.8605 -0.1104  1.6421 14.8411
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -18.32208    0.18839  -97.26   <2e-16 ***
## Total.length.m   3.77054    0.01905  197.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.176 on 2164 degrees of freedom
## Multiple R-squared:  0.9477, Adjusted R-squared:  0.9476
## F-statistic: 3.919e+04 on 1 and 2164 DF,  p-value: < 2.2e-16
```

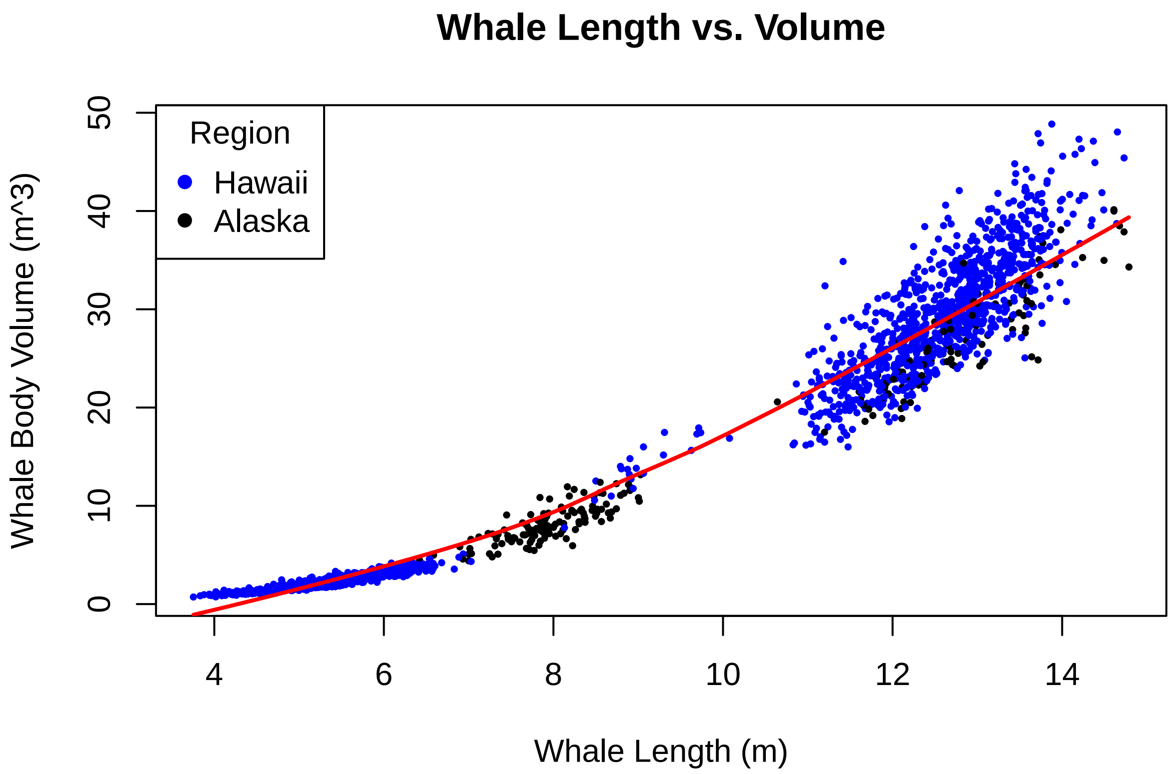
```r
r_squared <- summary_result$r.squared
r_squared
```

```
## [1] 0.9476658
```

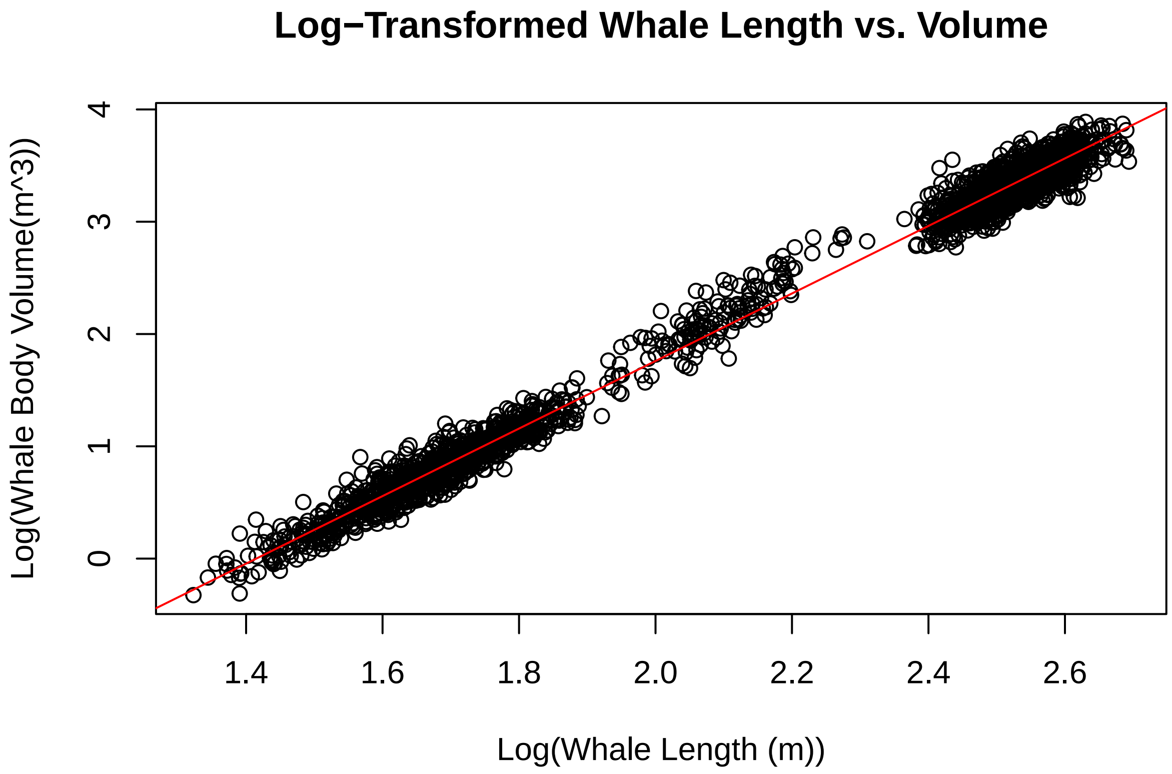
```
# Plot with points of different colors based on Region
colors <- ifelse(whales$Region == "HI", "blue", "black")
whale_plot <- plot(
  whales$Total.length.m, whales$Volume.m3,
  main = "Whale Length vs. Volume",
  xlab = "Whale Length (m)", ylab = "Whale Body Volume (m^3)",
  col = colors, pch = 16, cex = 0.5
)
whale_plot
```

```
## NULL
```

```
smoothed_data <- lowess(whales$Total.length.m, whales$Volume.m3)
lines(smoothed_data, col = "red", lwd = 2)
legend("topleft", legend = c("Hawaii", "Alaska"), col = c("blue", "black"), pch = 16, cex = 1, title =
```



```
###Transform Data###
whales$log_length <- log(whales$Total.length.m)
whales$log_volume <- log(whales$Volume.m3)
log_regression <- lm(log_volume ~ log_length, data = whales)
plot(whales$log_length, whales$log_volume, main = "Log-Transformed Whale Length vs. Volume",
     xlab = "Log(Whale Length (m))", ylab = "Log(Whale Body Volume(m^3))")
abline(log_regression, col = "red")
```



```
summary_result_log <- summary(log_regression)
summary_result_log
```

```
##
## Call:
## lm(formula = log_volume ~ log_length, data = whales)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.40714 -0.08384 -0.00541  0.07253  0.48363
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.254489   0.013381  -317.9   <2e-16 ***
## log_length   3.007065   0.006145   489.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1216 on 2164 degrees of freedom
## Multiple R-squared:  0.991,  Adjusted R-squared:  0.991
## F-statistic: 2.395e+05 on 1 and 2164 DF,  p-value: < 2.2e-16
```

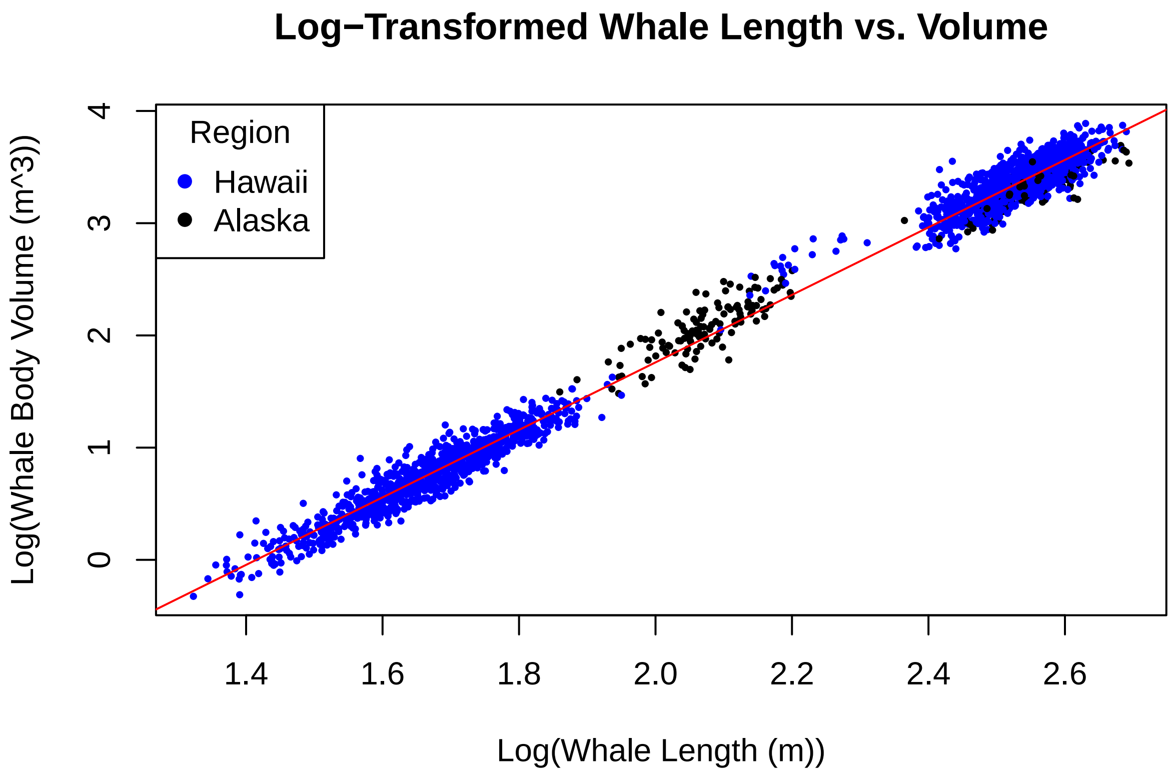
```
r_squared_log <- summary_result_log$r.squared
r_squared_log
```

```
## [1] 0.991045
```

```
# Plot with points of different colors based on Region
colors <- ifelse(whales$Region == "HI", "blue", "black")
whale_plot <- plot(
  whales$log_length, whales$log_volume,
  main = "Log-Transformed Whale Length vs. Volume",
  xlab = "Log(Whale Length (m))", ylab = "Log(Whale Body Volume (m^3))",
  col = colors, pch = 16, cex = 0.5
)
whale_plot
```

```
## NULL
```

```
legend("topleft", legend = c("Hawaii", "Alaska"), col = c("blue", "black"), pch = 16, cex = 1, title =
abline(log_regression, col = "red")
```



```
summary_result_log <- summary(log_regression)
summary_result_log
```

```
##
## Call:
## lm(formula = log_volume ~ log_length, data = whales)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.40714 -0.08384 -0.00541  0.07253  0.48363
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.254489   0.013381  -317.9   <2e-16 ***
## log_length   3.007065   0.006145   489.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1216 on 2164 degrees of freedom
## Multiple R-squared:  0.991,  Adjusted R-squared:  0.991
## F-statistic: 2.395e+05 on 1 and 2164 DF,  p-value: < 2.2e-16
```

```
r_squared_log <- summary_result_log$r.squared
r_squared_log
```

```
## [1] 0.991045
```

```
###Calculating residuals###
residuals_vector <- residuals(log_regression)
head(residuals_vector)
```

```
##            1            2            3            4            5            6
##  0.117148607  0.006315415 -0.018770543 -0.097434808  0.219220464  0.133003399
```

```
residual_variance <- var(residuals_vector)
head(residual_variance)
```

```
## [1] 0.01476808
```

```
whales$BCI <- residuals_vector
head(whales)
```

```
##    X Region Year DOY Grade_pass Altitude_Source         Filename Rep.class
## 1  2     HI 2019  15          Y           S_Alt 2019.01.15.03.02      Calf
## 2  7     HI 2019  16          Y           S_Alt 2019.01.16.20.01    Mother
## 3  8     HI 2019  16          Y           S_Alt 2019.01.16.20.02      Calf
## 4  9     HI 2019  16          Y           S_Alt 2019.01.16.28.03    Mother
## 5 10     HI 2019  16          Y           S_Alt 2019.01.16.28.04  Yearling
## 6 13     HI 2019  16          Y           S_Alt 2019.01.16.31.01    Mother
##   Corrected_height_m Pixel_length Width.5.proc.m Width.10.proc.m
## 1           29.66091      0.14691      0.3994949       0.5212653
## 2           26.92527      0.47405      0.9272948       1.5043068
## 3           27.23762      0.19656      0.4030241       0.5675568
## 4           44.31017      0.25962      0.8614074       1.3186065
## 5           40.52864      0.21997      0.6559182       1.0278384
## 6           33.51787      0.42361      1.0721390       1.6396154
##   Width.15.proc.m Width.20.proc.m Width.25.proc.m Width.30.proc.m
## 1       0.6488881       0.8399478       0.8453190        0.877800
## 2       1.7279477       1.7777515       2.1965369        2.160612
## 3       0.6837677       0.9158989       0.9669881        1.007797
## 4       1.4907987       1.5919845       2.0067111        2.015104
## 5       1.1970195       1.3543958       1.6632835        1.735527
## 6       1.8889308       1.9720644       2.3987504        2.590781
##   Width.35.proc.m Width.40.proc.m Width.45.proc.m Width.50.proc.m
## 1       0.8922854        0.897800       0.8941606       0.8448973
## 2       2.2309215        2.432123       2.4731709       2.4317089
## 3       1.0281975        1.016753       1.0083333       0.9816338
## 4       2.0436958        2.022627       2.0359649       2.0190499
## 5       1.8420096        1.954522       1.9650014       1.9417163
## 6       2.6096411        2.781060       2.8410066       2.8723615
##   Width.55.proc.m Width.60.proc.m Width.65.proc.m Width.70.proc.m
## 1       0.7539285       0.6294370       0.5073486       0.4096226
## 2       2.2540847       2.0864176       1.7889882       1.4551682
## 3       0.9146428       0.7672011       0.6455904       0.5056776
## 4       1.8520885       1.6312966       1.4870776       1.2118446
## 5       1.7207080       1.5889729       1.4316019       1.1083126
## 6       2.7950413       2.6061701       2.2883757       1.8558932
##   Width.75.proc.m Width.80.proc.m Width.85.proc.m Width.90.proc.m
## 1       0.3160273       0.2452473       0.2017922       0.3058264
## 2       1.0847535       0.6887451       0.4745769       0.3320024
## 3       0.3789513       0.2510993       0.2087616       0.4011689
## 4       0.9761627       0.6040148       0.3575452       0.3227636
## 5       0.8558828       0.5352642       0.2520273       0.4127937
## 6       1.3949643       0.9093462       0.4332564       0.3055112
##   Width.95.proc.m Total.length.m Volume.m3 log_length log_volume          BCI
## 1        0.831508       4.357485  1.334750   1.471895  0.2887440  0.117148607
## 2        1.333204      12.763923 30.255767   2.546623  3.4096868  0.006315415
## 3        1.040845       5.353827  2.164153   1.677812  0.7720293 -0.018770543
## 4        1.385998      11.503805 19.952769   2.442678  2.9933679 -0.097434808
## 5        1.557564       8.915085 12.723143   2.187745  2.5434226  0.219220464
## 6        1.638115      14.198503 47.307513   2.653137  3.8566691  0.133003399
```

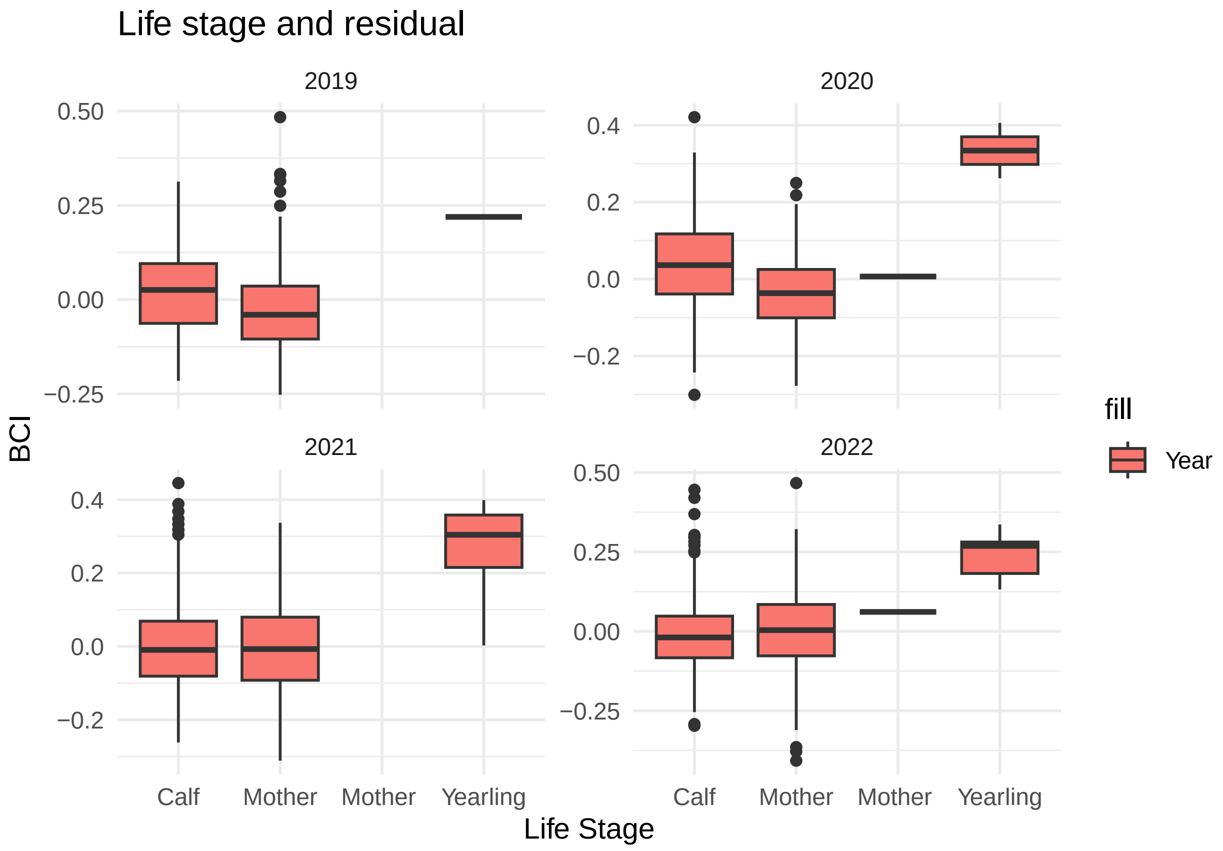
```
###Create month column###
whales <- whales %>%
  mutate(Date = as.Date(paste(Year, DOY, sep = "-"), format = "%Y-%j"))
whales <- whales %>%
  mutate(Month = format(Date, "%m"))
avg_bv_per_month <- whales %>%
  group_by(Year, Month, Rep.class) %>%
  summarise(Avg_BV = mean(Volume.m3))
```

```
## ‘summarise()‘ has grouped output by ’Year’, ’Month’. You can override using the
## ‘.groups‘ argument.
```

```
print(avg_bv_per_month)
```

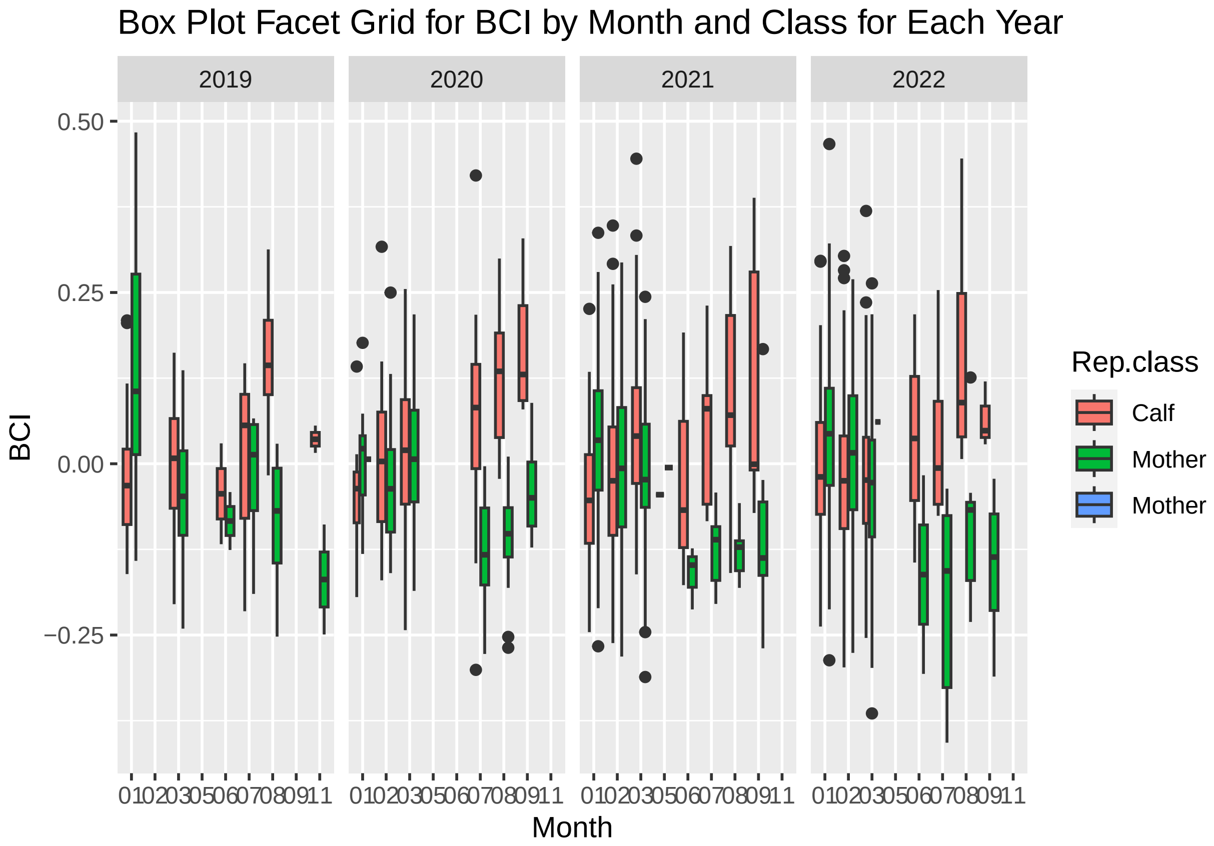
```
## # A tibble: 63 x 4
## # Groups:   Year, Month [27]
##     Year Month Rep.class Avg_BV
##    <int> <chr> <chr>      <dbl>
##  1  2019 01    Calf        1.75
##  2  2019 01    Mother     33.3
##  3  2019 01    Yearling   12.7
##  4  2019 03    Calf        2.10
##  5  2019 03    Mother     24.6
##  6  2019 06    Calf        5.08
##  7  2019 06    Mother     26.6
##  8  2019 07    Calf        7.09
##  9  2019 07    Mother     28.4
## 10  2019 08    Calf        7.59
## # i 53 more rows
```

```
###Create a facet grid###
whale_grid <- ggplot(whales, aes(x = Rep.class, y = BCI, fill = "Year")) +
  geom_boxplot() +
  facet_wrap(~Year, scales = "free_y") +
  labs(title = "Life stage and residual",
       x = "Life Stage",
       y = "BCI") +
  theme_minimal()
whale_grid
```

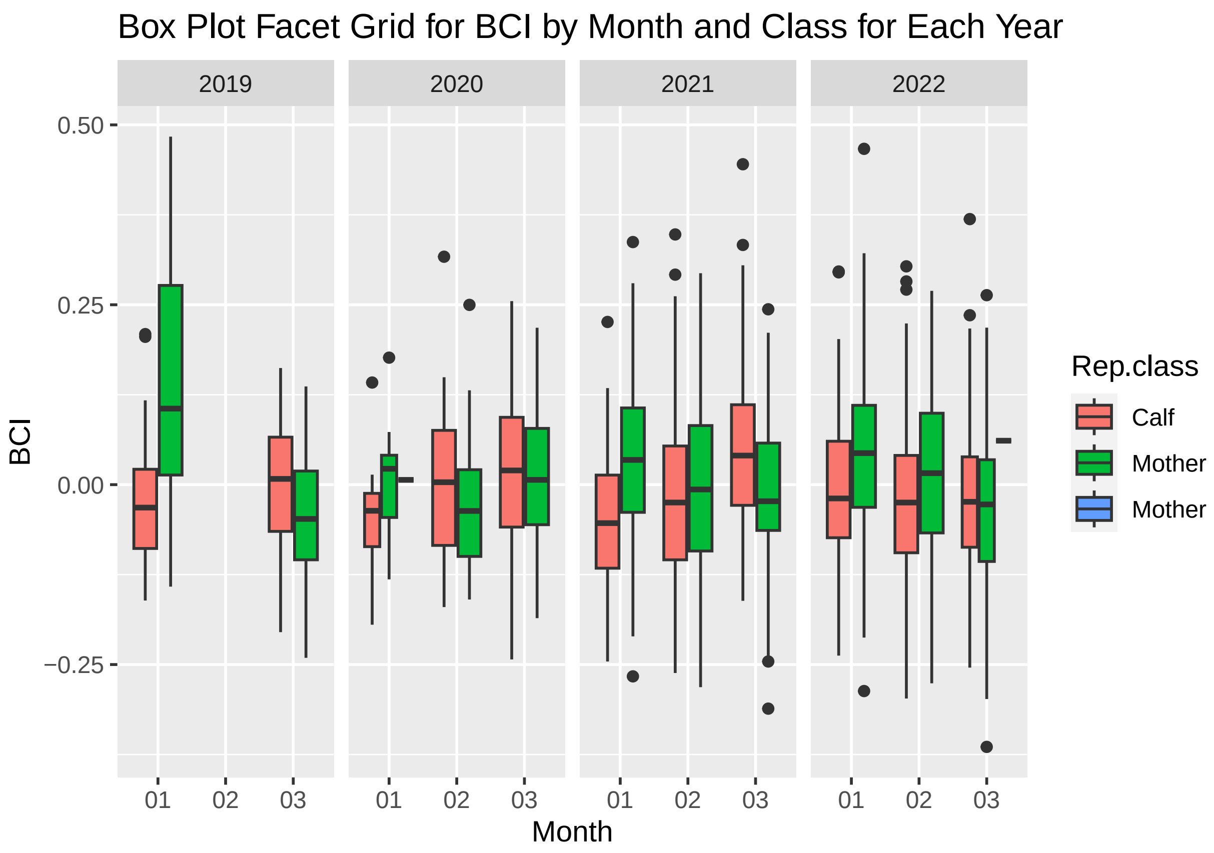


```
###Facet grid with month###
#omit yearling#
whales_filtered1 <- whales %>%
  filter(Rep.class != "Yearling")
```

```
ggplot(whales_filtered1, aes(x = Month, y = BCI, fill = Rep.class)) +
  geom_boxplot() +
  facet_grid(~Year) +
  labs(x = "Month", y = "BCI", fill = "Rep.class") +
  ggtitle("Box Plot Facet Grid for BCI by Month and Class for Each Year")
```



```
###Facet grid with month 1,2, and 3###
whales_subset <- whales %>% filter(Month %in% c("01", "02", "03"))
whales_filtered <- whales_subset %>%
  filter(Rep.class != "Yearling")
ggplot(whales_filtered, aes(x = as.factor(Month), y = BCI, fill = Rep.class)) +
  geom_boxplot() +
  facet_grid(~Year) +
  labs(x = "Month", y = "BCI", fill = "Rep.class") +
  ggtitle("Box Plot Facet Grid for BCI by Month and Class for Each Year")
```



```
###Perform a t-test to compare BCI of calf and mother###
# Exclude 'yearling' class
##################whales_filtered <- whales %>%
whales_filtered <- whales %>%
  filter(Rep.class != "Yearling")

table(whales_filtered$Rep.class)
```

```
##
##    Calf  Mother Mother
##    1046    1097       2
```

```
whales_filtered$Rep.class <- fct_collapse(whales_filtered$Rep.class, Mother = c('Mother', 'Mother '))
table(whales_filtered$Rep.class)
```

```
##
##   Calf Mother
##   1046   1099
```

```
t_test_result <- t.test(BCI ~ Rep.class, data = whales_filtered)
print(t_test_result)
```

```
##
##  Welch Two Sample t-test
```

```
##
## data:  BCI by Rep.class
## t = 1.6244, df = 2141.8, p-value = 0.1044
## alternative hypothesis: true difference in means between group Calf and group Mother is not equal to
## 95 percent confidence interval:
##  -0.001728381  0.018409689
## sample estimates:
##   mean in group Calf mean in group Mother
##          0.001745506         -0.006595148
```

```
#T-test for Hawaii
t_test_hawaii <- t.test(BCI ~ Rep.class, data = whales_filtered[whales_filtered$Region == "HI", ])
print("T-Test for Hawaii:")
```

```
## [1] "T-Test for Hawaii:"
```

```
print(t_test_hawaii)
```

```
##
##  Welch Two Sample t-test
##
## data:  BCI by Rep.class
## t = -3.1041, df = 1903, p-value = 0.001937
## alternative hypothesis: true difference in means between group Calf and group Mother is not equal to
## 95 percent confidence interval:
##  -0.026202972 -0.005912059
## sample estimates:
##   mean in group Calf mean in group Mother
##          -0.01075828           0.00529924
```

```
#T-test for Alaska
t_test_alaska <- t.test(BCI ~ Rep.class, data = whales_filtered[whales_filtered$Region == "AK", ])
print("T-Test for Alaska:")
```

```
## [1] "T-Test for Alaska:"
```

```
print(t_test_alaska)
```

```
##
##  Welch Two Sample t-test
##
## data:  BCI by Rep.class
## t = 13.145, df = 215.44, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group Calf and group Mother is not equal to
## 95 percent confidence interval:
##  0.1724013 0.2332230
## sample estimates:
##   mean in group Calf mean in group Mother
##           0.09733226          -0.10547985
```

```r
###Creating a time series plot###

##TIME SERIES

##First, find the average BV of calves and mothers per month
#convert year and DOY to date format
###make avg per month
whales <- whales %>%
  mutate(Date = as.Date(paste(Year, DOY, sep = "-"), format = "%Y-%j"))

# Extract month from the date
whales <- whales %>%
  mutate(Month = format(Date, "%m"))

# Group by Year, Month, and class, then calculate the average body volume
avg_bv_per_month <- whales %>%
  group_by(Year, Month, Rep.class) %>%
  summarise(Avg_BV = mean(Volume.m3))
```

```
## `summarise()` has grouped output by 'Year', 'Month'. You can override using the
## `.groups` argument.
```

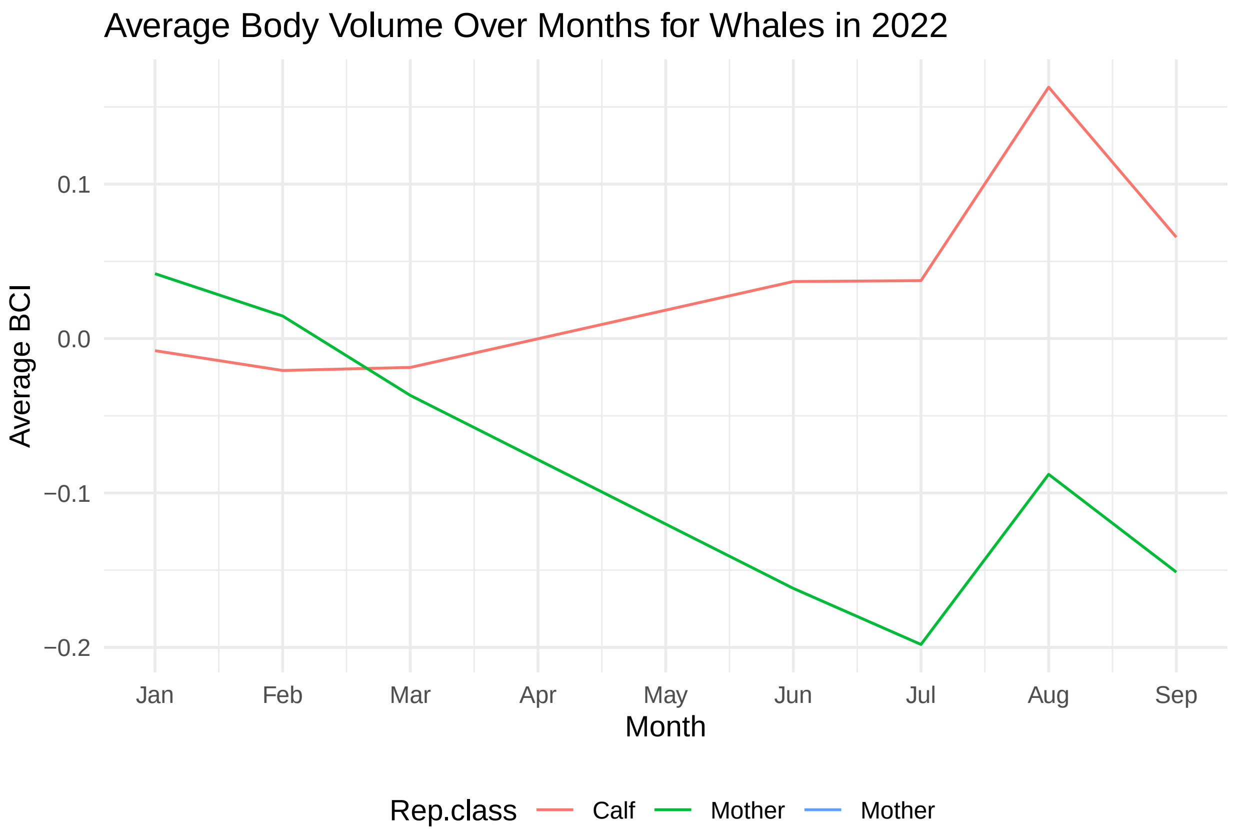
```r
# View the resulting dataset with average body volume per month for each year and Rep.class
print(avg_bv_per_month)
```

```
## # A tibble: 63 x 4
## # Groups:   Year, Month [27]
##     Year Month Rep.class Avg_BV
##    <int> <chr> <chr>      <dbl>
##  1  2019 01    Calf        1.75
##  2  2019 01    Mother     33.3
##  3  2019 01    Yearling   12.7
##  4  2019 03    Calf        2.10
##  5  2019 03    Mother     24.6
##  6  2019 06    Calf        5.08
##  7  2019 06    Mother     26.6
##  8  2019 07    Calf        7.09
##  9  2019 07    Mother     28.4
## 10  2019 08    Calf        7.59
## # i 53 more rows
```

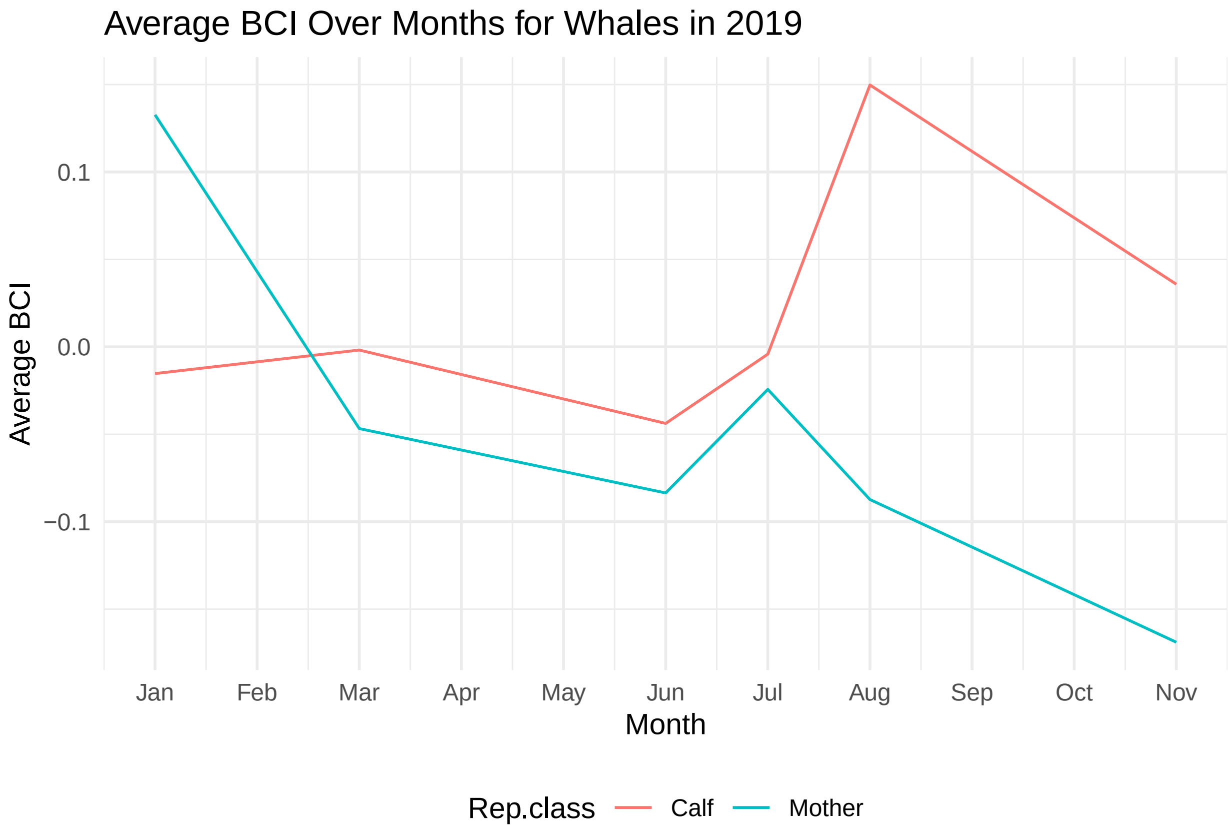
```r
##NEXT, prepare data by excluding yearling class to focus on mothers and calves,
#then group by year, month and class to calculate avg body volume for each year

# Exclude 'yearling' class
whales_filtered <- whales %>%
  filter(Rep.class != "Yearling")

# Group by Year, Month, and Rep.class, then calculate the average BCI
avg_BCI_per_month <- whales_filtered %>%
  group_by(Year, Month, Rep.class) %>%
  summarise(Avg_BCI = mean(BCI))
```

```
## ‘summarise()‘ has grouped output by ’Year’, ’Month’. You can override using the
## ‘.groups‘ argument.
```

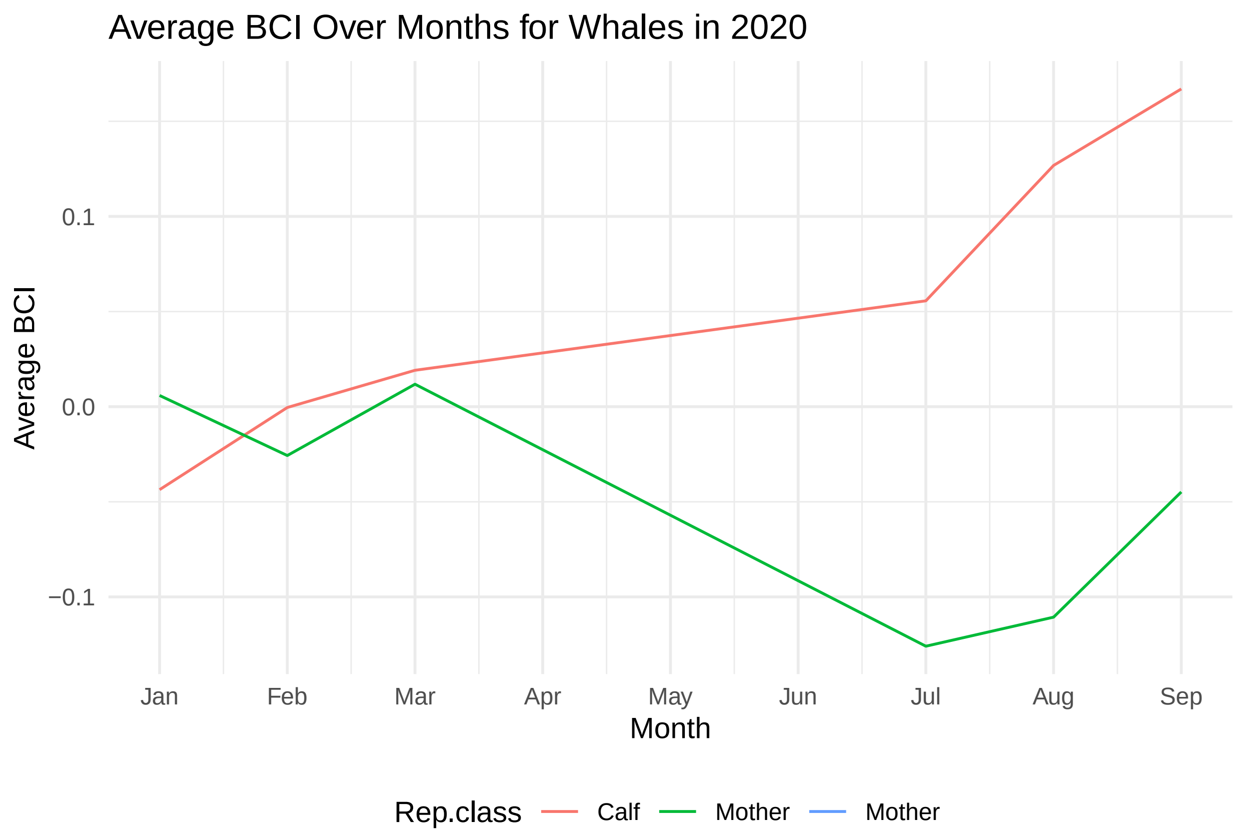
```
# Plotting for each year
for (year in 2019:2022) {
  # Subset data for the current year
  year_data <- avg_BCI_per_month %>%
    filter(Year == year) %>%
    filter(Rep.class != "Yearling")  # Filter out 'yearling' within each year

  # Plotting average body volume over months for each Rep.class
  ggplot(year_data, aes(x = as.numeric(Month), y = Avg_BCI, group = Rep.class, color = Rep.class)) +
    geom_line() +
    labs(title = paste("Average Body Volume Over Months for Whales in", year),
         x = "Month", y = "Average BCI") +
    scale_x_continuous(breaks = 1:12, labels = month.abb) +
    theme_minimal() +
    theme(legend.position = "bottom")
#Doesnt print all years, only 2022. So Next codes will plot each separately

  # You can save the plot here or use any preferred method to display it
  # ggsave(filename = paste("Avg_BV_over_months_", year, ".png"), plot = last_plot())
}
print(last_plot())
```



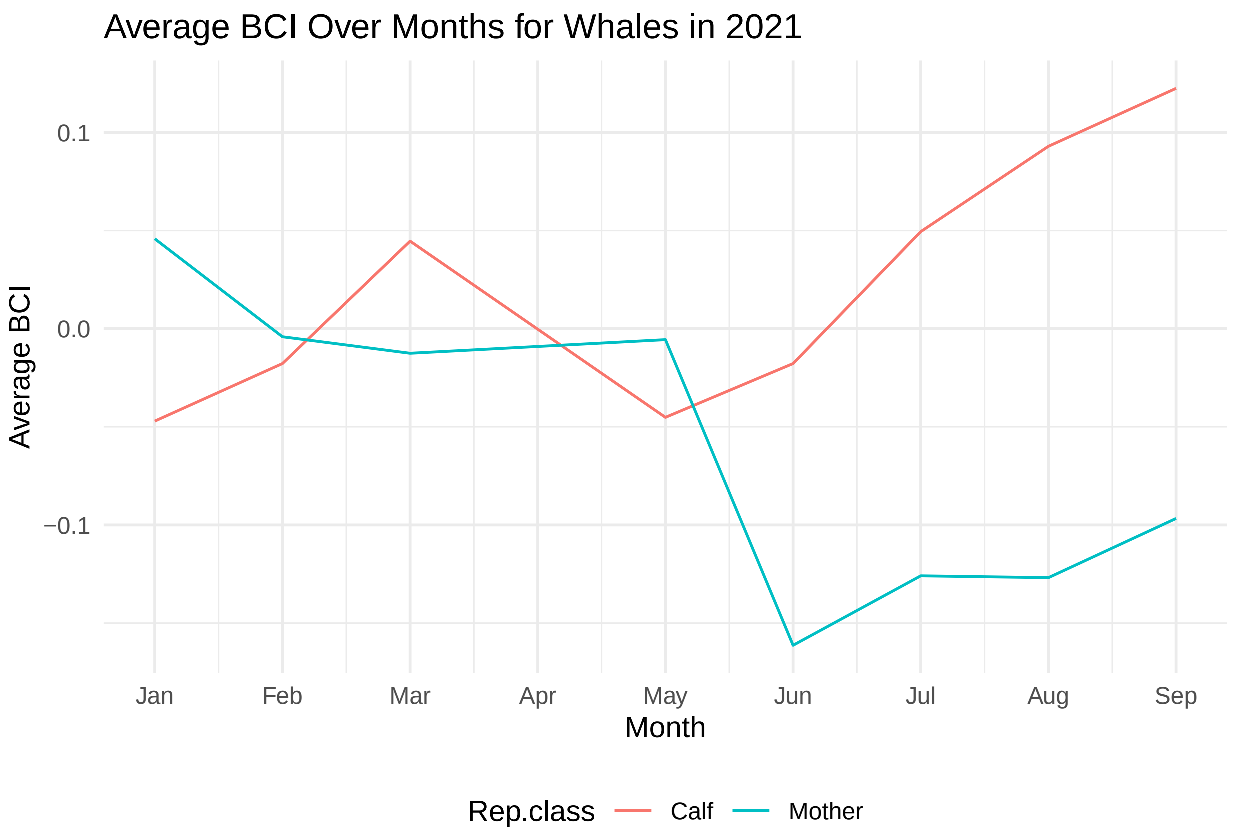
```
# Convert 'Year' and 'Month' to numeric for plotting
avg_BCI_per_month <- avg_BCI_per_month %>%
  mutate(Year = as.numeric(Year),
         Month = as.numeric(Month))

# Filter data for year 2019
data_2019 <- avg_BCI_per_month %>%
  filter(Year == 2019)

# Plotting for 2019
ggplot(data_2019, aes(x = Month, y = Avg_BCI, group = Rep.class, color = Rep.class)) +
  geom_line() +
  labs(title = "Average BCI Over Months for Whales in 2019",
       x = "Month", y = "Average BCI") +
  scale_x_continuous(breaks = 1:12, labels = month.abb) +
  theme_minimal() +
  theme(legend.position = "bottom")
```



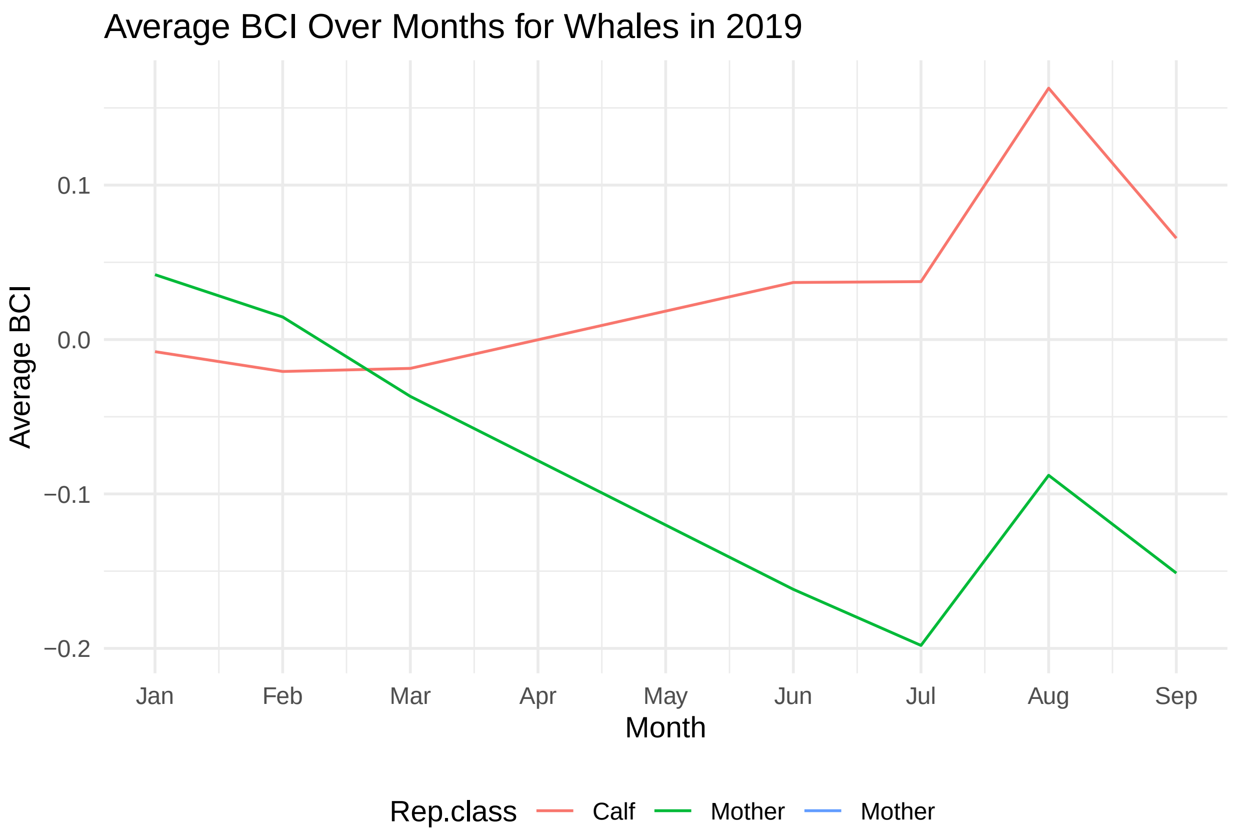
```
# Filter data for year 2020
data_2020 <- avg_BCI_per_month %>%
  filter(Year == 2020)

# Plotting for 2020
ggplot(data_2020, aes(x = Month, y = Avg_BCI, group = Rep.class, color = Rep.class)) +
  geom_line() +
  labs(title = "Average BCI Over Months for Whales in 2020",
       x = "Month", y = "Average BCI") +
  scale_x_continuous(breaks = 1:12, labels = month.abb) +
  theme_minimal() +
  theme(legend.position = "bottom")
```



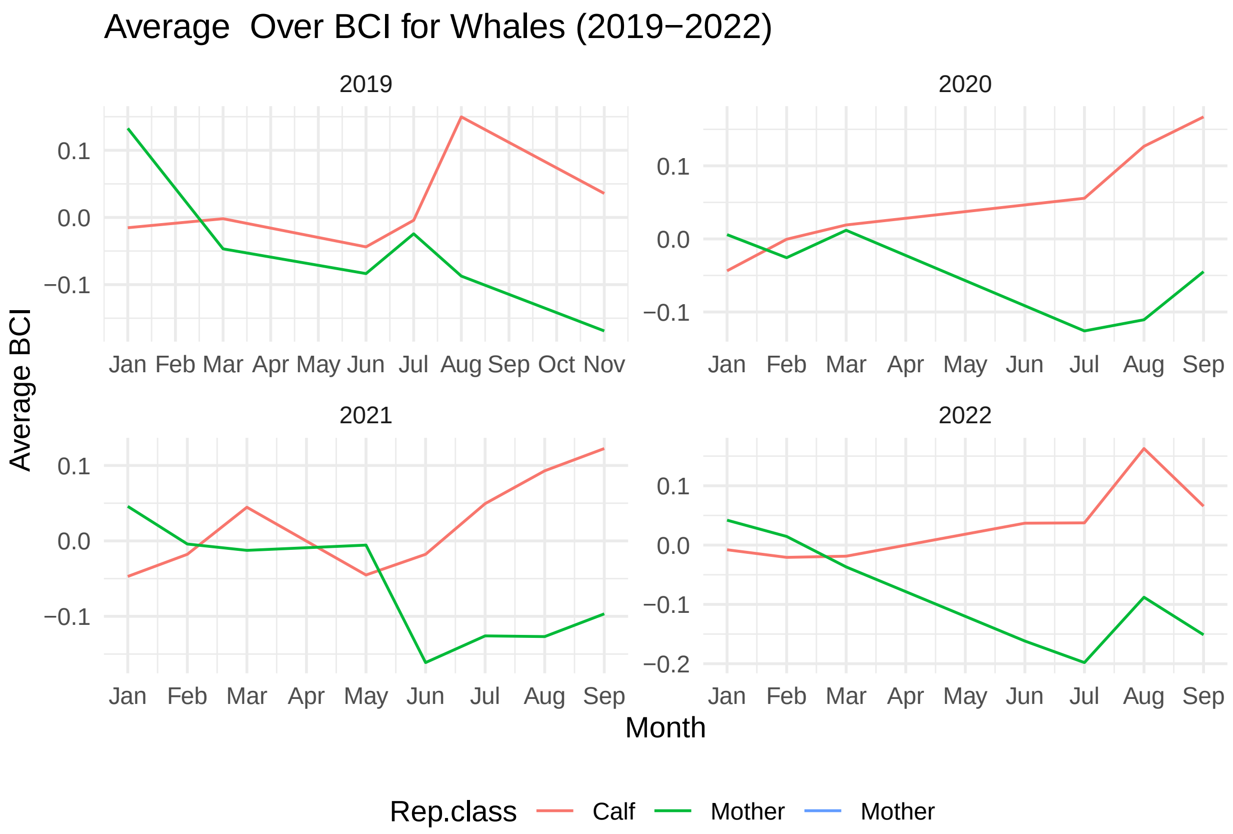
```
# Filter data for year 2021
data_2021 <- avg_BCI_per_month %>%
  filter(Year == 2021)

# Plotting for 2021
ggplot(data_2021, aes(x = Month, y = Avg_BCI, group = Rep.class, color = Rep.class)) +
  geom_line() +
  labs(title = "Average BCI Over Months for Whales in 2021",
       x = "Month", y = "Average BCI") +
  scale_x_continuous(breaks = 1:12, labels = month.abb) +
  theme_minimal() +
  theme(legend.position = "bottom")
```



```r
# Filter data for year 2022

data_2022 <- avg_BCI_per_month %>%
  filter(Year == 2022)

# Plotting for 2022
ggplot(data_2022, aes(x = Month, y = Avg_BCI, group = Rep.class, color = Rep.class)) +
  geom_line() +
  labs(title = "Average BCI Over Months for Whales in 2019",
       x = "Month", y = "Average BCI") +
  scale_x_continuous(breaks = 1:12, labels = month.abb) +
  theme_minimal() +
  theme(legend.position = "bottom")
```



```r
# Filter data for years 2019 to 2022
data_years <- avg_BCI_per_month %>%
  filter(Year %in% c(2019, 2020, 2021, 2022))

# Plotting for all years in one panel
ggplot(data_years, aes(x = Month, y = Avg_BCI, group = Rep.class, color = Rep.class)) +
  geom_line() +
  labs(title = "Average  Over BCI for Whales (2019-2022)",
       x = "Month", y = "Average BCI") +
  scale_x_continuous(breaks = 1:12, labels = month.abb) +
  facet_wrap(~Year, scales = "free") +
  theme_minimal() +
  theme(legend.position = "bottom")
```



```r
summary(lm(Avg_BCI~Month*Year, data_years))
```

```
##
## Call:
## lm(formula = Avg_BCI ~ Month * Year, data = data_years)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.174353 -0.048193 -0.002956  0.053466  0.189335
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.4515878 43.5106655   0.263    0.793
## Month       -0.4125044  6.9919074  -0.059    0.953
## Year        -0.0056655  0.0215338  -0.263    0.794
## Month:Year   0.0002026  0.0034606   0.059    0.954
##
## Residual standard error: 0.08721 on 52 degrees of freedom
## Multiple R-squared:  0.01491,    Adjusted R-squared:  -0.04192
## F-statistic: 0.2624 on 3 and 52 DF,  p-value: 0.8522
```