Homework-3

Alana Pengilley

15/04/2021

Challenge 1

For this exercise, the end aim is to fit a simple linear regression model to predict weaning age (WeaningAge_d) measured in days from species' brain size (Brain_Size_Species_Mean) measured in grams.

```
library(readr)
library(ggplot2)
library(broom)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v tibble  3.0.6     v dplyr   1.0.4
## v tidyr   1.1.2     v stringr 1.4.0
## v purrr   0.3.4     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(dplyr)
```

Load in Kamilar and Cooper Data

```
f <- "https://raw.githubusercontent.com/difiore/ada-2021-datasets/main/KamilarAndCooperData.csv"
d <- read_csv(f, col_names = T)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   Scientific_Name = col_character(),
##   Family = col_character(),
##   Genus = col_character(),
##   Species = col_character(),
##   Brain_size_Ref = col_character(),
##   Mass_Ref = col_character(),
##   Social_Organization_Ref = col_character(),
##   Life_History_Ref = col_character(),
##   Climate_Ref = col_character(),
##   HomeRangeRef = col_character(),
##   DayLengthRef = col_character(),
##   Leaves = col_character(),
##   Fauna = col_character(),
##   DietRef1 = col_character(),
##   Canine_Dimorphism_Ref = col_character(),
```

```
##   Activity_Budget_Ref = col_character()
## )
## i Use `spec()` for the full column specifications.
```

```
head(d)
```

```
## # A tibble: 6 x 44
##   Scientific_Name Family Genus Species Brain_Size_Spec~ Brain_Size_Fema~
##   <chr>           <chr>  <chr> <chr>              <dbl>            <dbl>
## 1 Allenopithecus~ Cerco~ Alle~ nigrov~             58.0             53.7
## 2 Allocebus_tric~ Cerco~ Allo~ tricho~             NA               NA
## 3 Alouatta_belze~ Ateli~ Alou~ belzeb~             52.8             51.2
## 4 Alouatta_caraya Ateli~ Alou~ caraya              52.6             47.8
## 5 Alouatta_guari~ Ateli~ Alou~ guariba             51.7             49.1
## 6 Alouatta_palli~ Ateli~ Alou~ pallia~             49.9             48.0
## # ... with 38 more variables: Brain_size_Ref <chr>, Body_mass_male_mean <dbl>,
## #   Body_mass_female_mean <dbl>, Mass_Dimorphism <dbl>, Mass_Ref <chr>,
## #   MeanGroupSize <dbl>, AdultMales <dbl>, AdultFemale <dbl>,
## #   AdultSexRatio <dbl>, Social_Organization_Ref <chr>,
## #   InterbirthInterval_d <dbl>, Gestation <dbl>, WeaningAge_d <dbl>,
## #   MaxLongevity_m <dbl>, LitterSz <dbl>, Life_History_Ref <chr>,
## #   GR_MidRangeLat_dd <dbl>, Precip_Mean_mm <dbl>, Temp_Mean_degC <dbl>,
## #   AET_Mean_mm <dbl>, PET_Mean_mm <dbl>, Climate_Ref <chr>,
## #   HomeRange_km2 <dbl>, HomeRangeRef <chr>, DayLength_km <dbl>,
## #   DayLengthRef <chr>, Territoriality <dbl>, Fruit <dbl>, Leaves <chr>,
## #   Fauna <chr>, DietRef1 <chr>, Canine_Dimorphism <dbl>,
## #   Canine_Dimorphism_Ref <chr>, Feed <dbl>, Move <dbl>, Rest <dbl>,
## #   Social <dbl>, Activity_Budget_Ref <chr>
```

Fit the regression on a normal model and using {ggplot2}, produce a scatterplot with the fitted line superimposed on the data and add fitted model equation to plot.

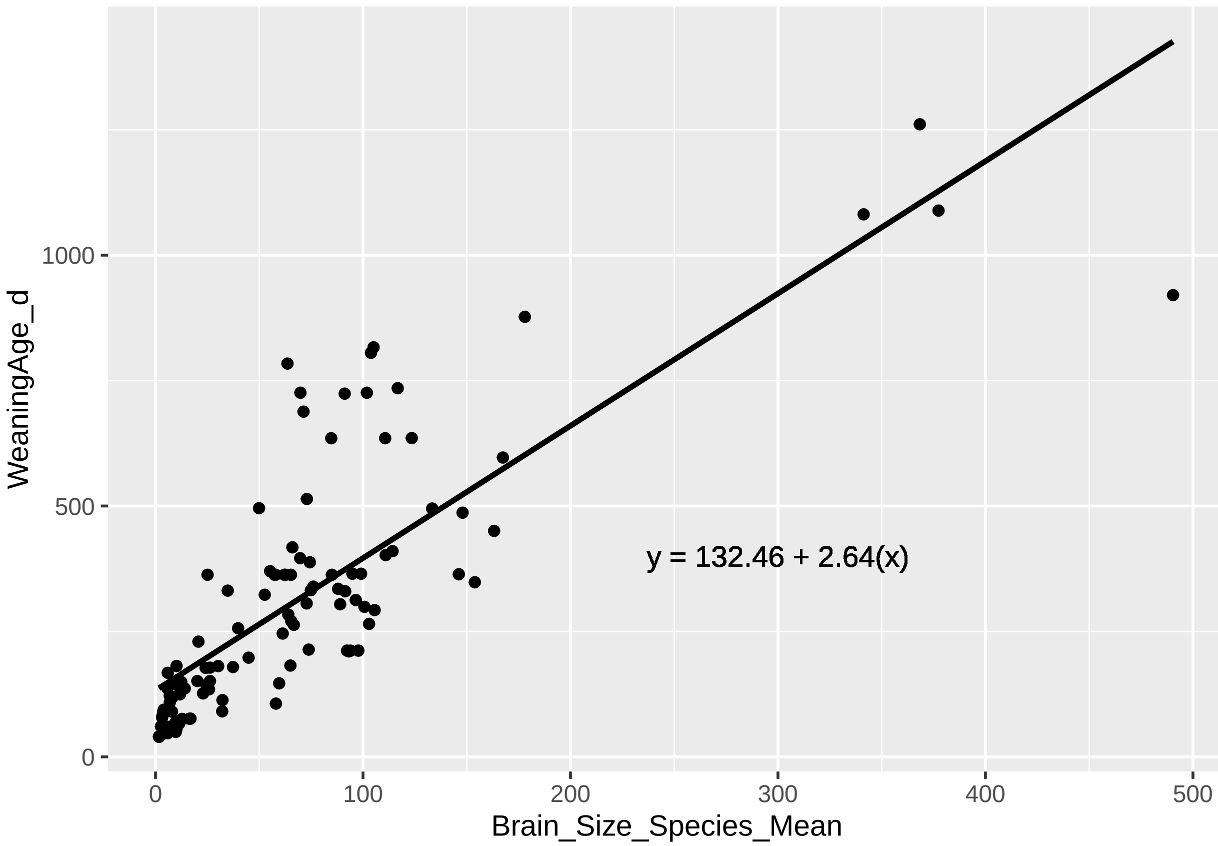
```
#normal model
Mod_1 <- lm(WeaningAge_d~Brain_Size_Species_Mean, data = d)
Mod_1
```

```
##
## Call:
## lm(formula = WeaningAge_d ~ Brain_Size_Species_Mean, data = d)
##
## Coefficients:
##             (Intercept)  Brain_Size_Species_Mean
##                 132.464                    2.637
```

```
p1 <- ggplot(data = d, aes(x = Brain_Size_Species_Mean, y = WeaningAge_d), na.rm = T)+
              geom_point() +
              geom_smooth(method = "lm", formula = y~x, se = F, color = "black")

#append fitted model equation to plot
p1 <- p1 + geom_text(x = 300, y = 400, label = "y = 132.46 + 2.64(x)", col = "black")
p1
```

```
## Warning: Removed 104 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 104 rows containing missing values (geom_point).
```



Identify and interpret the point estimate of the slope (beta1)

```
Mod_1.summary <- tidy(Mod_1)

beta1 <- Mod_1.summary %>%
  filter(term == "Brain_Size_Species_Mean") %>%
  pull(estimate)

beta0 <- Mod_1.summary %>%
  filter(term == "(Intercept)") %>%
  pull(estimate)

beta1
```

```
## [1] 2.637107
```

```
beta0
```

```
## [1] 132.464
```

```
#test null hypothesis that beta1 = 0
beta1 ==0
```

```
## [1] FALSE
```

```
#test alternative hypothesis that beta1 does not = 0
beta1!=0
```

```
## [1] TRUE
```

```r
#Rejects the null hypothesis. Beta1 is the true slope; every time the x value increased, the y value wi
```

Find a 90% CI for the slope (beta1) parameter. Using your model, add lines for the 90% confidence

```r
ci <- predict(Mod_1,
              newdata = data.frame(Brain_Size_Species_Mean = d$Brain_Size_Species_Mean),
              interval = "confidence", level = 0.9)
ci
```

```
##         fit      lwr      upr
## 1  285.4690 261.3345 309.6034
## 2        NA       NA       NA
## 3  271.8088 247.4128 296.2047
## 4  271.2550 246.8463 295.6636
## 5  268.8025 244.3355 293.2694
## 6  264.0029 239.4127 288.5931
## 7  267.2993 242.7950 291.8036
## 8  278.0851 253.8218 302.3483
## 9  186.9730 158.9221 215.0240
## 10       NA       NA       NA
## 11 175.4489 146.6757 204.2220
## 12       NA       NA       NA
## 13       NA       NA       NA
## 14 176.8993 148.2194 205.5791
## 15       NA       NA       NA
## 16 150.7128 120.2480 181.1776
## 17 441.0583 412.8866 469.2299
## 18 433.7271 405.9961 461.4581
## 19 409.5976 383.1743 436.0209
## 20 406.3276 380.0635 432.5917
## 21 158.4659 128.5508 188.3811
## 22 153.4290 123.1587 183.6993
## 23 440.2671 412.1439 468.3903
## 24 424.3390 397.1437 451.5343
## 25 332.8841 308.8521 356.9162
## 26 313.8179 289.8967 337.7390
## 27       NA       NA       NA
## 28       NA       NA       NA
## 29       NA       NA       NA
## 30       NA       NA       NA
## 31       NA       NA       NA
## 32 162.6062 132.9775 192.2348
## 33 153.4290 123.1587 183.6993
## 34       NA       NA       NA
## 35       NA       NA       NA
## 36       NA       NA       NA
## 37 151.5567 121.1525 181.9608
## 38       NA       NA       NA
## 39 151.7677 121.3786 182.1567
## 40 143.4608 112.4670 174.4546
## 41 305.0627 281.1238 329.0015
## 42 308.1745 284.2469 332.1021
## 43 324.7882 300.8282 348.7482
## 44 316.6396 292.7149 340.5642
## 45 306.7504 282.8183 330.6825
```

```
## 46   439.5024  411.4258  467.5789
## 47   393.7222  368.0283  419.4161
## 48   411.9710  385.4293  438.5126
## 49   382.1453  356.9115  407.3791
## 50   317.3252  293.3990  341.2514
## 51   289.5829  265.5072  313.6585
## 52   283.8076  259.6468  307.9684
## 53   304.5616  280.6204  328.5028
## 54   297.5733  273.5850  321.5616
## 55   304.9308  280.9914  328.8703
## 56   306.2494  282.3154  330.1833
## 57   328.1374  304.1520  352.1227
## 58   320.5689  296.6318  344.5059
## 59   295.5427  271.5356  319.5498
## 60   306.4340  282.5007  330.3672
## 61   320.0414  296.1066  343.9763
## 62   277.7159  253.4454  301.9863
## 63   289.5301  265.4538  313.6064
## 64         NA        NA        NA
## 65         NA        NA        NA
## 66   294.5142  270.4967  318.5318
## 67   147.7856  117.1090  178.4622
## 68   139.3205  108.0185  170.6225
## 69   259.9154  235.2108  284.6200
## 70   303.8760  279.9314  327.8206
## 71   297.4942  273.5052  321.4832
## 72   303.6386  279.6928  327.5845
## 73   337.3672  313.2796  361.4549
## 74   328.6384  304.6487  352.6281
## 75   327.1616  303.1843  351.1389
## 76   329.9833  305.9813  353.9853
## 77   325.1574  301.1950  349.1199
## 78   250.7383  225.7458  275.7307
## 79   390.1885  364.6417  415.7353
## 80   189.0300  161.1032  216.9568
## 81   190.7441  162.9196  218.5686
## 82         NA        NA        NA
## 83   186.9203  158.8661  214.9744
## 84   200.4223  173.1551  227.6894
## 85   197.0995  169.6449  224.5541
## 86   185.6545  157.5232  213.7858
## 87   201.6353  174.4356  228.8351
## 88   199.4465  172.1248  226.7683
## 89         NA        NA        NA
## 90   147.0472  116.3168  177.7777
## 91   144.6475  113.7412  175.5537
## 92   142.2477  111.1641  173.3313
## 93   142.9070  111.8722  173.9417
## 94   139.4524  108.1603  170.7445
## 95   141.7203  110.5975  172.8431
## 96  1427.9955 1296.2182 1559.7727
## 97  1425.7275 1294.2095 1557.2456
## 98         NA        NA        NA
## 99   169.6209  140.4658  198.7759
```

```
## 100  168.8561  139.6502  198.0621
## 101        NA        NA        NA
## 102  372.8627  347.9513  397.7741
## 103  364.5031  339.8453  389.1608
## 104  401.1061  375.0867  427.1255
## 105        NA        NA        NA
## 106  356.9609  332.5010  381.4208
## 107  355.8006  331.3685  380.2327
## 108  224.2617  198.2108  250.3126
## 109  386.9448  361.5279  412.3618
## 110  192.8538  165.1537  220.5539
## 111  163.6874  134.1327  193.2421
## 112        NA        NA        NA
## 113  166.2981  136.9206  195.6756
## 114  150.1326  119.6261  180.6392
## 115  151.5831  121.1808  181.9853
## 116  150.5810  120.1067  181.0552
## 117        NA        NA        NA
## 118        NA        NA        NA
## 119  154.2202  124.0061  184.4342
## 120  380.2730  355.1076  405.4384
## 121  400.3677  374.3819  426.3535
## 122  149.1833  118.6081  179.7585
## 123  147.9438  117.2787  178.6090
## 124  398.0207  372.1403  423.9011
## 125  371.0167  346.1644  395.8690
## 126  348.7068  324.4290  372.9845
## 127  301.1861  277.2256  325.1466
## 128  403.8751  377.7274  430.0228
## 129  358.2004  333.7100  382.6907
## 130  367.1138  342.3806  391.8470
## 131  410.9161  384.4274  437.4049
## 132  382.7255  357.4701  407.9808
## 133        NA        NA        NA
## 134  329.9042  305.9030  353.9055
## 135  356.6181  332.1665  381.0697
## 136  316.2704  292.3464  340.1943
## 137  378.2424  353.1493  403.3355
## 138        NA        NA        NA
## 139  522.7558  488.6033  556.9084
## 140  538.2620  502.8007  573.7233
## 141  136.7625  105.2680  168.2571
## 142        NA        NA        NA
## 143  136.9999  105.5232  168.4765
## 144  230.9600  205.2075  256.7125
## 145  147.7593  117.0807  178.4378
## 146  375.8690  350.8578  400.8801
## 147        NA        NA        NA
## 148  447.2818  418.7215  475.8422
## 149        NA        NA        NA
## 150  168.0386  138.7780  197.2992
## 151  159.1779  129.3124  189.0435
## 152        NA        NA        NA
## 153  163.5292  133.9636  193.0947
```

```
## 154  162.7908  133.1747  192.4068
## 155 1032.4822  945.5088 1119.4556
## 156 1103.8423 1008.8675 1198.8171
## 157  573.9684  535.3420  612.5949
## 158  562.8135  525.1964  600.4306
## 159  517.9299  484.1754  551.6844
## 160  601.8690  560.6475  643.0906
## 161  165.2169  135.7662  194.6676
## 162  150.0799  119.5695  180.5903
## 163  300.1576  276.1900  324.1253
## 164  283.4384  259.2715  307.6053
## 165        NA        NA        NA
## 166        NA        NA        NA
## 167  319.5668  295.6337  343.4998
## 168        NA        NA        NA
## 169  217.5371  191.1669  243.9073
## 170 1159.6171 1058.3556 1260.8786
## 171 1127.6554 1029.9996 1225.3111
## 172  344.2237  320.0295  368.4179
## 173  303.4804  279.5337  327.4271
## 174  271.1758  246.7654  295.5863
## 175  212.0783  185.4349  238.7217
## 176  211.9728  185.3240  238.6216
## 177  237.4209  211.9370  262.9048
## 178  236.6034  211.0865  262.1202
## 179        NA        NA        NA
## 180  201.5826  174.3799  228.7853
## 181  373.5220  348.5891  398.4548
## 182        NA        NA        NA
## 183        NA        NA        NA
## 184  443.0097  414.7176  471.3018
## 185        NA        NA        NA
## 186  153.4027  123.1305  183.6748
## 187  159.2043  129.3406  189.0680
## 188        NA        NA        NA
## 189  158.0440  128.0993  187.9886
## 190  158.2549  128.3251  188.1848
## 191  161.7095  132.0193  191.3998
## 192  157.4638  127.4786  187.4490
## 193  158.2022  128.2686  188.1358
## 194        NA        NA        NA
## 195        NA        NA        NA
## 196  198.5763  171.2055  225.9471
## 197  196.1238  168.6134  223.6342
## 198  424.9983  397.7664  452.2302
## 199  458.1467  428.8774  487.4161
## 200  140.7973  109.6057  171.9888
## 201        NA        NA        NA
## 202  141.3247  110.1725  172.4770
## 203  484.0695  452.9663  515.1726
## 204        NA        NA        NA
## 205  285.0470  260.9060  309.1880
## 206  346.8608  322.6188  371.1028
## 207  355.5633  331.1368  379.9898
```

```
## 208  296.2811  272.2811  320.2811
## 209  324.5509  300.5925  348.5093
## 210  405.7738  379.5362  432.0114
## 211  294.0923  270.0703  318.1143
## 212  214.4253  187.9009  240.9497
## 213  217.1679  190.7796  243.5562
```

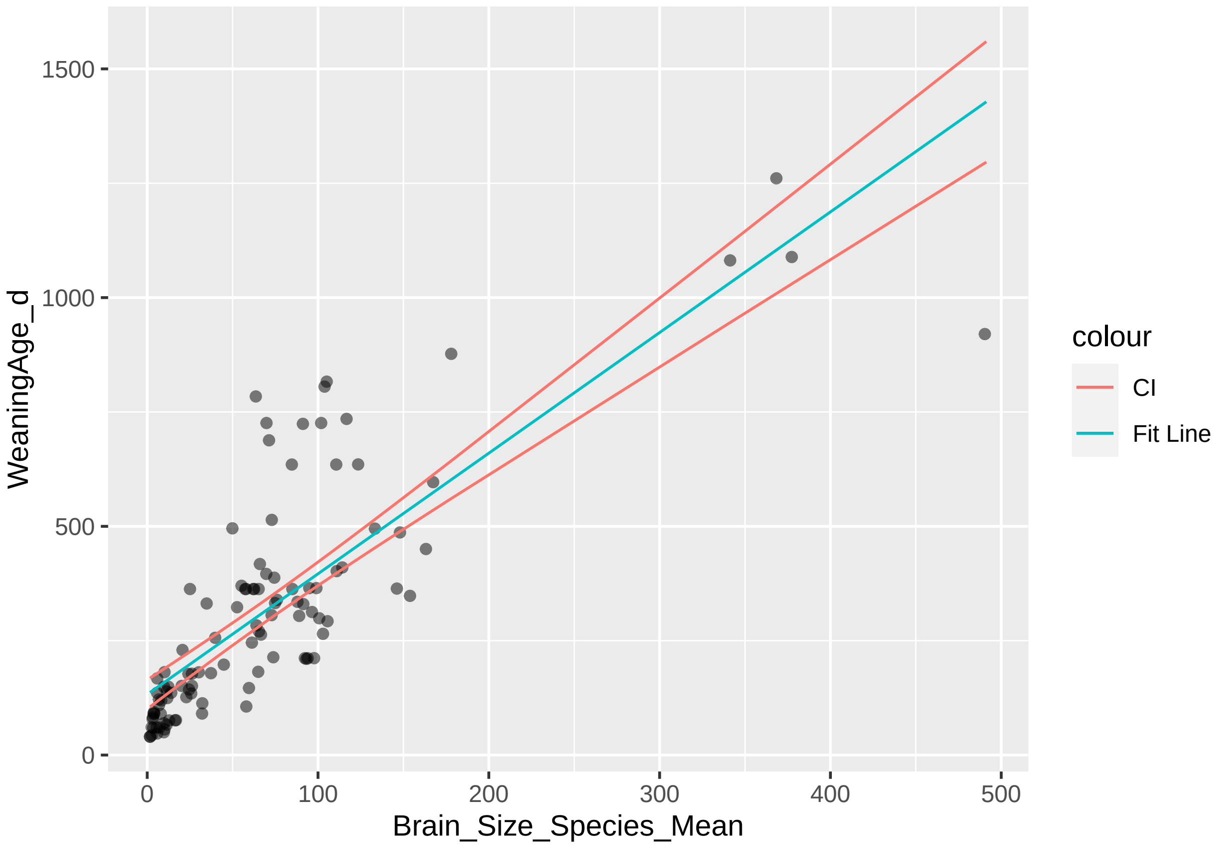
```
ci <- data.frame(ci)
ci<- cbind(d$Brain_Size_Species_Mean, ci)
names(ci) <- c("BrainSizeSpeciesMean", "c.fit", "c.lwr", "c.upr")

p1 <- ggplot(data = d, aes(x = Brain_Size_Species_Mean, y = WeaningAge_d), na.rm = T)
p1 <- p1 + geom_point(alpha = 0.5)
p1 <- p1 + geom_line(
  data = ci, aes(x = BrainSizeSpeciesMean, y = c.fit,
  color = "Fit Line"))
p1 <- p1 + geom_line(
  data = ci, aes(x = BrainSizeSpeciesMean, y = c.lwr,
  color = "CI"))
p1 <- p1 + geom_line(
  data = ci, aes(x = BrainSizeSpeciesMean, y = c.upr,
  color = "CI"))
p1
```

```
## Warning: Removed 104 rows containing missing values (geom_point).
```

```
## Warning: Removed 42 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 42 row(s) containing missing values (geom_path).
```

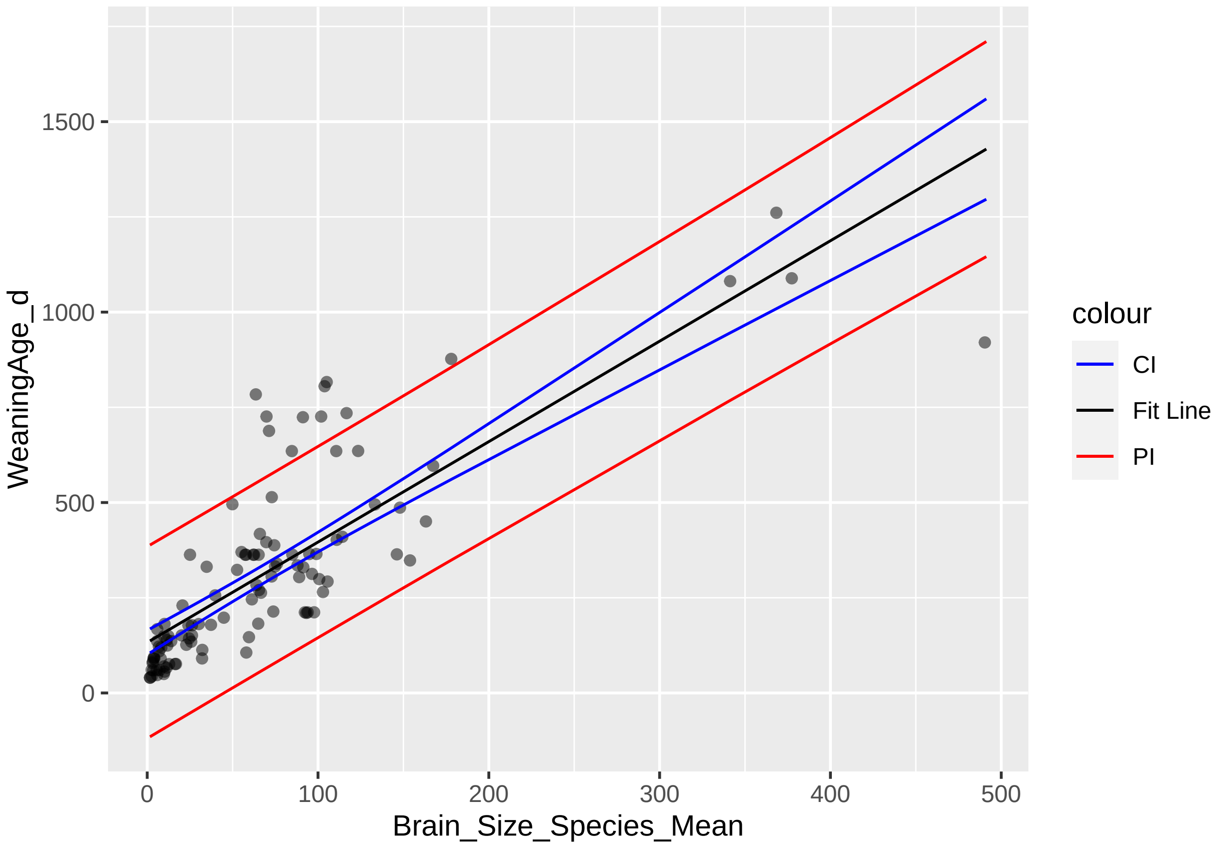
```
## Warning: Removed 42 row(s) containing missing values (geom_path).
```



Add lines for predication interval bands on the plot

```
pi <- predict(Mod_1, newdata = data.frame(Brain_Size_Species_Mean=d$Brain_Size_Species_Mean), interval =
pi <- data.frame(pi)
pi <- cbind(d$Brain_Size_Species_Mean, pi)
names(pi) <- c("BrainSizeSpeciesMean", "p.fit", "p.lwr", "p.upr")

p1 <- p1 + geom_line(data = pi, aes(x = BrainSizeSpeciesMean, y = p.lwr, col = "PI"))
p1 <- p1 + geom_line(data = pi, aes(x = BrainSizeSpeciesMean, y = p.upr, col = "PI"))
p1 <- p1 + scale_color_manual(values = c("blue", "black", "red"))
p1
```

```
## Warning: Removed 104 rows containing missing values (geom_point).

## Warning: Removed 42 row(s) containing missing values (geom_path).

## Warning: Removed 42 row(s) containing missing values (geom_path).

## Warning: Removed 42 row(s) containing missing values (geom_path).

## Warning: Removed 42 row(s) containing missing values (geom_path).

## Warning: Removed 42 row(s) containing missing values (geom_path).
```



Produce a point estimate for the weaning age of a species whose brain weight is 750 gm

```
(h.hat <- beta1 * 750 + beta0)
```

```
## [1] 2110.294
```

Produce associated 90% prediction interval for the weaning age of a species whose brain wight is 750 gm.

```
pi <- predict(Mod_1,
              newdata = data.frame(Brain_Size_Species_Mean = 750),
              interval = "prediction", level = 0.90)
pi
```

```
##        fit      lwr      upr
## 1 2110.294 1783.832 2436.757
```

```
#1783.8 - 243.8
```

Log Model for Weaning age and Brain Size

Fit the regression on a normal model and using {ggplot2}, produce a scatterplot with the fitted line superimposed on the data and add fitted model equation to plot.

```
#log model
head(d)
```

```
## # A tibble: 6 x 44
##   Scientific_Name Family Genus Species Brain_Size_Spec~ Brain_Size_Fema~
##   <chr>           <chr>  <chr> <chr>              <dbl>            <dbl>
## 1 Allenopithecus~ Cerco~ Alle~ nigrov~             58.0             53.7
```

```
## 2 Allocebus_tric~ Cerco~ Allo~ tricho~            NA              NA
## 3 Alouatta_belze~ Ateli~ Alou~ belzeb~            52.8            51.2
## 4 Alouatta_caraya Ateli~ Alou~ caraya            52.6            47.8
## 5 Alouatta_guari~ Ateli~ Alou~ guariba           51.7            49.1
## 6 Alouatta_palli~ Ateli~ Alou~ pallia~            49.9            48.0
## # ... with 38 more variables: Brain_size_Ref <chr>, Body_mass_male_mean <dbl>,
## #   Body_mass_female_mean <dbl>, Mass_Dimorphism <dbl>, Mass_Ref <chr>,
## #   MeanGroupSize <dbl>, AdultMales <dbl>, AdultFemale <dbl>,
## #   AdultSexRatio <dbl>, Social_Organization_Ref <chr>,
## #   InterbirthInterval_d <dbl>, Gestation <dbl>, WeaningAge_d <dbl>,
## #   MaxLongevity_m <dbl>, LitterSz <dbl>, Life_History_Ref <chr>,
## #   GR_MidRangeLat_dd <dbl>, Precip_Mean_mm <dbl>, Temp_Mean_degC <dbl>,
## #   AET_Mean_mm <dbl>, PET_Mean_mm <dbl>, Climate_Ref <chr>,
## #   HomeRange_km2 <dbl>, HomeRangeRef <chr>, DayLength_km <dbl>,
## #   DayLengthRef <chr>, Territoriality <dbl>, Fruit <dbl>, Leaves <chr>,
## #   Fauna <chr>, DietRef1 <chr>, Canine_Dimorphism <dbl>,
## #   Canine_Dimorphism_Ref <chr>, Feed <dbl>, Move <dbl>, Rest <dbl>,
## #   Social <dbl>, Activity_Budget_Ref <chr>
```

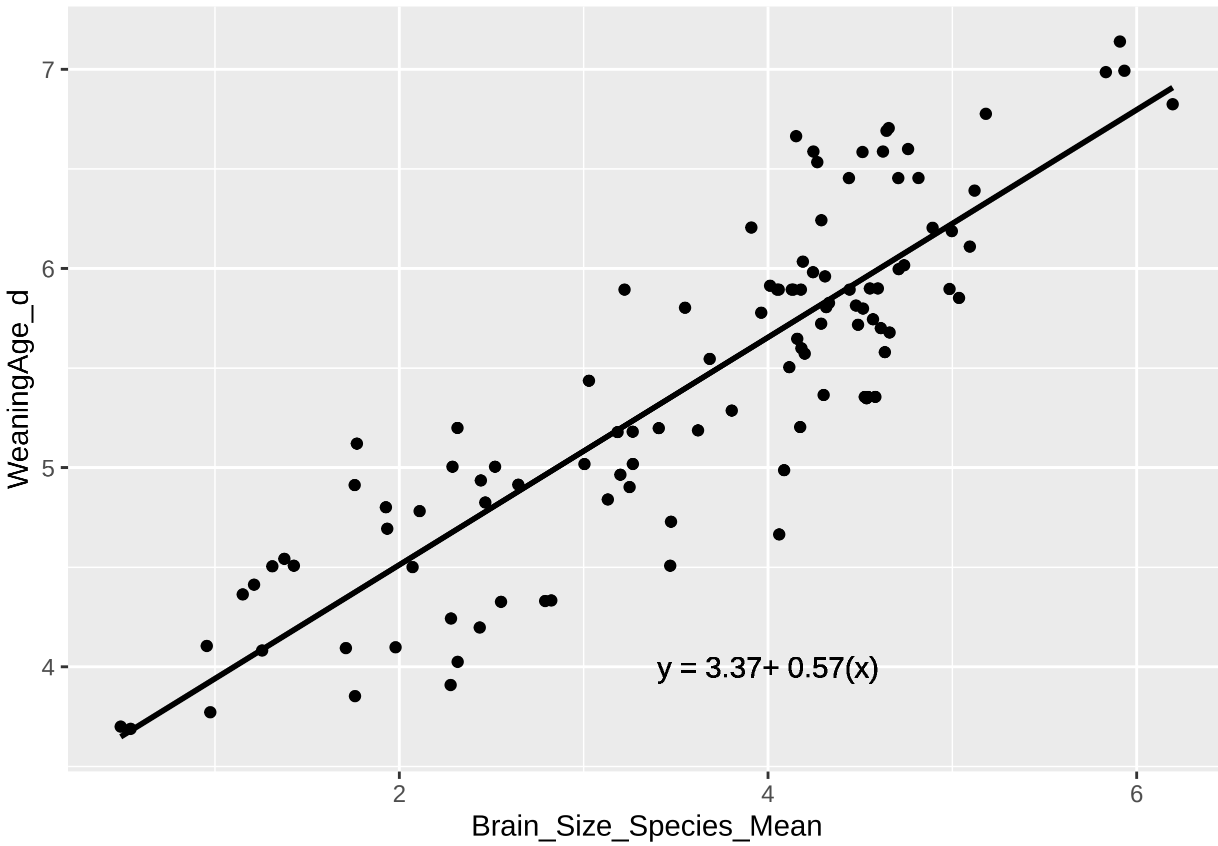
```
d$Brain_Size_Species_Mean<-(log(d$Brain_Size_Species_Mean))
d$WeaningAge_d<-(log(d$WeaningAge_d))


Mod_2 <- lm(WeaningAge_d ~ Brain_Size_Species_Mean, data = d)
Mod_2
```

```
##
## Call:
## lm(formula = WeaningAge_d ~ Brain_Size_Species_Mean, data = d)
##
## Coefficients:
##             (Intercept)  Brain_Size_Species_Mean
##                  3.3698                   0.5712
```

```
p2 <- ggplot(data = d, aes(x = Brain_Size_Species_Mean, y = WeaningAge_d), na.rm = T) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, se = F, color = "black") +
  geom_text(x = 4, y = 4, label = "y = 3.37+ 0.57(x)", col = "black")#append fitted model equation to p
p2
```

```
## Warning: Removed 104 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 104 rows containing missing values (geom_point).
```



Identify and interpret the point estimate of the slope (beta1)

```
Mod_2.summary <- tidy(Mod_2)

beta1 <- Mod_2.summary %>%
  filter(term == "Brain_Size_Species_Mean") %>%
  pull(estimate)

beta0 <- Mod_2.summary %>%
  filter(term == "(Intercept)") %>%
  pull(estimate)

beta1
```

```
## [1] 0.57116
```

```
beta0
```

```
## [1] 3.369846
```

```
#test null hypothesis that beta1 = 0
beta1 ==0
```

```
## [1] FALSE
```

```
#test alternative hypothesis that beta1 does not = 0
beta1!=0
```

```
## [1] TRUE
```

```
#therefore, we reject the null hypothesis. Beta1 is the true slope; every time the x value increased, t
```

Find a 90% CI for the slope (beta1) parameter. Using your model, add lines for the 90% confidence.

```
#log transform
ci2 <- predict(Mod_2,
               newdata = data.frame(Brain_Size_Species_Mean = d$Brain_Size_Species_Mean),
               interval = "confidence", level = 0.90)
ci2
```

```
##          fit      lwr      upr
## 1   5.689206 5.619675 5.758737
## 2         NA       NA       NA
## 3   5.635791 5.567747 5.703835
## 4   5.633517 5.565529 5.701504
## 5   5.623334 5.555594 5.691074
## 6   5.602865 5.535587 5.670143
## 7   5.617002 5.549409 5.684594
## 8   5.660955 5.592247 5.729663
## 9   5.099709 5.028472 5.170946
## 10        NA       NA       NA
## 11  4.964048 4.887187 5.040909
## 12        NA       NA       NA
## 13        NA       NA       NA
## 14  4.983002 4.907017 5.058988
## 15        NA       NA       NA
## 16  4.474707 4.368265 4.581149
## 17  6.089907 6.001544 6.178270
## 18  6.076174 5.988630 6.163719
## 19  6.028492 5.943712 6.113271
## 20  6.021712 5.937316 6.106108
## 21  4.676938 4.584040 4.769836
## 22  4.553959 4.452990 4.654928
## 23  6.088441 6.000166 6.176716
## 24  6.058093 5.971611 6.144574
## 25  5.843388 5.768034 5.918742
## 26  5.786291 5.713339 5.859243
## 27        NA       NA       NA
## 28        NA       NA       NA
## 29        NA       NA       NA
## 30        NA       NA       NA
## 31        NA       NA       NA
## 32  4.761330 4.673611 4.849049
## 33  4.553959 4.452990 4.654928
## 34        NA       NA       NA
## 35        NA       NA       NA
## 36        NA       NA       NA
## 37  4.500527 4.395889 4.605165
## 38        NA       NA       NA
## 39  4.506803 4.402601 4.611006
## 40  4.185415 4.057680 4.313149
## 41  5.758030 5.686164 5.829896
## 42  5.768236 5.695986 5.840486
## 43  5.819837 5.745506 5.894168
## 44  5.795110 5.721805 5.868415
```

```
## 45  5.763588 5.691514 5.835662
## 46  6.087020 5.998830 6.175210
## 47  5.994799 5.911898 6.077699
## 48  6.033362 5.948306 6.118418
## 49  5.968911 5.887409 6.050414
## 50  5.797232 5.723841 5.870623
## 51  5.704360 5.634354 5.774366
## 52  5.682970 5.613628 5.752313
## 53  5.756369 5.684565 5.828174
## 54  5.732692 5.661738 5.803646
## 55  5.757593 5.685743 5.829443
## 56  5.761943 5.689931 5.833956
## 57  5.829698 5.754944 5.904451
## 58  5.807167 5.733368 5.880966
## 59  5.725624 5.654914 5.796335
## 60  5.762550 5.690515 5.834585
## 61  5.805563 5.731831 5.879296
## 62  5.659505 5.590837 5.728173
## 63  5.704168 5.634168 5.774168
## 64        NA       NA       NA
## 65        NA       NA       NA
## 66  5.722011 5.651423 5.792599
## 67  4.374848 4.261264 4.488433
## 68  3.915596 3.766736 4.064456
## 69  5.584835 5.517925 5.651744
## 70  5.754089 5.682369 5.825810
## 71  5.732419 5.661474 5.803363
## 72  5.753298 5.681607 5.824989
## 73  5.856023 5.780102 5.931944
## 74  5.831158 5.756341 5.905975
## 75  5.826842 5.752212 5.901473
## 76  5.835061 5.760074 5.910048
## 77  5.820932 5.746555 5.895310
## 78  5.542153 5.475970 5.608335
## 79  5.987020 5.904545 6.069496
## 80  5.120866 5.050360 5.191371
## 81  5.137916 5.067969 5.207864
## 82        NA       NA       NA
## 83  5.099156 5.027899 5.170413
## 84  5.225666 5.158124 5.293207
## 85  5.197033 5.128796 5.265271
## 86  5.085723 5.013980 5.157466
## 87  5.235771 5.168454 5.303088
## 88  5.217406 5.149672 5.285139
## 89        NA       NA       NA
## 90  4.346637 4.230991 4.462283
## 91  4.243947 4.120656 4.367237
## 92  4.118655 3.985784 4.251526
## 93  4.155902 4.025905 4.285898
## 94  3.926476 3.778485 4.074467
## 95  4.087004 3.951675 4.222333
## 96  6.909321 6.761700 7.056942
## 97  6.908320 6.760780 7.055861
## 98        NA       NA       NA
```

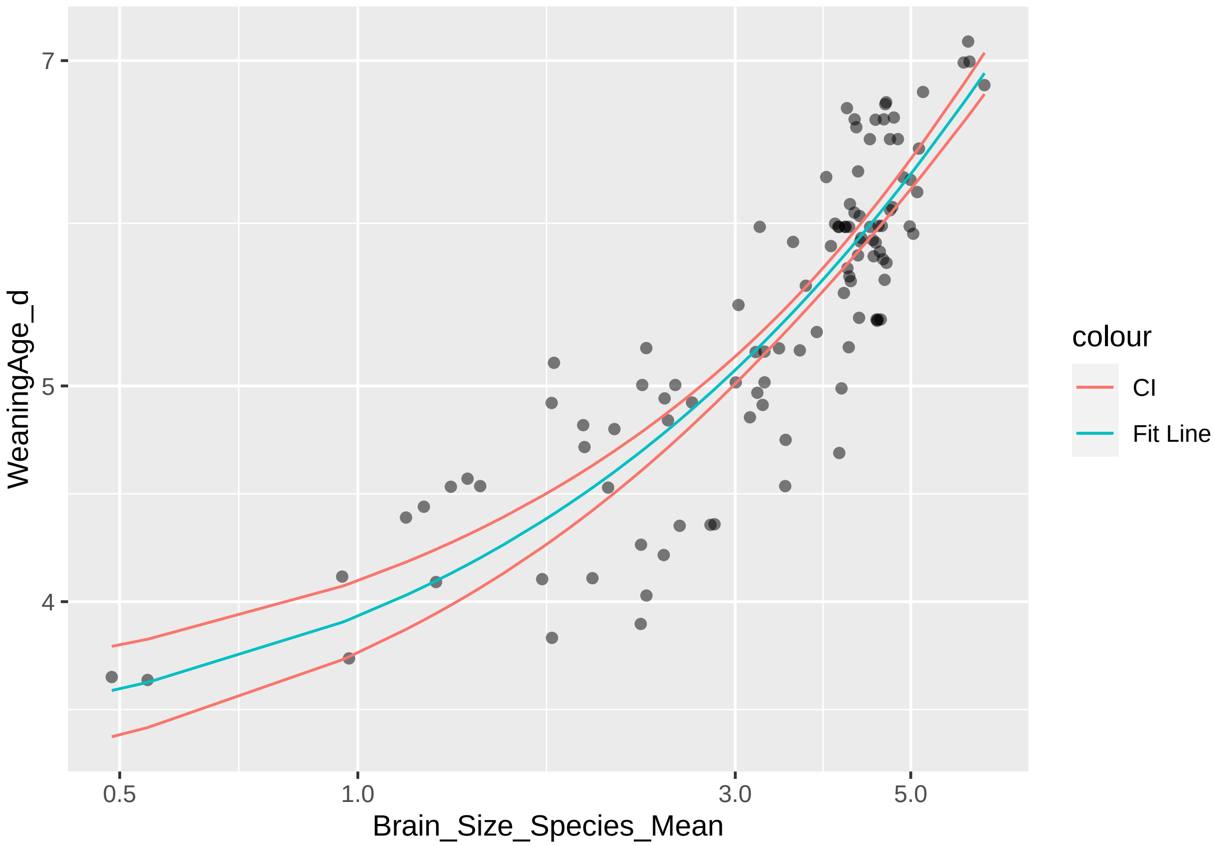
```
## 99   4.880830 4.799820 4.961840
## 100 4.868952 4.787313 4.950591
## 101       NA       NA       NA
## 102 5.947272 5.866907 6.027637
## 103 5.927057 5.847726 6.006388
## 104 6.010717 5.926937 6.094497
## 105       NA       NA       NA
## 106 5.908184 5.829794 5.986573
## 107 5.905224 5.826980 5.983468
## 108 5.397410 5.332090 5.462730
## 109 5.979786 5.897702 6.061871
## 110 5.158226 5.088907 5.227546
## 111 4.781459 4.694925 4.867993
## 112       NA       NA       NA
## 113 4.827324 4.743411 4.911238
## 114 4.456254 4.348512 4.563996
## 115 4.501315 4.396732 4.605898
## 116 4.470565 4.363832 4.577298
## 117       NA       NA       NA
## 118       NA       NA       NA
## 119 4.575116 4.475574 4.674657
## 120 5.964612 5.883338 6.045886
## 121 6.009145 5.925453 6.092838
## 122 4.424710 4.314723 4.534696
## 123 4.380716 4.267558 4.493875
## 124 6.004119 5.920706 6.087533
## 125 5.942869 5.862732 6.023007
## 126 5.886788 5.809435 5.964141
## 127 5.745055 5.673664 5.816447
## 128 6.016574 5.932467 6.100682
## 129 5.911328 5.832784 5.989873
## 130 5.933447 5.853793 6.013102
## 131 6.031203 5.946269 6.116136
## 132 5.970237 5.888664 6.051810
## 133       NA       NA       NA
## 134 5.834832 5.759855 5.909809
## 135 5.907311 5.828964 5.985658
## 136 5.793964 5.720705 5.867223
## 137 5.959913 5.878887 6.040938
## 138       NA       NA       NA
## 139 6.224054 6.127252 6.320856
## 140 6.246307 6.148035 6.344578
## 141 3.648904 3.478417 3.819391
## 142       NA       NA       NA
## 143 3.679600 3.511631 3.847569
## 144 5.437636 5.372328 5.502943
## 145 4.373864 4.260208 4.487521
## 146 5.954370 5.873636 6.035105
## 147       NA       NA       NA
## 148 6.101311 6.012262 6.190361
## 149       NA       NA       NA
## 150 4.855975 4.773638 4.938313
## 151 4.692368 4.600441 4.784295
## 152       NA       NA       NA
```

```
## 153 4.778557 4.691854 4.865260
## 154 4.764817 4.677305 4.852329
## 155 6.701270 6.569984 6.832555
## 156 6.744850 6.610189 6.879511
## 157 6.294474 6.192962 6.395987
## 158 6.279858 6.179337 6.380379
## 159 6.216948 6.120611 6.313284
## 160 6.329474 6.225559 6.433388
## 161 4.808774 4.723815 4.893734
## 162 4.454546 4.346683 4.562410
## 163 5.741563 5.670296 5.812830
## 164 5.681575 5.612274 5.750876
## 165       NA       NA       NA
## 166       NA       NA       NA
## 167 5.804116 5.730443 5.877789
## 168       NA       NA       NA
## 169 5.353958 5.288404 5.419512
## 170 6.776738 6.639591 6.913885
## 171 6.758683 6.622945 6.894420
## 172 5.874822 5.798035 5.951610
## 173 5.752770 5.681098 5.824442
## 174 5.633191 5.565212 5.701170
## 175 5.316080 5.250138 5.382022
## 176 5.315323 5.249371 5.381274
## 177 5.473924 5.408461 5.539387
## 178 5.469458 5.404022 5.534893
## 179       NA       NA       NA
## 180 5.235335 5.168009 5.302662
## 181 5.948836 5.868390 6.029283
## 182       NA       NA       NA
## 183       NA       NA       NA
## 184 6.093508 6.004929 6.182086
## 185       NA       NA       NA
## 186 4.553240 4.452222 4.654258
## 187 4.692932 4.601040 4.784823
## 188       NA       NA       NA
## 189 4.667594 4.574103 4.761084
## 190 4.672285 4.579092 4.765478
## 191 4.744082 4.655332 4.832832
## 192 4.654490 4.560162 4.748818
## 193 4.671116 4.577849 4.764383
## 194       NA       NA       NA
## 195       NA       NA       NA
## 196 5.209936 5.142023 5.277849
## 197 5.188346 5.119879 5.256812
## 198 6.059381 5.972824 6.145938
## 199 6.120691 6.030461 6.210920
## 200 4.027007 3.886982 4.167032
## 201       NA       NA       NA
## 202 4.062058 3.924783 4.199334
## 203 6.164433 6.071477 6.257390
## 204       NA       NA       NA
## 205 5.687629 5.618145 5.757112
## 206 5.881891 5.804771 5.959012
```

```
## 207 5.904617 5.826402 5.982831
## 208 5.728205 5.657406 5.799004
## 209 5.819132 5.744831 5.893432
## 210 6.020556 5.936225 6.104887
## 211 5.720522 5.649984 5.791059
## 212 5.332674 5.266923 5.398426
## 213 5.351474 5.285900 5.417048
```

```
ci2 <- data.frame(ci2)
ci2 <- cbind(d$Brain_Size_Species_Mean, ci2)
names(ci2) <- c("LogBrainSize", "c.fit", "c.lwr", "c.upr")

p2 <- ggplot(data = d, aes(x = Brain_Size_Species_Mean, y = WeaningAge_d), na.rm = T)
p2 <- p2 + geom_point(alpha = 0.5)
p2 <- p2 + geom_line(
  data = ci2, aes(x = LogBrainSize, y = c.fit,
  colour = "Fit Line"))
p2 <- p2 + geom_line(
  data = ci2, aes(x = LogBrainSize, y = c.lwr,
  color = "CI"))
p2 <- p2 + geom_line(
  data = ci2, aes(x = LogBrainSize, y = c.upr,
  color = "CI"))
p2 <- p2 + scale_y_continuous(trans = "log10")
p2 <- p2 + scale_x_continuous(trans = "log10")
p2
```

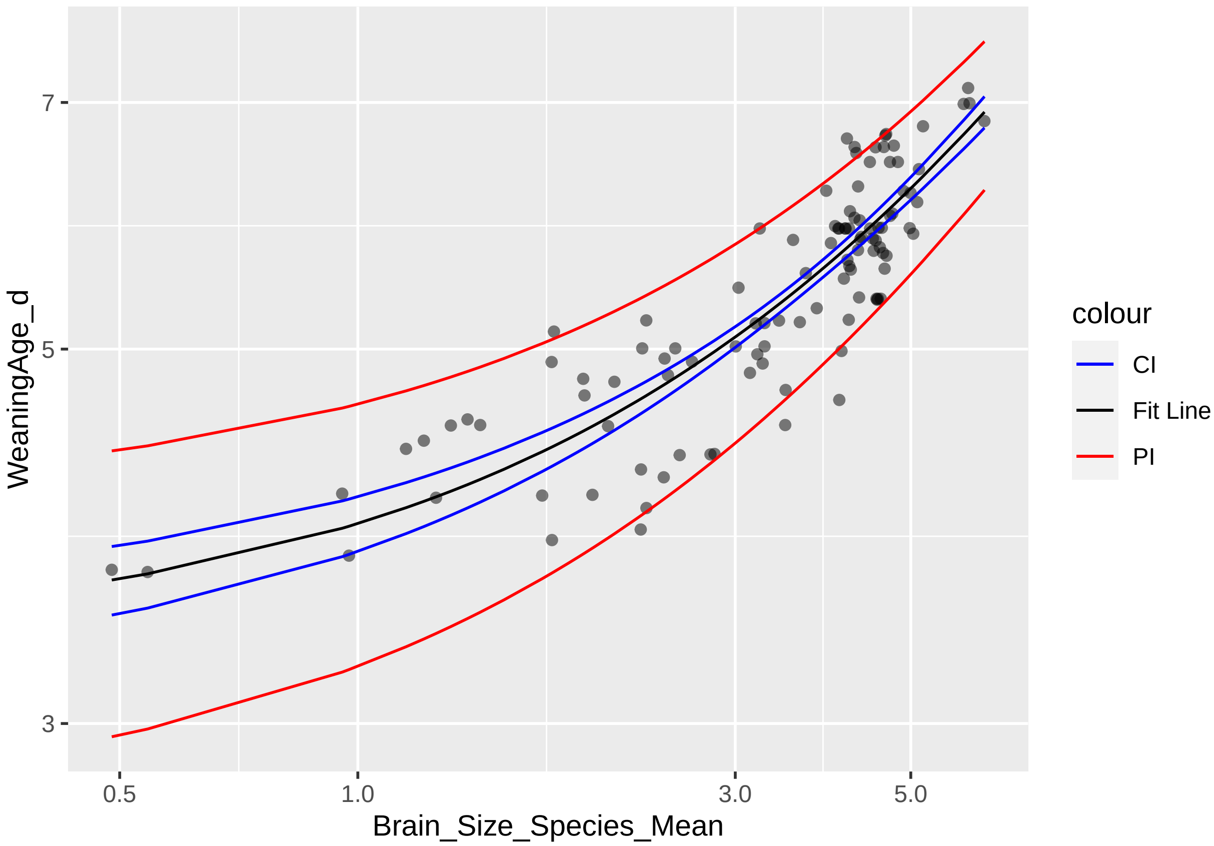
```
## Warning: Removed 104 rows containing missing values (geom_point).

## Warning: Removed 42 row(s) containing missing values (geom_path).

## Warning: Removed 42 row(s) containing missing values (geom_path).

## Warning: Removed 42 row(s) containing missing values (geom_path).
```



Add lines for predication interval bands on the plot

```
pi2 <- predict(Mod_2, newdata = data.frame(Brain_Size_Species_Mean=d$Brain_Size_Species_Mean), interval

pi2 <- data.frame(pi2)
pi2 <- cbind(d$Brain_Size_Species_Mean, pi2)
names(pi2) <- c("LogBrainSize", "p.fit", "p.lwr", "p.upr")

p2 <- p2 + geom_line(data = pi2, aes(x = LogBrainSize, y = p.lwr, color = "PI"))
p2 <- p2 + geom_line(data = pi2, aes(x = LogBrainSize, y = p.upr, color = "PI"))
p2 <- p2 + scale_color_manual(values = c("blue", "black", "red"))
p2
```

```
## Warning: Removed 104 rows containing missing values (geom_point).

## Warning: Removed 42 row(s) containing missing values (geom_path).

## Warning: Removed 42 row(s) containing missing values (geom_path).

## Warning: Removed 42 row(s) containing missing values (geom_path).

## Warning: Removed 42 row(s) containing missing values (geom_path).

## Warning: Removed 42 row(s) containing missing values (geom_path).
```



Produce a point estimate for the weaning age of a species whose brain weight is 750 gm

```
(h.hat <- beta1 * log(750) + beta0)
```

```
## [1] 7.150967
```

Produce associated 90% prediction interval for the weaning age of a species whose brain wight is 750 gm.

```
pi2 <- predict(Mod_2,
               newdata = data.frame(Brain_Size_Species_Mean = log(750)),
               interval = "prediction", level = 0.90)
pi2
```

```
##        fit      lwr    upr
## 1 7.150967 6.449135 7.8528
```

```
#3.76 - 5.14
```

Q: Do you trust the model to predict observations accurately for this value of the explanatory variable? Why or why not? A: I would not trust the transformed model to accurately predict observations as the variables are too skewed to be able be able to accuratly predict.

Q: Looking at your two models (i.e., transformed versus log-log transformed), which do you think is better? Why? A: The log-transformed model would be better as the variables in the other model are highly skewed. Log transformation of the data produced a more normal data set and linear regression.

Challenge 2

```
library(infer)
library(car)
```

```
## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode

## The following object is masked from 'package:purrr':
##
##     some
```

```
library(boot)
```

```
##
## Attaching package: 'boot'

## The following object is masked from 'package:car':
##
##     logit
```

```
library(stats)
```

Using bootstrapping, we can also do the same for estimating standard errors and CIs around regression parameters, such as beta coefficients.

Using the "KamilarAndCooperData.csv" dataset, run a linear regression looking at log(MeanGroupSize) in relation to log(Body_mass_female_mean) and report your beta coeffiecients (slope and intercept)

```
head(d) #recall data
```

```
## # A tibble: 6 x 44
##   Scientific_Name Family Genus Species Brain_Size_Spec~ Brain_Size_Fema~
##   <chr>           <chr>  <chr> <chr>              <dbl>            <dbl>
## 1 Allenopithecus~ Cerco~ Alle~ nigrov~             4.06             53.7
## 2 Allocebus_tric~ Cerco~ Allo~ tricho~            NA                NA
## 3 Alouatta_belze~ Ateli~ Alou~ belzeb~             3.97             51.2
## 4 Alouatta_caraya Ateli~ Alou~ caraya              3.96             47.8
## 5 Alouatta_guari~ Ateli~ Alou~ guariba             3.95             49.1
## 6 Alouatta_palli~ Ateli~ Alou~ pallia~             3.91             48.0
## # ... with 38 more variables: Brain_size_Ref <chr>, Body_mass_male_mean <dbl>,
## #   Body_mass_female_mean <dbl>, Mass_Dimorphism <dbl>, Mass_Ref <chr>,
## #   MeanGroupSize <dbl>, AdultMales <dbl>, AdultFemale <dbl>,
## #   AdultSexRatio <dbl>, Social_Organization_Ref <chr>,
## #   InterbirthInterval_d <dbl>, Gestation <dbl>, WeaningAge_d <dbl>,
## #   MaxLongevity_m <dbl>, LitterSz <dbl>, Life_History_Ref <chr>,
## #   GR_MidRangeLat_dd <dbl>, Precip_Mean_mm <dbl>, Temp_Mean_degC <dbl>,
## #   AET_Mean_mm <dbl>, PET_Mean_mm <dbl>, Climate_Ref <chr>,
## #   HomeRange_km2 <dbl>, HomeRangeRef <chr>, DayLength_km <dbl>,
## #   DayLengthRef <chr>, Territoriality <dbl>, Fruit <dbl>, Leaves <chr>,
## #   Fauna <chr>, DietRef1 <chr>, Canine_Dimorphism <dbl>,
## #   Canine_Dimorphism_Ref <chr>, Feed <dbl>, Move <dbl>, Rest <dbl>,
## #   Social <dbl>, Activity_Budget_Ref <chr>
```

```
lmGroupSize <- lm(log(MeanGroupSize) ~ log(Body_mass_female_mean), data = d)
summary(lmGroupSize)
```

```
##
```

```
## Call:
## lm(formula = log(MeanGroupSize) ~ log(Body_mass_female_mean),
##     data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.85428 -0.66210  0.03604  0.63941  2.51998
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -1.77726    0.43752  -4.062 8.06e-05 ***
## log(Body_mass_female_mean)  0.50628    0.05563   9.102 8.08e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9373 on 140 degrees of freedom
##   (71 observations deleted due to missingness)
## Multiple R-squared:  0.3717, Adjusted R-squared:  0.3673
## F-statistic: 82.84 on 1 and 140 DF,  p-value: 8.078e-16
```

```
#slope coefficient
coef <- lmGroupSize$coefficients
(beta1 <- as.numeric(coef[2]))#slope
```

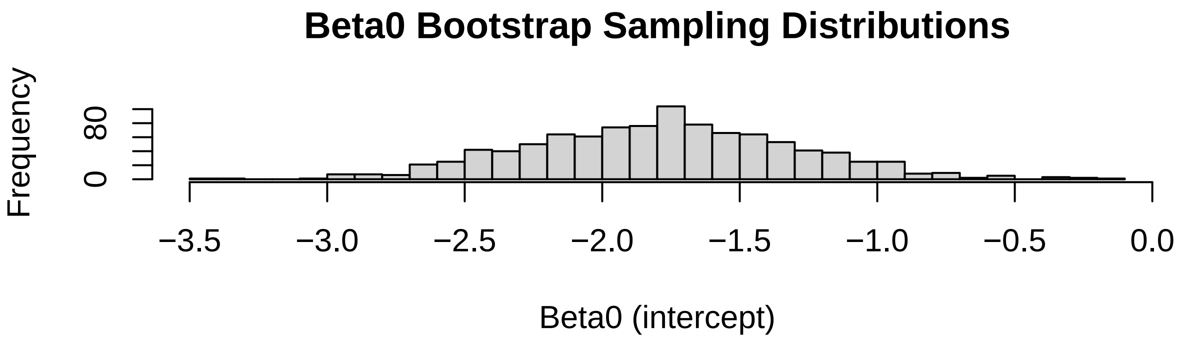
```
## [1] 0.5062813
```

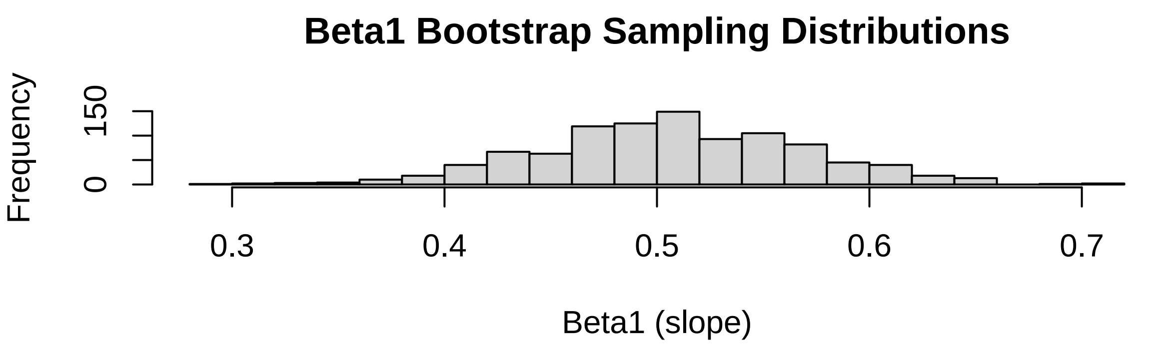
```
(beta0 <- as.numeric(coef[1])) #intercept
```

```
## [1] -1.777258
```

Then, use bootstrapping to sample from the dataset 1000 times with replacement, each time fitting the same model and calculating the appropriate coefficients. [The size of each sample should be equivalent to the total number of observations in the dataset.] This generates a bootstrap sampling distribution for each beta coefficient. Plot a histogram of these sampling distributions for beta0 and beta1.

```
#bootstrap 1000 times
set.seed(5)
bootCoefs <- data.frame(beta0 = 1:1000, beta1 = 1:1000) #store outcome
n <- nrow(d)
for (i in 1:1000) {
  s <- sample_n(d, size = n, replace = T)
  lmboot <- lm(log(MeanGroupSize) ~ log(Body_mass_female_mean), data = s)
  coef <- lmboot$coefficients
  beta0 <- as.numeric(coef[1])
  beta1 <- as.numeric(coef[2])
  bootCoefs$beta0[[i]] <- beta0
  bootCoefs$beta1[[i]] <- beta1
}

#plot a histogram of the sampling distributions for beta 0 and 1
par(mfrow=c(2, 1))
hist(bootCoefs$beta0, breaks = 30, xlab = "Beta0 (intercept)", main = "Beta0 Bootstrap Sampling Distribu
hist(bootCoefs$beta1, breaks = 30, xlab = "Beta1 (slope)", main = "Beta1 Bootstrap Sampling Distributio
```





Estimate the standard error for each of your beta coefficients as the standard deviation of the sampling distribution from your bootstrap.

```
(beta1SE <- sd(bootCoefs$beta1)) #beta1
```

```
## [1] 0.06187259
```

```
(beta0SE <- sd(bootCoefs$beta0)) #beta0
```

```
## [1] 0.4887741
```

```
(BootSE <- cbind(beta0SE, beta1SE)) #combine both values
```

```
##        beta0SE    beta1SE
## [1,] 0.4887741 0.06187259
```

Also determine the 95% CI for each of your beta coefficients based on the appropriate quantiles from your sampling distribution.

```
alpha <- 0.05
beta0lower <- quantile(bootCoefs$beta0, alpha/2)
beta0upper <- quantile(bootCoefs$beta0, 1 - (alpha/2))
beta1lower <- quantile(bootCoefs$beta1, alpha/2)
beta1upper <- quantile(bootCoefs$beta1, 1 - (alpha/2))
(Bootstrapquantiles <- cbind(beta0lower, beta0upper, beta1lower, beta1upper))
```

```
##      beta0lower beta0upper beta1lower beta1upper
## 2.5%  -2.690826 -0.8501434  0.3845505  0.6254107
```

How do the SEs estimated from the bootstrap sampling distribution compare to those estimated mathematically as part of lm() function?

A: They are very similar.

```
m <- tidy(lmGroupSize)
m$std.error #model
```

```
## [1] 0.43751572 0.05562507
```

```
b <- BootSE #bootstrap dist.

rbind(b, m$std.error)
```

```
##        beta0SE    beta1SE
## [1,] 0.4887741 0.06187259
## [2,] 0.4375157 0.05562507
```

```
#bootstrap values are slightly higher than the model values
```

How do your bootstrap CIs compare to those estimated mathematically as part of the lm() function?

A: Also very similar.

```
CImodel <- confint(lmGroupSize, level = 1 - alpha)
CImodel_lower <- CImodel[, 1]
CImodel_upper <- CImodel[, 2]

(cbind(CImodel_lower, CImodel_upper))
```

```
##                            CImodel_lower CImodel_upper
## (Intercept)                   -2.6422501    -0.9122659
## log(Body_mass_female_mean)     0.3963075     0.6162550
```

```
Bootstrapquantiles
```

```
##      beta0lower beta0upper beta1lower beta1upper
## 2.5%  -2.690826 -0.8501434  0.3845505  0.6254107
```

Challenge 3

Write your own function, called boot_lm(), that takes as its arguments a dataframe (d=), a linear model (model=, written as a character string, e.g., "logGS ~ logBM"), a user-defined confidence interval level (conf.level=, with default "0.95"), and a number of bootstrap replicates (reps=, with default "1000").

```
#model arguments
boot_lm <- function(d, model, conf.level = 0.95, reps = 1000)  {
  df <- data.frame(Coefficient = c("Beta0", "Beta1"), Coefficientvalue = c(0,0), SE = c(0,0), UpperCI =
  m <- lm(eval(parse(text = model)), data = d)#linear model using the model and d inputs
  mtidy <- tidy(m)
  df$Coefficientvalue[1] <- as.numeric(mtidy[1,2])
  df$Coefficientvalue[2] <- as.numeric(mtidy[2,2])
  df$SE[1] <- as.numeric(mtidy[1,3])
  df$SE[2] <- as.numeric(mtidy[2,3])
  modelCI <- confint(m, level = conf.level)
  df$UpperCI[1] <- modelCI[1,2]
  df$UpperCI[2] <- modelCI[2,2]
  df$LowerCI[1] <- modelCI[1,1]
  df$LowerCI[1] <- modelCI[2,1]
  set.seed(5) #for bootstrap
  bootstrap <- data.frame(beta0 = 1:reps, beta1 = 1:reps)
  n <- nrow(d)
  for (i in 1:reps){
```

```r
    s <- sample_n(d, size = n, replace = T)
    mboot <- lm(eval(parse(text = model)), data = s)
    Bootlm <- mboot$coefficients
    beta0 <- as.numeric(Bootlm[1])
    beta1 <- as.numeric(Bootlm[2])
    bootstrap$beta0[[i]] <- beta0
    bootstrap$beta1[[i]] <- beta1
  }
  df$MeanBetaBoot[1] <- mean(bootstrap$beta0)
  df$MeanBetaBoot[2] <- mean(bootstrap$beta1)
  df$SEBoot[1] <- sd(bootstrap$beta0)
  df$SEBoot[2] <- sd(bootstrap$beta1)
  alpha <- 1 - conf.level
  df$UpperCIBoot[1] <- quantile(bootstrap$beta0, 1 - (alpha/2))
  df$UpperCIBoot[2] <- quantile(bootstrap$beta1, 1 - (alpha/2))
  df$LowerCIBoot[1] <- quantile(bootstrap$beta0, alpha/2)
  df$LowerCIBoot[2] <- quantile(bootstrap$beta1, alpha/2)
  df
}

a <- boot_lm(d, "log(MeanGroupSize) ~ log(Body_mass_female_mean)")
a
```

```
##   Coefficient Coefficientvalue         SE    UpperCI   LowerCI MeanBetaBoot
## 1       Beta0       -1.7772580 0.43751572 -0.9122659 0.3963075    -1.785919
## 2       Beta1        0.5062813 0.05562507  0.6162550 0.0000000     0.507383
##       SEBoot UpperCIBoot LowerCIBoot
## 1 0.48877408  -0.8501434  -2.6908259
## 2 0.06187259   0.6254107   0.3845505
```

```r
b <- boot_lm(d, "log(DayLength_km) ~ log(Body_mass_female_mean)")
b
```

```
##   Coefficient Coefficientvalue        SE   UpperCI     LowerCI MeanBetaBoot
## 1       Beta0       -0.1427624 0.4650068 0.7795764 -0.07347022  -0.13683825
## 2       Beta1        0.0389422 0.0566739 0.1513546  0.00000000   0.03791644
##       SEBoot UpperCIBoot LowerCIBoot
## 1 0.48780057   0.7656472 -1.13451781
## 2 0.06153541   0.1608548 -0.07604676
```

```r
c <- boot_lm(d, "log(DayLength_km) ~ log(Body_mass_female_mean) + log(MeanGroupSize)")
c
```

```
##   Coefficient Coefficientvalue         SE    UpperCI    LowerCI MeanBetaBoot
## 1       Beta0      -0.29916271 0.48590764 0.66663134 -0.2157728  -0.26437133
## 2       Beta1      -0.08813683 0.06421586 0.03949915  0.0000000  -0.09409768
##       SEBoot UpperCIBoot LowerCIBoot
## 1 0.46010210  0.62590709  -1.1245216
## 2 0.05800141  0.01554985  -0.2110745
```