Week 1 assignment: linear models

Example solutions

January 15, 2015

Loading the relevant R packages

To complete these problems, you'll need some R packages loaded and ready to go. We suggest starting with a package for plotting (ggplot2), and potentially some packages for manipulating data frames (dplyr and tidyr), depending on how you prefer to use R.

```
library(ggplot2)
library(dplyr)
library(tidyr)
```

Problem 1

You've discovered a single adult of a new species of amniote. Though you don't know the sex, you're worried that it might lay eggs soon and you want to build something to protect them. To do so, you need to predict the mass of this new creature's eggs. You first weigh your amniote and find that it weighs 3000 grams. Luckily, some researchers have just recently published a full database of aminote life history traits, which can give you some information about how amniote adult mass relates to amniote egg mass. (Myhrvold et al. (2015))

Loading the data

Your first task will be to acquire the published dataset, which is available here. Download the data file `Amniote_Database_Aug_2015.csv` and save it in a location that makes sense (e.g., `~/hmods/class1/data/`). Then, you'll need to load the data with the `read.csv()` function. Do not use `file.choose()`.

```
dat <- read.csv("../data/Amniote_Database_Aug_2015.csv")
```

Preparing the data

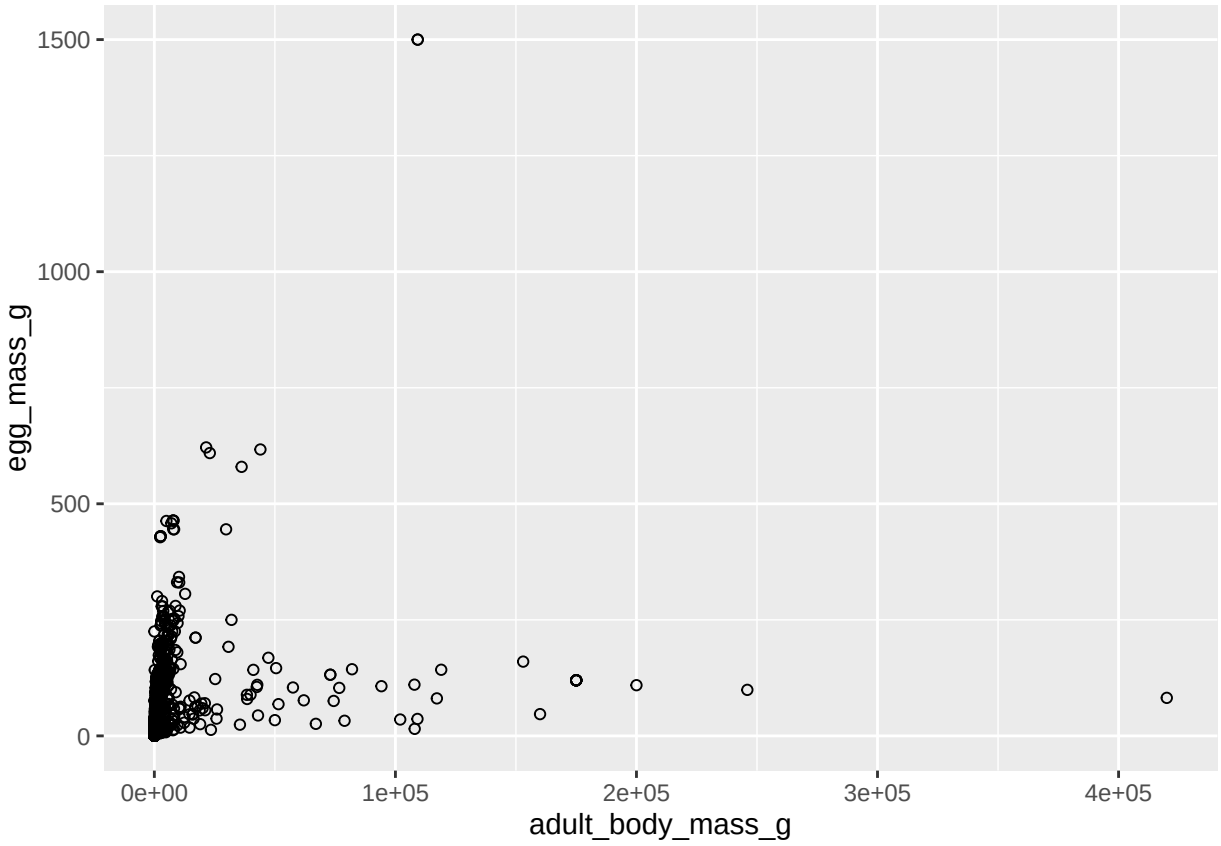
In this dataset, missing values are coded as `-999`. We want to replace these values with `NA` which indicates missing values to R.

```
dat[dat == -999] <- NA
```

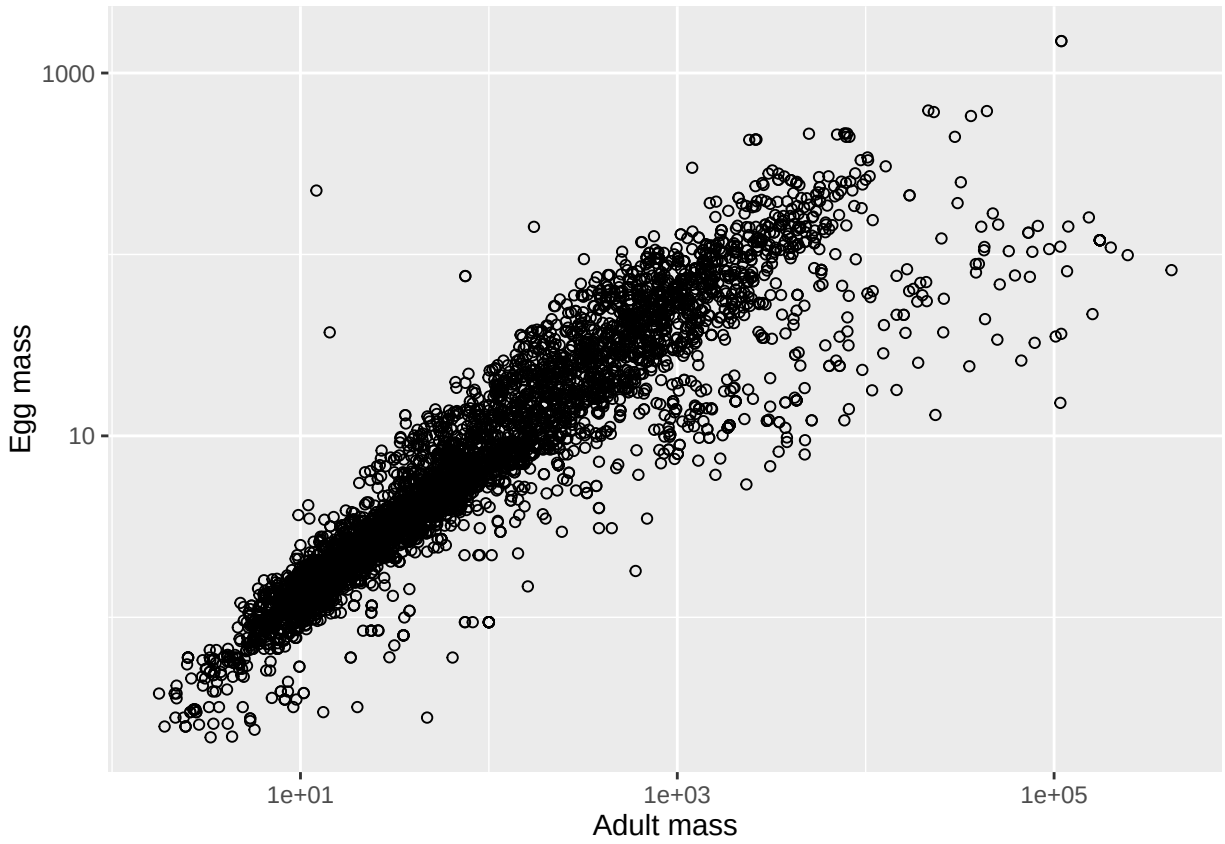
Visualizing the data

We want to predict egg mass from adult body mass. Visualize the relationship between these two variables below. Transformation will be helpful for both variables.

```r
# remove points without the data we want
dat_use <- dat %>%
  filter(!is.na(adult_body_mass_g) & !is.na(egg_mass_g))

# plot the raw data
ggplot(dat_use, aes(adult_body_mass_g, egg_mass_g)) +
  geom_point(shape = 1) # plots data as points
```




```r
# plot transformred
ggplot(dat_use, aes(adult_body_mass_g, egg_mass_g)) +
  geom_point(shape = 1) +
  xlab('Adult mass') + # specifies axis labels
  ylab('Egg mass') +
  scale_y_log10() +
  scale_x_log10()
```



Modeling the data

Use the `lm()` function to construct a linear model that could be used to predict egg mass based on adult body mass. (Hint: what other kinds of data transformations might be helpful prior to fitting the model?)

```
# create a log transformed input variable
dat_use <- dat_use %>%
  mutate(lmass = log(adult_body_mass_g),
         legg = log(egg_mass_g))

# fit the model
model <- lm(legg ~ lmass, data = dat_use)
summary(model)
```

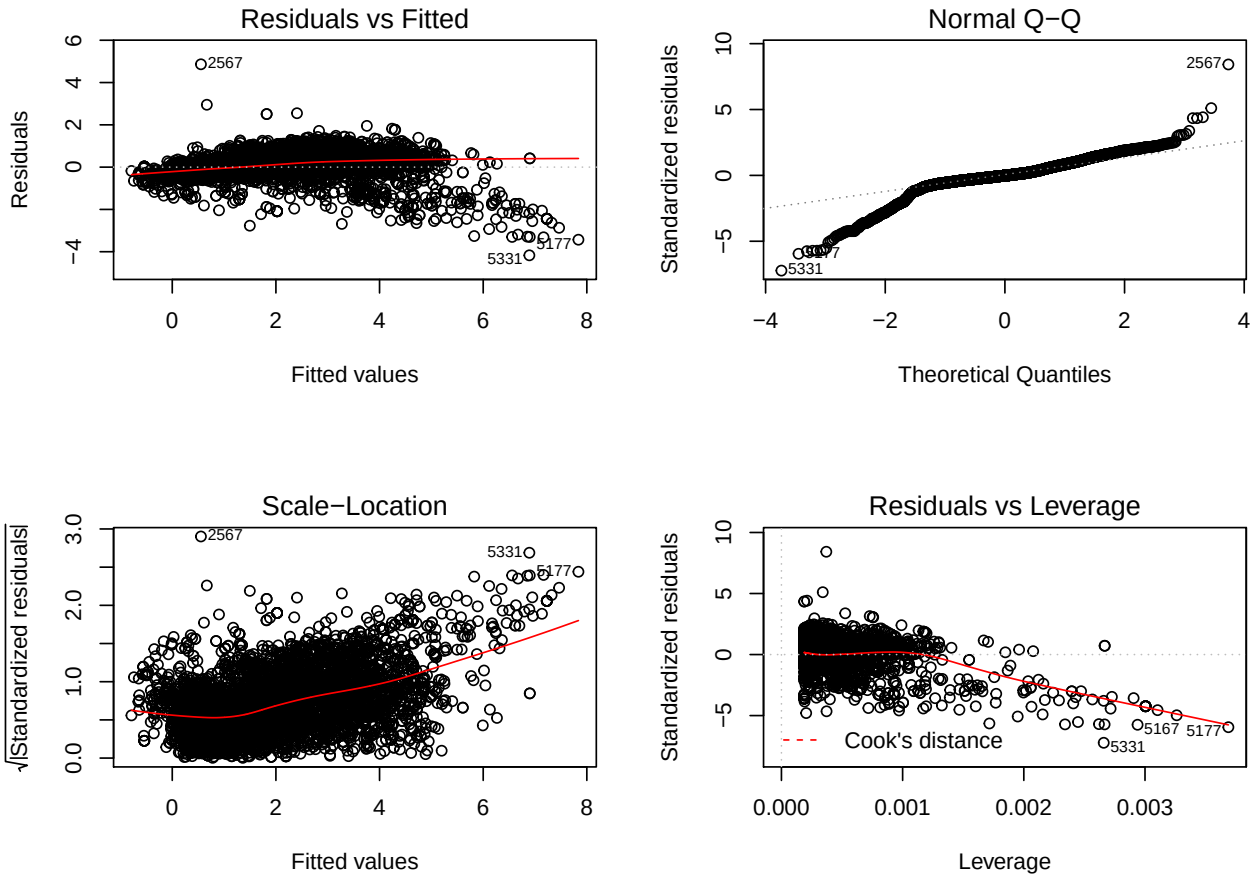
```
##
## Call:
## lm(formula = legg ~ lmass, data = dat_use)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1699 -0.2121 -0.0101  0.2873  4.8626
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) -1.187244   0.019524  -60.81   <2e-16 ***
## lmass        0.697026   0.004018  173.47   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5776 on 5336 degrees of freedom
## Multiple R-squared:  0.8494, Adjusted R-squared:  0.8494
## F-statistic: 3.009e+04 on 1 and 5336 DF,  p-value: < 2.2e-16
```

```
# note that we could center lmass via scale(lmass), but not necessary here
```

Evaluate the homoscedasticity and normality assumptions graphically (e.g., `plot(mymodel)`).

```
par(mfrow=c(2, 2))
plot(model)
```



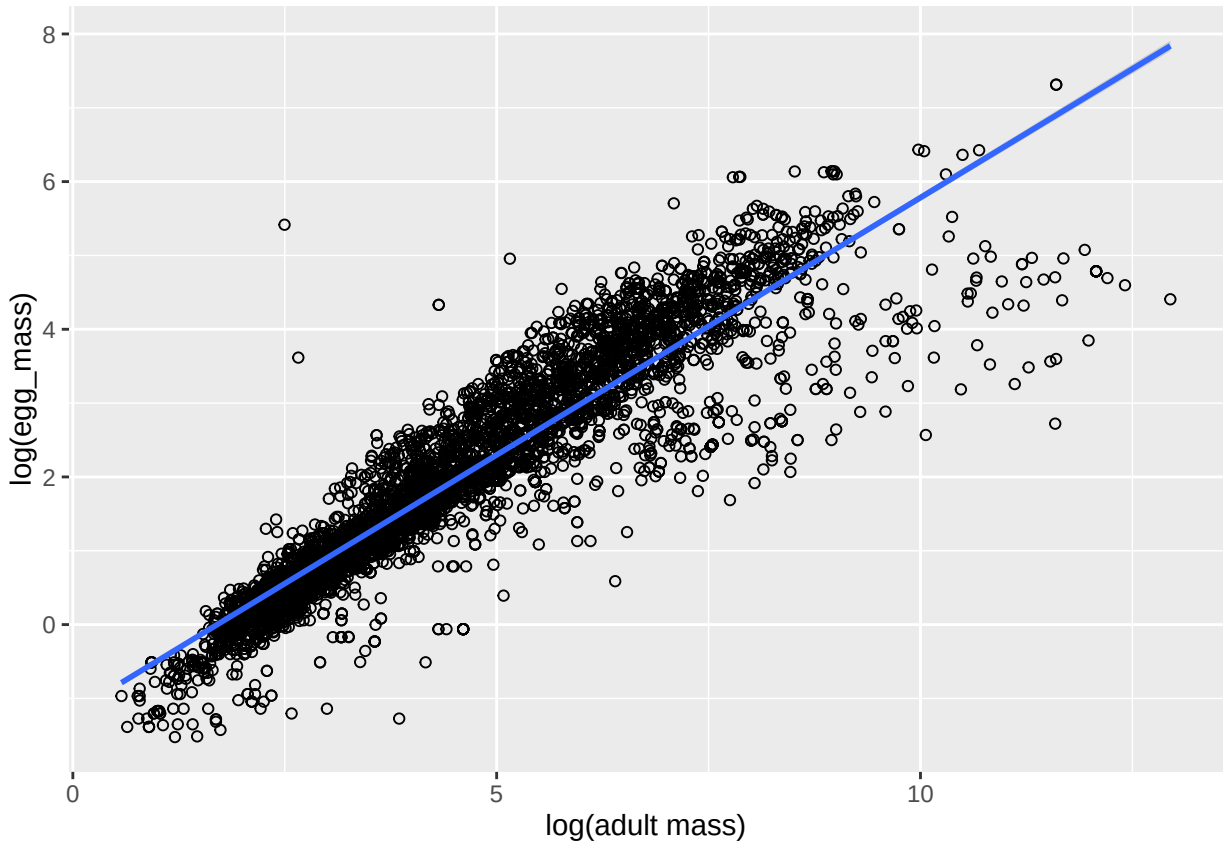
Are the assumptions met? Why or why not?

The downward curve in the Q-Q plot shows that the residuals are becoming consistently negative for larger values of adult mass. This suggests that we might benefit from adding a polynomial (i.e. squared) term to the model, though predictions for most values of adult mass will likely be pretty good. Nothing else is overly concerning, though the Q-Q plot is a bit wonky on the ends.

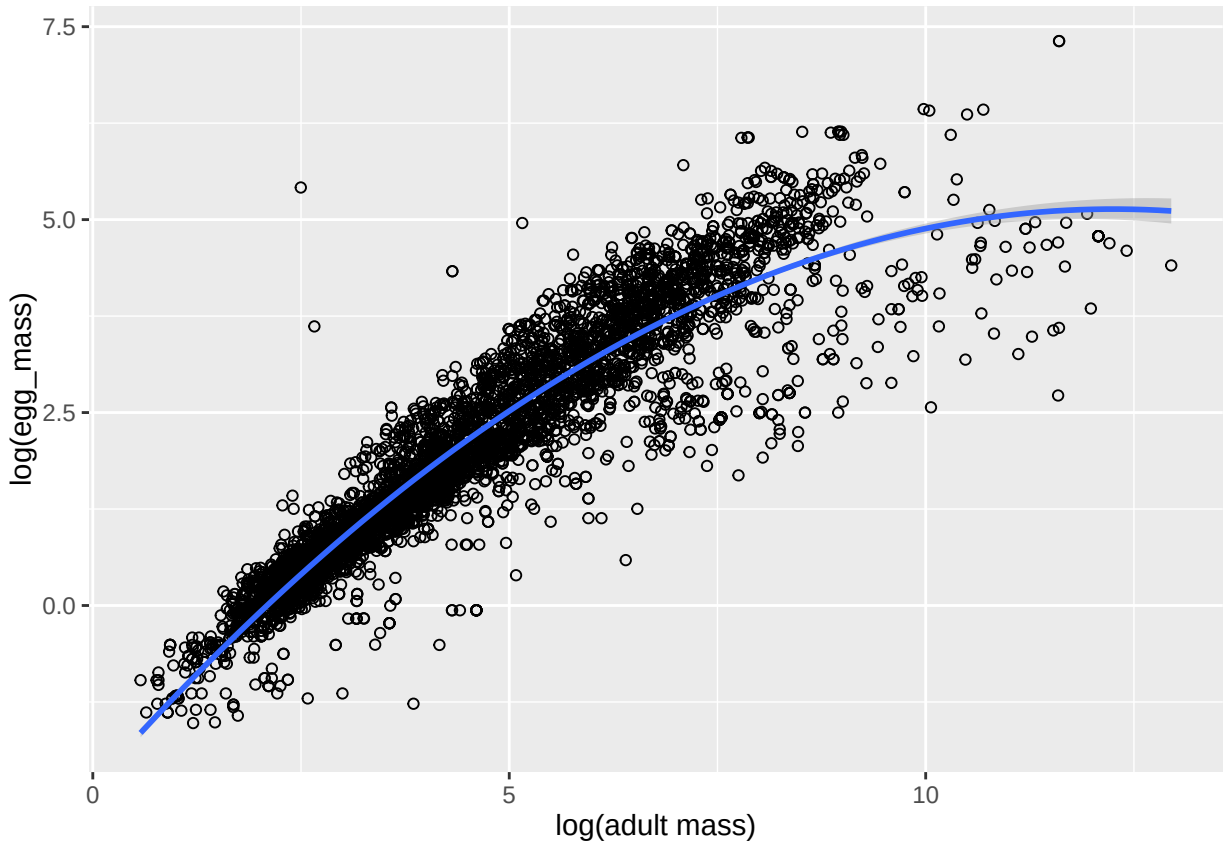
Understanding the model

Produce a scatterplot as before, but this time add a trendline that represents the expected value of the response as a function of the predictor.

```r
# plot with best fit linear trendline
ggplot(dat_use, aes(lmass, legg)) +
  geom_point(shape = 1) +
  geom_smooth(method = "lm") +  # adds line via lm()
  labs(x = "log(adult mass)", y = "log(egg_mass)")
```



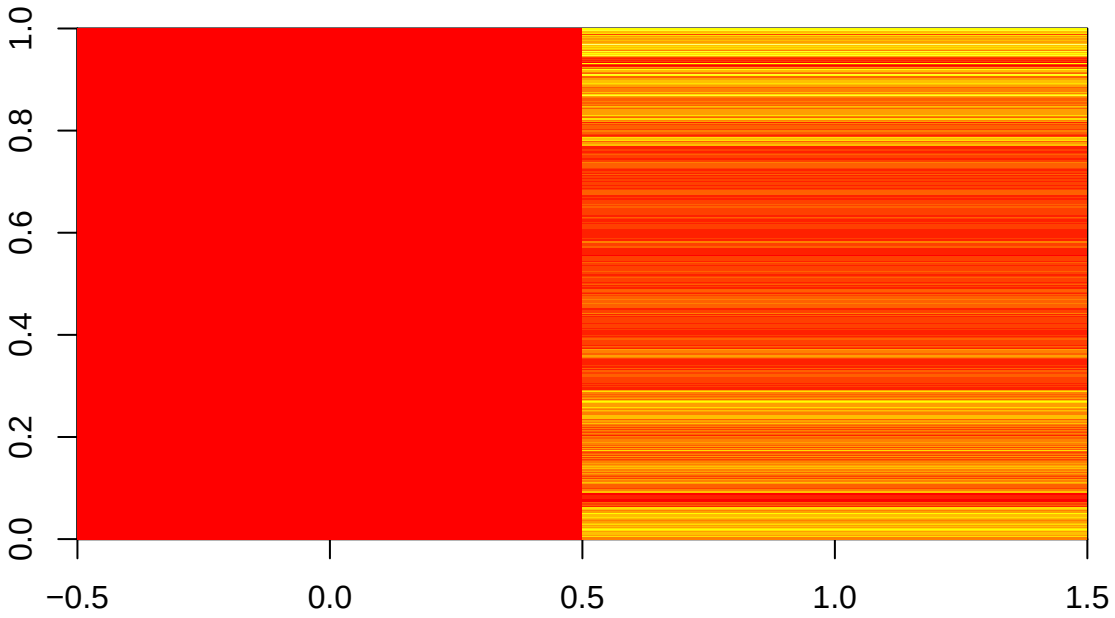

```r
# plot with polynomial trendline
ggplot(dat_use, aes(lmass, legg)) +
  geom_point(shape = 1) +
  geom_smooth(method = "lm", formula = y ~ poly(x,2)) +
  labs(x = "log(adult mass)", y = "log(egg_mass)")
```




```
# note that geom_smooth is now calling lm() with the log
# squared adult body mass included in the formula
```

Make an image plot of the design matrix for your model (e.g., `image(t(model.matrix(m)))`):

```
par(mfrow=c(1, 1))
# below, I use piping to avoid "onion" code
# but the result is the same as image(t(model.matrix(model)))
model %>%
  model.matrix() %>%
  t() %>%
  image()
```



Why does this image plot look the way it does, and what is the result of multiplying the design matrix by the vector of estimated coefficients (e.g., `model.matrix(m) %*% coef(m)`)?

The first column in the matrix is all ones, and corresponds to the intercept (or global mean egg mass). The second column indicates the adult mass values for each individual. Multiplying the model matrix by the given model coefficients yields the predicted values for each individual - aka the linear predictor.

Predicting egg mass for the new critter

Predict the egg mass for the new species and provide upper and lower bounds on this estimate, in units of grams. Remember that interval should incorporate both *predictive* uncertainty (error term in the model) and inferential uncertainty (uncertainty about the coefficients and amount of residiual error) (Hint: there's a built-in R function that should help generate prediction intervals)

```
# predict the new value
new_logmass <- log(3000) # model uses log values

# predict() is a useful function (see ?predict for syntax)
pred <- predict(model, data.frame(lmass = new_logmass),
                se.fit=TRUE, interval = 'prediction')

# convert prediction from log grams to grams scale
exp(pred$fit)
```

```
##        fit      lwr      upr
## 1 80.91532 26.06418 251.1987
```

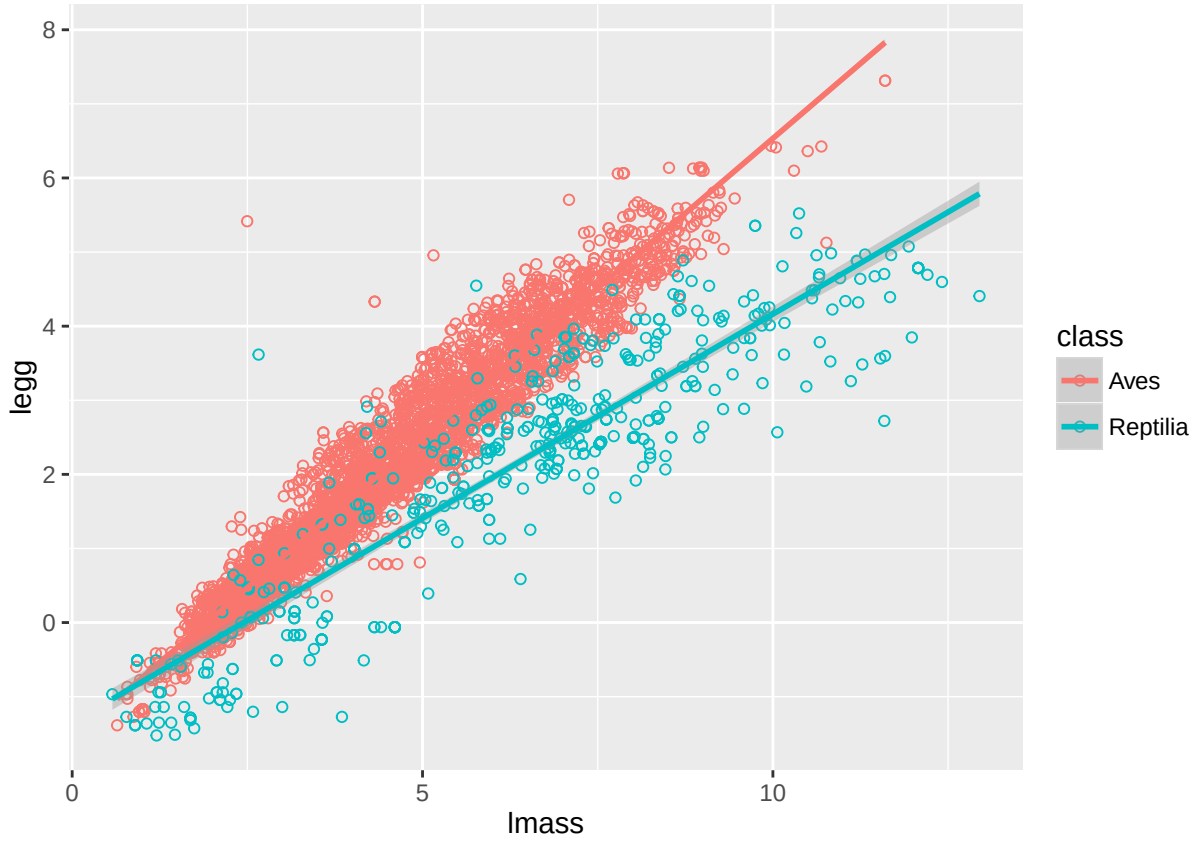
Problem 2

A week later, you are told that the critter has been identified to be in the class Reptilia. Use this new information to update your prediction in the code block below, commenting your code to document your thought process.

```
# how many classes are there?
table(dat_use$class)
```

```
##
##      Aves Mammalia Reptilia
##      4843        0      495
```

```
# does the relationship between adult mass and egg mass differ for the two classes?
ggplot(dat_use, aes(lmass, legg, color = class)) +
  geom_point(shape = 1) +
  stat_smooth(method = "lm")
```

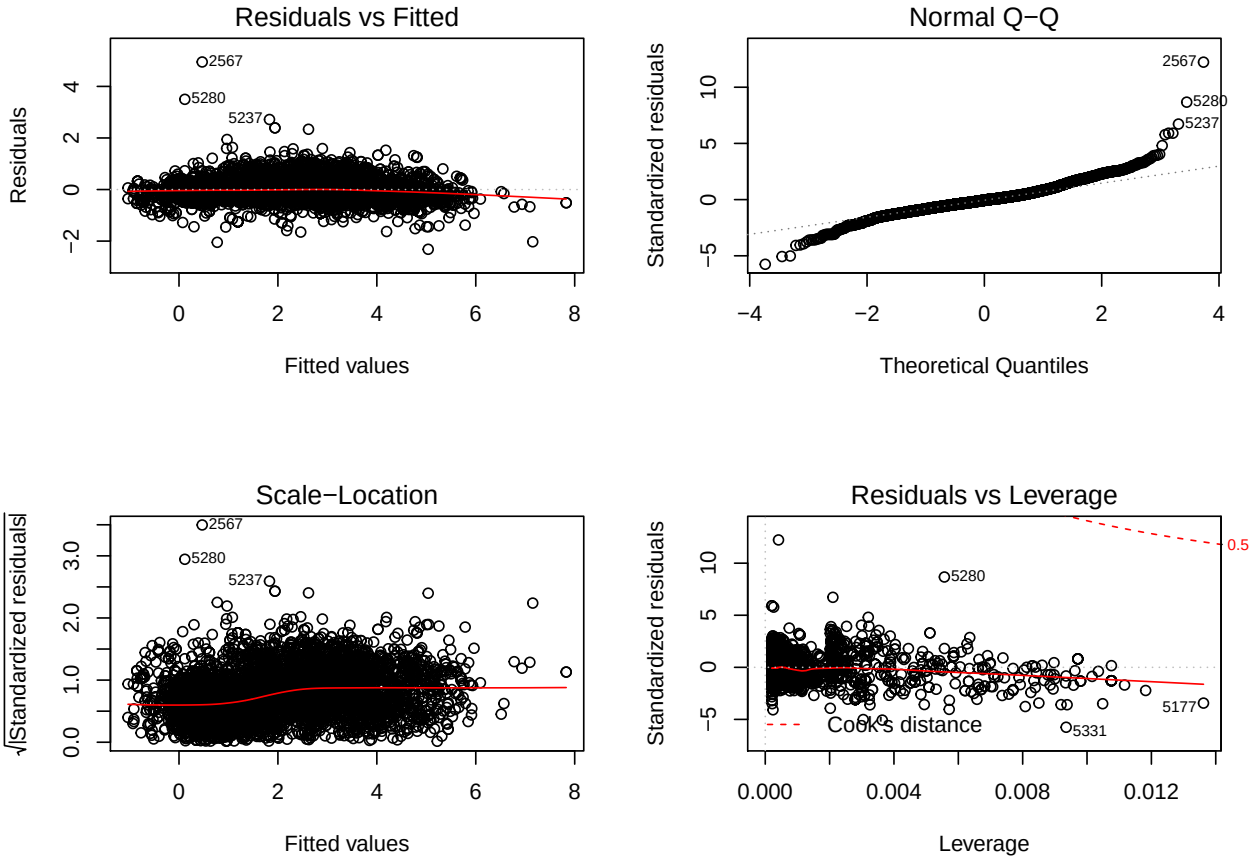



```
# fit a new model
model2 <- lm(legg ~ lmass * class, data = dat_use)
summary(model2)
```

```
##
## Call:
## lm(formula = legg ~ lmass * class, data = dat_use)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.3169 -0.2292 -0.0238  0.1846  4.9470
##
```

```
## Coefficients:
##                       Estimate Std. Error  t value Pr(>|t|)
## (Intercept)         -1.549718   0.015218 -101.833  < 2e-16 ***
## lmass                0.808376   0.003309  244.305  < 2e-16 ***
## classReptilia        0.201718   0.047878    4.213 2.56e-05 ***
## lmass:classReptilia -0.257363   0.007363  -34.953  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4046 on 5334 degrees of freedom
## Multiple R-squared:  0.9261, Adjusted R-squared:  0.9261
## F-statistic: 2.229e+04 on 3 and 5334 DF,  p-value: < 2.2e-16
```

```
# evaluate diagnostic plots
par(mfrow=c(2, 2))
plot(model2)
```




```
# new prediction
pred2 <- predict(model2, data.frame(lmass = new_logmass, class = "Reptilia"),
  se.fit=TRUE, interval = 'prediction')

# convert to grams
exp(pred2$fit)
```

```
##       fit      lwr      upr
```

```
## 1 21.4048 9.673022 47.3653
```

```
# compare on the natural scale (grams)
round(rbind(exp(pred$fit), exp(pred2$fit)), digits = 1)
```

```
##    fit  lwr   upr
## 1 80.9 26.1 251.2
## 1 21.4  9.7  47.4
```

```
# compare on log scale
round(rbind(pred$fit, pred2$fit), digits = 1)
```

```
##   fit lwr upr
## 1 4.4 3.3 5.5
## 1 3.1 2.3 3.9
```

How does your new prediction compare to your prediction from Problem 1 in terms of accuracy and precision? Is it lower or higher, and why?

No idea about the accuracy (because we don't know what the actual underlying value is), but the prediction interval is a bit narrower, so the precision has increased a bit. The new prediction is lower, because the slope is different for reptiles vs. birds.

Problem 3

Myrdahl et al. just retracted all of the adult mass data from their data set, and have advised researchers to stop using the existing adult mass data until further notice! Given this new development, what's your best prediction for the critter's egg mass? Update your prediction in the block below, commenting the code as necessary.

```
# our best guess will be to use the mean egg size for Reptilia,
# we can estimate this using a liner model
model3 <- lm(legg ~ 0 + class, data = dat_use)

# prediction interval
pred3 <- predict(model3, data.frame(class = "Reptilia"),
  se.fit=TRUE, interval = 'prediction')

# compare to the previous fit
round(rbind(exp(pred$fit), exp(pred2$fit),exp(pred3$fit)),digits = 1)
```

```
##    fit  lwr   upr
## 1 80.9 26.1 251.2
## 1 21.4  9.7  47.4
## 1  8.5  0.5 156.6
```

Ouch! Our predictive ability really takes quite a hit when we don't have any information about adult body mass!

Bonus Problem (optional)

When predicting the egg mass value for your unknown amniote, you probably used a built-in function in R (i.e. *predict*) to automatically generate prediction intervals. Can you generate prediction intervals from the model without resorting to a built-in *predict* fuction?
(Hint 1: how many parameters were estimated?)
(Hint 2: can prediction intervals be simulated?)
(Hint 3: check out chapter 7 of Gelman and Hill if you have it)

```
# first we need to use the uncertainty in the fitted parameters to simulate a
# range of new parameter values consistent with the data we have.

# to do so we will use the sim() function in the 'arm' package

# load the package
library(arm)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

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

```
## Loading required package: lme4
```

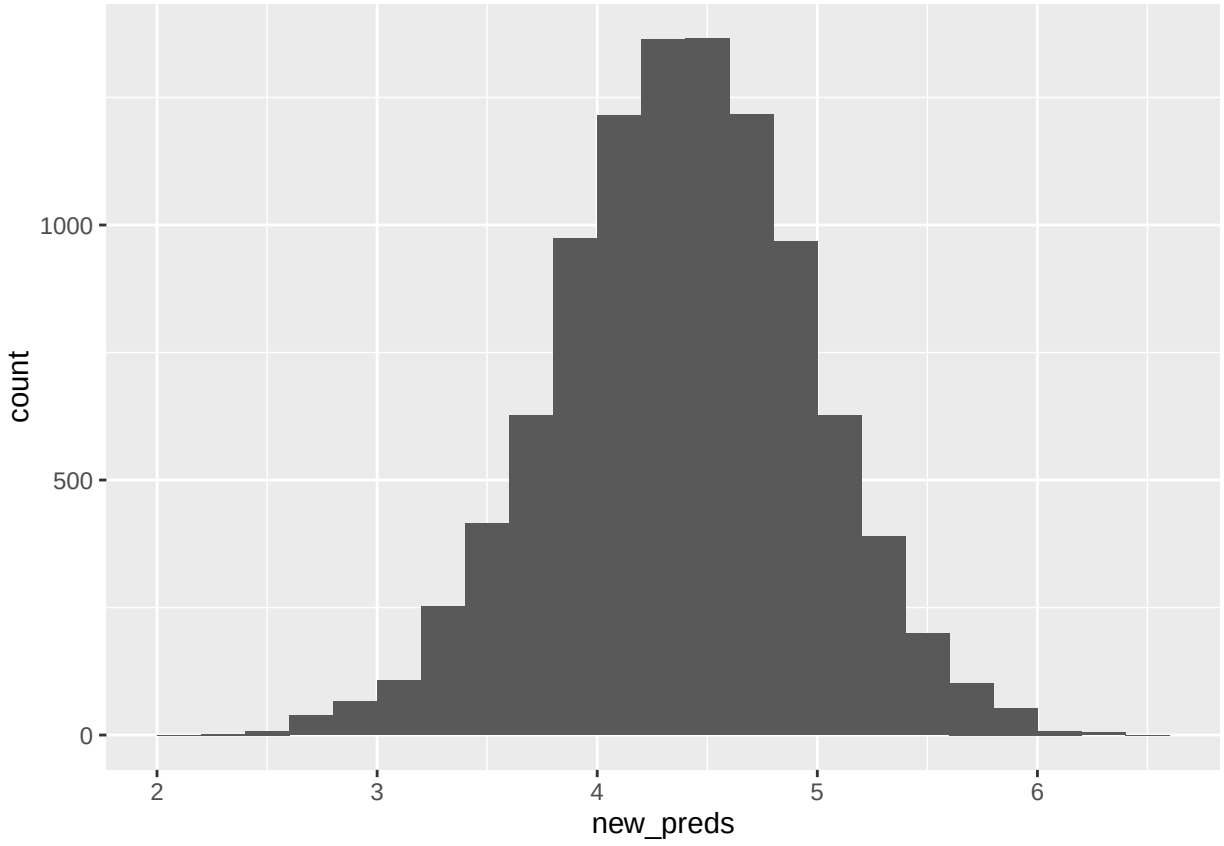
```
##
## arm (Version 1.8-6, built: 2015-7-7)
```

```
## Working directory is /home/max/Documents/classes/hmods/hmods_secret/wk1_linmods/solutions
```

```
# simulate new values for each parameter using sim()
n.sims <- 10000
sim.1 <- sim(model,n.sims)
sim_params <- data.frame(alpha = sim.1@coef[,1], beta = sim.1@coef[,2],
  sigma = sim.1@sigma)

# what does sim_params look like?
head(sim_params)
```

```
##       alpha      beta     sigma
## 1 -1.215383 0.7052854 0.5664709
## 2 -1.164897 0.6920203 0.5808202
## 3 -1.178713 0.6955515 0.5699780
## 4 -1.203548 0.6992007 0.5854688
## 5 -1.186762 0.6936801 0.5839987
## 6 -1.194038 0.6962553 0.5775487
```

```r
  # okay so it's generated a three column data frame with one row per simulation
  # add one column per fitted parameter in the model

## calculate prediction for our critter for each simulation
new_preds <- rnorm(n = n.sims,
                   mean = sim_params$alpha + sim_params$beta * new_logmass,
                   sd = sim_params$sigma)

# what does the simulated distribution of predicted values look like for our
# new creature?
qplot(new_preds, geom = "histogram", binwidth = 0.2)
```




```r
# calculate new 95% prediction intervals
pred4 <- data.frame(fit = mean(new_preds),
    lwr = quantile(new_preds, probs = 0.025),
    upr = quantile(new_preds, probs = 0.975),
    row.names = "simulated prediction")
```

```r
# how does it compare to our original *predict()* prediction?
round(rbind(pred$fit,pred4), digits = 3)
```

```
##                         fit   lwr   upr
## 1                     4.393 3.261 5.526
## simulated prediction 4.383 3.237 5.503
```

```r
  # nice!
```