Nonlinear Regression and Heteroscedasticity

Statistics 516, Homework 2

This homework assignment concerns specifying and making inferences from nonlinear regression models, and methods for accounting for heteroscedasticity.

Instructions

1. This assignment is due by 11:59 PM on Wednesday, March 8th. Email me your homework at trjohns@uidaho.edu. Please submit your homework as a PDF file. Late assignments will be penalized by 10% if turned-in within 12 hours of the deadline, and 10% more for each additional 12 hour interval.
2. Your solutions must be **typed** and **very** neatly organized. I will not try to infer your solutions if they are not clearly presented. Mathematical expressions need not be typeset perfectly but they should be clear. You may substitute letters for symbols (e.g., b1 for β_1) and use other shortcuts for mathematical notation if no meaning is lost.
3. You must include with your solutions the relevant R output **and** R code that created them. Be sure that you provide sufficient code that I can replicate your results. Include both the code and the output within the text of your solutions (not in an appendix) using cut-and-paste. Edit your output so as to provide only that which is relevant to answering the questions. Use a monospace font (e.g., Courier or Monoco) for R code and output for clarity. Do not use a monospace font for text that is not R code or output. You can use the R command `options(digits = 4)` (or some other small number of digits) to exert some control over the width of the output by reducing the number of digits shown by R.
4. Plots from R Studio can be exported in various formats or directly to the clipboard using the "export" menu in the top-left part of the plot panel.
5. It is permitted for you to discuss the homework with other students in the course. However your work including R code, output, and written answers must be your own.
6. You are very welcome to ask me questions. I will be happy to clarify what I am asking in any of the questions and will provide you some help with solving problems by showing you how to work through similar problems from class. I will also be open to helping with any R problems. If you email me with a R question, it will usually be helpful for you to include enough of your R script so that I can replicate your issue. But please avoid saving all your questions for just before the assignment is due. I can usually respond quickly to questions, but I will sometimes need time to respond.

Using `nls` for Linear Models

Both the `nls` and the `lm` functions can estimate *linear* regression models, but their interfaces are different. The `lm` function allows us to specify a model *symbolically* whereas `nls` requires us to specify the model *mathematically*. In practice we usually use `lm` for linear models because it takes care of some of the intricacies of specifying the model (e.g., indicator variables and interactions), but in some cases it can be useful to use `nls`, particularly if the linear model you are using has a peculiar parameterization. But I think it can be a useful exercise for the student to have experience using `nls` to specify linear models.[1] Here you will use `nls` to estimate some of the models you encountered in the previous homework assignment. Note that since all of these models are linear, you need not worry about specifying good starting values. You can safely specify values of zero for all starting values.

1. The last homework assignment considered a couple of parameterizations of models for the `Dopamine` data from the **BSDA** package. Below the models are estimated using the `lm` function. See the solutions for the previous homework to see how these models can be written mathematically.

```
library(BSDA)
m1 <- lm(dbh ~ group, data = Dopamine)
summary(m1)$coefficients
```

```
               Estimate Std. Error t value  Pr(>|t|)
(Intercept)      164.27      12.59  13.052 4.059e-12
grouppsychotic    78.33      19.90   3.936 6.587e-04
```

```
m2 <- lm(dbh ~ -1 + group, data = Dopamine)
summary(m2)$coefficients
```

```
                  Estimate Std. Error t value  Pr(>|t|)
groupnonpsychotic    164.3      12.59   13.05 4.059e-12
grouppsychotic       242.6      15.41   15.74 8.320e-14
```

   For each of these two models, use the `nls` function to estimate the model parameters once by specifying an indicator variable within the model itself (using the `==` operator), and again by using *either* the `ifelse` function *or* the `case_when` function from the **dplyr** package. Show the output of summary for all of these estimated models to verify that you obtained the same results (there may be minor differences several places after the decimal). Be sure the estimates are in the same order in the output of `summary`. This can be controlled by the order in which the starting values are specified to `nls`.

2. Another problem in the last homework concerned the `rat` data from the **ALA** package and featured a couple of different models. Again, see the solutions for the previous homework to see how these models can be written mathematically.[2]

```
m <- lm(weight ~ treatment + week + treatment:week, data = ALA::rat)
summary(m)$coefficients
```

```
                         Estimate Std. Error t value  Pr(>|t|)
(Intercept)               52.8800      2.648 19.9694 2.758e-41
treatmentthiouracil        4.8200      3.745  1.2871 2.004e-01
treatmentthyroxin         -0.7943      4.127 -0.1925 8.477e-01
week                      26.4800      1.081 24.4944 2.373e-50
treatmentthiouracil:week  -9.4300      1.529 -6.1680 8.257e-09
treatmentthyroxin:week     0.6629      1.685  0.3935 6.946e-01
```

[1] I have sometimes toyed with the idea of *starting* the class by teaching students to use `nls` rather than `lm` to help them better appreciate and understand what `lm` is doing for them.

[2] Note that here you may want to use `ALA::rat` to refer to the data frame because there is a data frame of the same name in the **alr3** package which you will be using in a later problem. If you load the **alr3** package after the **ALA** package during the same R session then you run into a name conflict. Using `ALA::rat` avoids that problem.

```
m <- lm(weight ~ treatment:week, data = ALA::rat)
summary(m)$coefficients
```

```
                          Estimate Std. Error t value  Pr(>|t|)
(Intercept)                  54.46     1.6141   33.74 1.829e-66
treatmentcontrol:week        25.95     0.8248   31.47 6.268e-63
treatmentthiouracil:week     18.13     0.8248   21.98 8.733e-46
treatmentthyroxin:week       26.35     0.9207   28.62 3.247e-58
```

As in the previous problem, use the `nls` function to estimate the model parameters for each model once by specifying an indicator variable within the model itself (using the `==` operator), and again by using the `case_when` function from the **dplyr** package (the `ifelse` function could be used, but is cumbersome for more than two cases since it requires multiple and nested `ifelse` statements). Show the output of summary for all of these to verify that you obtained the same results.

Another Michaelis-Menten Model

This problem features using a Michaelis-Menten model similar to that featured in class. The data frame `inhibitor` from the **isdals** package is from an experiment conducted by students in a biochemistry course at the University of Copenhagen.[3] The first few observations can be seen below.

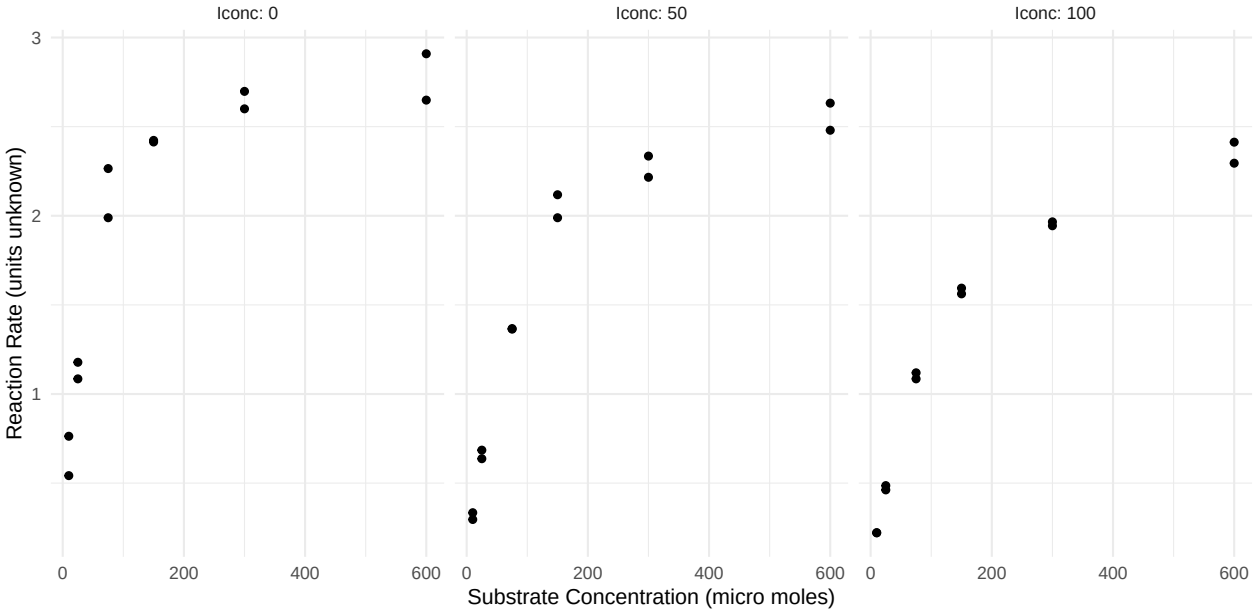
```
library(isdals)
data(inhibitor) # required for this package to make the data available
head(inhibitor)
```

```
  Iconc Sconc    RR
1     0    10 0.542
2     0    10 0.763
3     0    25 1.178
4     0    25 1.085
5     0    75 2.265
6     0    75 1.989
```

The variables `Sconc` and `RR` are the substrate concentration and reaction rate, respectively. As in a typical experiment using this model, assays were conducted at several substrate concentrations and the reaction rate was observed. But this experiment also featured introducing an inhibitor at concentrations of 0, 50, and 100 μmoles. The variable `Iconc` is the inhibitor concentration. The plot below shows the raw data.

```
library(ggplot2)
p <- ggplot(inhibitor, aes(x = Sconc, y = RR)) +
  theme_minimal() + geom_point() +
  facet_wrap(~Iconc, labeller = label_both) +
  labs(x = "Substrate Concentration (micro moles)",
    y = "Reaction Rate (units unknown)")
plot(p)
```

[3]Source: Ekstrom, C. T. & Sorensen, H. (2010). *Introduction to statistical data analysis for the life sciences.* CRC Press.



Recall that the basic Michaelis-Menten regression model can be written as

$$E(R_i) = \frac{\alpha s_i}{\lambda + s_i},$$

where R_i and s_i are the i-th observation of reaction rate and substrate concentration, respectively, and α and λ here denote the "asymptote" and "half-life" parameters, respectively. Here we want to model how the inhibitor concentration "interacts" with the substrate concentration in the sense that one or both parameters of this model depend on the concentration.

1. Estimate a nonlinear regression model using the `nls` function that allows for each of the three levels of concentration to have a different value of the α and λ parameters, similar to how the example from lecture allowed these parameters to be different for cells that were treated or untreated by puromycin. So your model should estimate *six* parameters which we can denote using subscripts as α_0 and λ_0 for the control condition, α_1 and λ_1 for a inhibitor concentration of 50 μmoles, and α_2 and λ_2 for an inhibitor concentration of 100 μmoles. Essentially this is a case-wise model like that featured in class but with three cases rather than two. I *strongly* recommend that you use the `case_when` function from the **dplyr** package to manage the case-wise structure of this model.[4] Note that you should be able to "eyeball" (i.e., guess) the starting values from the plot of the raw data. Report the parameter estimates using `summary`. Finally, plot the model by producing a plot similar to that shown above but with curves depicting the estimated model.[5]

2. Using `summary` you can make inferences about the six model parameters, but they do not provide comparisons of the parameters. Comparisons can be made using the `lincon` function from the **trtools** package. Use `lincon` to estimate the *difference* between the corresponding Michaelis-Menten parameters between the control condition and the condition with an inhibition concentration of 50 μmoles (i.e., $\alpha_1 - \alpha_0$ and $\lambda_1 - \lambda_0$), and also between the control condition and the condition with an inhibition concentration of 100 μmoles (i.e., $\alpha_2 - \alpha_0$ and $\lambda_2 - \lambda_0$).

3. Now consider an alternative parameterization of the model where we write the model case-wise as

$$E(R_i) = \frac{\alpha_0 s_i}{\lambda_0 + s_i}$$

[4] It could, in principle, be done using indicator variables or even the `ifelse` function, but the code would be more complex.

[5] A plot like this is a good way to determine if you made significant mistake in estimating your model. If you do not produce curves that look consistent with the raw data you may have made a mistake.

if the i-th observation is from the control condition with an inhibitor concentration of zero,

$$E(R_i) = \frac{(\alpha_0 + \delta_1)s_i}{\lambda_0 + \tau_1 + s_i}$$

if the i-th observation is from the condition with an inhibitor concentration of 50 μmoles, and

$$E(R_i) = \frac{(\alpha_0 + \delta_2)s_i}{\lambda_0 + \tau_2 + s_i}$$

if the i-th observation is from the condition with an inhibitor concentration of 100 μmoles The parameters of this model are related to previous parameterization. We can see that $\alpha_1 = \alpha_0 + \delta_1$, $\alpha_2 = \alpha_1 + \delta_2$, $\lambda_1 = \lambda_0 + \tau_1$, and $\lambda_2 = \lambda_0 + \tau_2$. Estimate this model using the `nls` and give the parameter estimates using `summary`. Again, I would strongly recommend you use the `case_when` function here. Finally plot the model with the raw data like you did with the previous model. Your plot should look the same as that for the previous model.

4. The model you estimated in the previous problem provides estimates of $\delta_1 = \alpha_1 - \alpha_0$, $\delta_2 = \alpha_2 - \alpha_0$ as well as $\tau_1 = \lambda_1 - \lambda_0$ and $\tau_2 = \lambda_2 - \lambda_0$, so you can use just the output from `summary` to make comparisons with the control condition. And you should find that these estimates agree with what you obtained using `lincon` and the first model you estimated. Now use `lincon` with the model you estimated in the previous problem to estimate $\alpha_1 = \alpha_0 + \delta_1$, $\alpha_2 = \alpha_0 + \delta_2$, $\lambda_1 = \lambda_0 + \tau_1$, and $\lambda_2 = \lambda_0 + \tau_2$. These inferences should agree with what you obtained from `summary` in the first problem.

5. The models considered above treat the inhibitor concentration as a categorical variable (i.e., a factor with three levels). Another approach motivated by the biochemistry of the inhibitor is to let the λ parameter depend on the inhibitor concentration so that

$$E(R_i) = \frac{\alpha s_i}{\lambda_0(1 + h_i/\kappa) + s_i},$$

where h_i is the inhibitor concentration for the i-th observation. This model has three parameters: α, λ_0, and κ. Here the inhibitor does not affect the asymptote (α), but the half-life parameter is a linear function of the inhibitor concentration.[6] Here λ_0 is the value of the half-life parameter when the inhibitor concentration is zero, and κ is the inhibitor concentration necessary to double the half-life parameter from this value.[7] Estimate this nonlinear regression model and report the parameter estimates and their confidence intervals using `summary`.[8] Finally plot the model with the raw data like you did with the previous models.

Aerial Survey of Snow Geese

The data frame `snowgeese` from the **alr3** package is from an unpublished study of aerial survey methods for estimating the number of snow geese (*Anser caerulescens*) in their summer range areas west of Hudson Bay in Canada.[9] Counts were made separately by two observers from an aircraft that flew near flocks of geese. For comparison, an exact count of the number of geese in the flock was also obtained from a photograph. The first few observations can be seen below.

[6] We could alternatively write the model as

$$E(R_i) = \frac{\alpha s_i}{\lambda_i + s_i},$$

where $\lambda_i = \lambda_0(1 + h_i/\kappa)$ to show how the half-life parameter λ_i is now indexed by the observation since it depends on the inhibitor concentration, h_i.

[7] To see why we can interpret κ this way, note that if $h_i = \kappa$ then $\lambda_0(1 + h_i/\kappa) = 2\lambda_0$.

[8] To specify your starting values try the following strategy. The α parameter is the asymptote for the curve, regardless of the inhibitor concentration, so you can guess this from the plot of the raw data. And because λ_0 is the value of the half-life parameter when the concentration is zero, you can guess this from the plot or use an estimate from one of the previous models. For κ try guessing the value of the half-life parameter when the inhibitor concentration is at, say, $h_i = 50$, or use an estimate from one of the previous models. Call this guess/estimate $\hat{\lambda}_1$. We have that $\lambda_1 = \lambda_0(1 + 50/\kappa)$. Replace λ_0 and λ_1 in that expression with guesses/estimates as described above and then solve for κ to get a starting value for that parameter.

[9] Source: Cook, R. D. & Jacobsen, J. O. (1978). *Analysis of the 1977 West Hudson Bay snow goose surveys.* Unpublished report, Canadian Wildlife Service.

```
library(alr3)
head(snowgeese)
```

```
  photo obs1 obs2
1    56   50   40
2    38   25   30
3    25   30   40
4    48   35   45
5    38   25   30
6    22   20   20
```

Note: To install the **alr3** package use `install.packages("alr3", repos = "http://R-Forge.R-project.org")` since it is no longer available on the Comprehensive R Archive Network (CRAN) repository, which is the default for `install.packages`.[10] The variable `photo` is the exact count of the number of geese in a flock while `obs1` and `obs2` are the visual counts from the two observers. To plot and model these data they need to be reshaped into "long form" so that we have one observer count in each row. This can be done as follows.

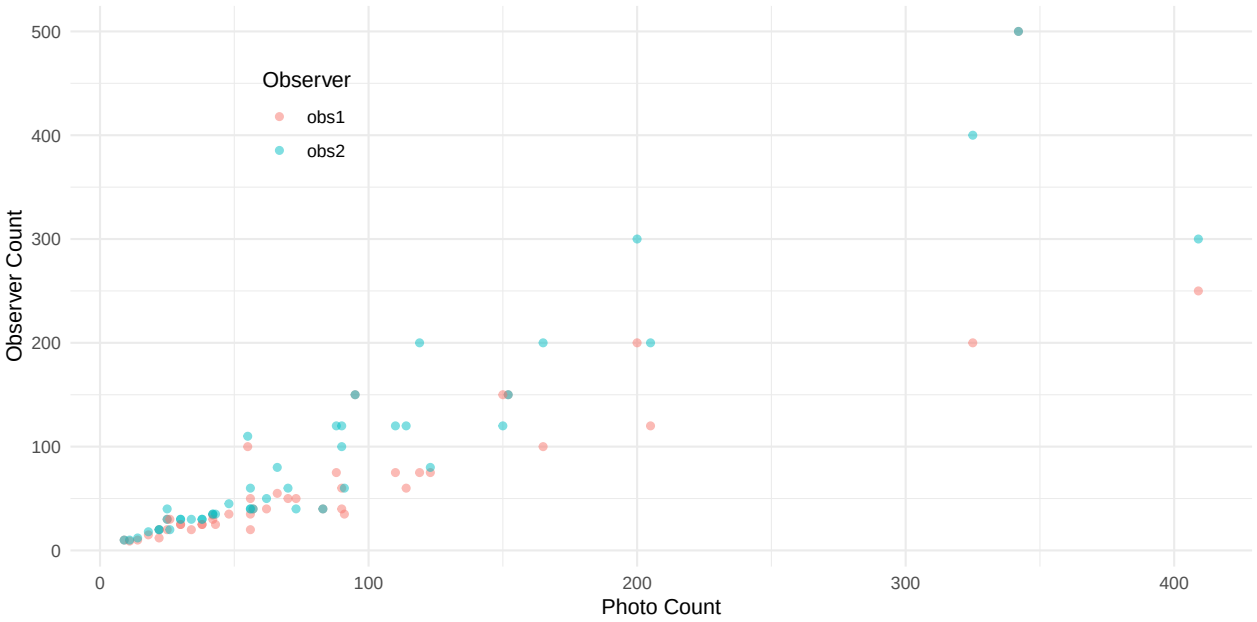
```
library(dplyr)
library(tidyr)
goosecount <- snowgeese %>% mutate(flock = 1:n()) %>%
  pivot_longer(c(obs1, obs2), names_to = "observer", values_to = "count")
head(goosecount)
```

```
# A tibble: 6 x 4
  photo flock observer count
  <int> <int> <chr>    <int>
1    56     1 obs1        50
2    56     1 obs2        40
3    38     2 obs1        25
4    38     2 obs2        30
5    25     3 obs1        30
6    25     3 obs2        40
```

Here is a plot of these data.

```
library(ggplot2)
p <- ggplot(goosecount, aes(x = photo, y = count, color = observer)) +
  theme_minimal() + geom_point(alpha = 0.5) +
  labs(x = "Photo Count", y = "Observer Count", color = "Observer") +
  theme(legend.position = c(0.2, 0.8))
plot(p)
```

[10] The **alr3** package was replaced with a newer package, **alr4**, but unfortunately the `snowgeese` data are not included with this newer package.



The goal here is to use a regression model to investigate the accuracy (or lack thereof) of using human observers in aerial surveys to estimate the size of snow geese flocks.

1. Here you will use a model like that used for the `biomass` data featured in lecture. The model can be written as

$$E(Y_i) = \begin{cases} \beta_1 x_i, & \text{if the } i\text{-th observation is from observer 1,} \\ \beta_2 x_i, & \text{if the } i\text{-th observation is from observer 2,} \end{cases}$$

   where Y_i is the observer count and x_i is the photo count. See the lecture notes for how to specify this model. Estimate this model and give the parameter estimates using `summary`. Also plot the model with the raw data.

2. Using the model you estimated estimated above, determine if there is a statistically significant difference between β_1 and β_2 using the `lincon` function. Also conduct a test of the null hypothesis $\beta_1 = 1$, and then again for the null hypothesis $\beta_2 = 1$. The reason why the null hypotheses $\beta_1 = 1$ and $\beta_2 = 1$ are interesting here is that if we can show that the slope of the line is greater/less than one then we can show that the observers tend to overestimate/underestimate the the size of a flock. You can test these hypotheses one of two ways: use the `lincon` function (hint: use the `b` argument for the `lincon` function and note that, for example, $\beta_1 = 1$ can also be written as $\beta_1 - 1 = 0$), or use a confidence interval to conduct the test. For each test be sure to state your conclusion for each test (i.e., reject or do not reject the null hypothesis). Use a significance level of $\alpha = 0.05$.

3. As with the `biomass` data, these data show considerable heteroscedasticity. As shown in lecture, use an iteratively weighted least squares approach assuming that $\text{Var}(Y_i) \propto E(Y_i)^p$ for some value of p to determine your weights. Try different values of p and decide what you think is a reasonable value of p to address the heteroscedasticity based on a plot of the studentized residuals against the predicted values. Provide your residual plot for that value of p, and then repeat what you did in the previous problem using estimates obtained using the iteratively weighted least squares with your choice of p.

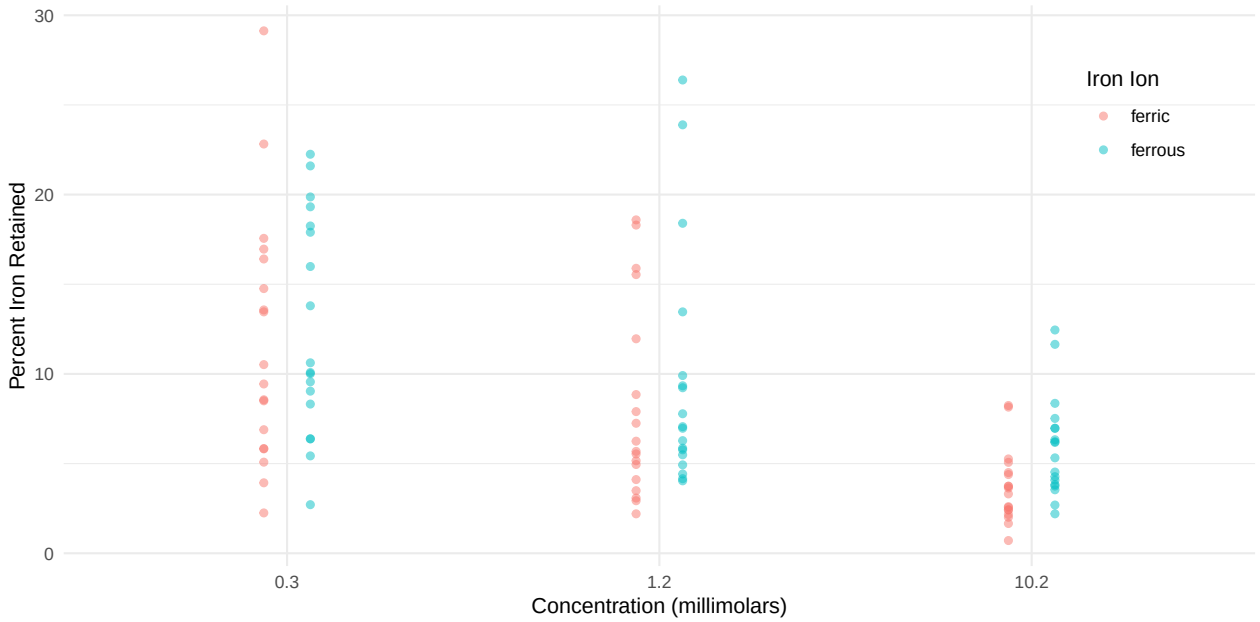
Iron Retention in Rats

The data frame `ironretention` from the **trtools** package is from a randomized experiment on the retention of two iron ions, ferrous (Fe2+) and ferric (Fe3+), administered to mice at three different concentrations (0.3, 1.2, and 10.2 millimolars).[11] The response variable is the percent of iron retained. The data are shown in the

[11] Rice, J. A. (1998). *Mathematical statistics and data analysis*. Pacific Grove, CA: Wadsworth & Brook/Cole.

plot below.[12]

```
library(trtools)
library(ggplot2)
p <- ggplot(ironretention, aes(x = factor(concentration), y = retention, color = ion)) +
  geom_point(alpha = 0.5, position = position_dodge(width = 0.25)) +
  labs(x = "Concentration (millimolars)",
    y = "Percent Iron Retained", color = "Iron Ion") +
  theme_minimal() + theme(legend.pos = c(0.9, 0.8))
plot(p)
```



Some descriptive statistics for these data can be obtained as follows.

```
library(dplyr)
ironretention %>% group_by(concentration, ion) %>%
  summarize(mean = mean(retention), sd = sd(retention), obs = n())
```

```
# A tibble: 6 x 5
# Groups:   concentration [3]
  concentration ion      mean    sd   obs
          <dbl> <chr>   <dbl> <dbl> <int>
1           0.3 ferric  11.8   7.03    18
2           0.3 ferrous 12.6   6.08    18
3           1.2 ferric   8.20  5.45    18
4           1.2 ferrous  9.63  6.69    18
5          10.2 ferric   3.70  2.03    18
6          10.2 ferrous  5.94  2.81    18
```

Notice that the variability of the observations of percent iron retention tends to be higher when the mean retention is higher.

In what follows we will treat concentration as a categorical variable. We can see that it is a numeric variable in the data frame using `str` (structure).

[12]Note the use of the `position` argument in `geom_point`. Without that argument the points denoting the ion would be in the same position over the abscissa. But by specifying `position = position_dodge(width = 0.25)` we "dodge" the points by moving them slightly sideways (with the amount determined by the `width` argument to `position_dodge`).

```
str(ironretention)
```

```
'data.frame':   108 obs. of  3 variables:
 $ retention    : num  2.71 6.38 9.56 10.62 17.9 ...
 $ concentration: num  0.3 0.3 0.3 0.3 0.3 0.3 1.2 1.2 1.2 1.2 ...
 $ ion          : chr  "ferrous" "ferrous" "ferrous" "ferrous" ...
```

We can use the following code to create a variable `concf` which is a factor created from the variable `concentration`.

```
ironretention$concf <- factor(ironretention$concentration)
```

That `concf` is a factor can be confirmed by using `str(ironretention)`.[13] In what follows you will start with the following linear model estimated by (ordinary/unweighted) least squares.

```
m <- lm(retention ~ concf + ion + concf:ion, data = ironretention)
summary(m)$coefficients
```

```
                    Estimate Std. Error t value  Pr(>|t|)
(Intercept)          11.7500      1.265  9.2882 3.061e-15
concf1.2             -3.5461      1.789 -1.9821 5.015e-02
concf10.2            -8.0511      1.789 -4.5002 1.805e-05
ionferrous            0.8894      1.789  0.4972 6.201e-01
concf1.2:ionferrous   0.5389      2.530  0.2130 8.318e-01
concf10.2:ionferrous  1.3483      2.530  0.5329 5.952e-01
```

Note that this model can be written as

$$E(Y_i) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} + \beta_5 x_{i5},$$

where Y_i is the i-th observation of iron retention,

$$x_{i1} = \begin{cases} 1, & \text{if the } i\text{-th observation of concentration is 1.2,} \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{i2} = \begin{cases} 1, & \text{if the } i\text{-th observation of concentration is 10.2,} \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{i3} = \begin{cases} 1, & \text{if the } i\text{-th observation is iron ion is ferrous,} \\ 0, & \text{otherwise,} \end{cases}$$

$x_{i4} = x_{i1}x_{i3}$ and $x_{i5} = x_{i2}x_{i3}$. So the model can be written case-wise as

$$E(Y_i) = \begin{cases} \beta_0, & \text{if concentration is 0.3 and the ion is ferric,} \\ \beta_0 + \beta_1, & \text{if the concentration is 1.2 and the ion is ferric,} \\ \beta_0 + \beta_2, & \text{if the concentration is 10.2 and the ion is ferric,} \\ \beta_0 + \beta_3, & \text{if concentration is 0.3 and the ion is ferrous,} \\ \beta_0 + \beta_1 + \beta_3 + \beta_4, & \text{if the concentration is 1.2 and the ion is ferrous,} \\ \beta_0 + \beta_2 + \beta_3 + \beta_5, & \text{if the concentration is 10.2 and the ion is ferrous.} \end{cases}$$

[13] Notice that the variable `ion` is not a factor but is instead a *character* (`chr`) variable (i.e., a word or phrase, what is sometimes called a "string"). A character variable is not treated quite the same as a factor in R, and sometimes this is an issue, but most regression modeling functions will automatically convert a character variable to a factor when they are used in the model formula so the distinction is not generally a concern unless we try to use functions to manipulate the variable that are specifically designed for factors (e.g., `level`, `reorder`, and the functions in the **forcats** package).

This model specifies an interaction between concentration and ion so that the difference in the expected iron retention between the two ions can be different at each of the three concentrations.[14]

1. Based on the model defined above, use either the `contrast` function from the **trtools** package or functions from the **emmeans** package to estimate (a) the expected iron retention at each of the six combinations of concentration and ion and (b) the *difference* in the expected iron retention between the two ions at each of the three concentrations (i.e., the difference between ferrous and ferric at concentrations of 0.3, 1.2, and 10.2 millimolars).

2. Use a residual plot to check for heteroscedasticity. Comment briefly on if you believe there is evidence of heteroscedasticity and why. Be sure to include your plot.

3. In models where the response variable is a percent so that $0 \leq Y_i \leq 100$ as it is here, it has been suggested that an appropriate variance structure might be

$$\text{Var}(Y_i) \propto [E(Y_i)/100]^p[1 - E(Y_i)/100]^p,$$

where $p \geq 1$.[15] Use an *iteratively weighted least squares* algorithm similar to that we used in class, but specify your weights so that they are the *reciprocal* of $[E(Y_i)/100]^p[1 - E(Y_i)/100]^p$ where each $E(Y_i)$ is estimated using $\hat{y}_i$ so that

$$w_i = \frac{1}{(\hat{y}_i/100)^p(1 - \hat{y}_i/100)^p}.$$

Be careful that you compute the weights correctly.[16] Try this for values of p of 1, 2, 3, 4, and 5.[17] For each of p produce a plot of the studentized residuals against the predicted values. Comment briefly on

[14] Models with interactions can often be confusing and intimidating. The parameterization that is used by default is selected largely for computational convenience, and to some degree by tradition. Perhaps a simpler way to look at this model is with an alternative parameterization.

```
m <- lm(retention ~ -1 + concf:ion, data = ironretention)
summary(m)$coefficients
```

```
                    Estimate Std. Error t value  Pr(>|t|)
concf0.3:ionferric    11.750      1.265   9.288 3.061e-15
concf1.2:ionferric     8.204      1.265   6.485 3.220e-09
concf10.2:ionferric    3.699      1.265   2.924 4.259e-03
concf0.3:ionferrous   12.639      1.265   9.991 8.511e-17
concf1.2:ionferrous    9.632      1.265   7.614 1.386e-11
concf10.2:ionferrous   5.937      1.265   4.693 8.400e-06
```

Here there is effectively an indicator variable (formed as the product of two indicator variables) for each combination of concentration and ion, so the model can be written case-wise as

$$E(Y_i) = \begin{cases} \beta_1, & \text{if concentration is 0.3 and the ion is ferric,} \\ \beta_2, & \text{if the concentration is 1.2 and the ion is ferric,} \\ \beta_3, & \text{if the concentration is 10.2 and the ion is ferric,} \\ \beta_4, & \text{if concentration is 0.3 and the ion is ferrous,} \\ \beta_5, & \text{if the concentration is 1.2 and the ion is ferrous,} \\ \beta_6, & \text{if the concentration is 10.2 and the ion is ferrous.} \end{cases}$$

Note that this is the *same* model, just written (i.e., parameterized) differently. But this parameterization shows that a model with two categorical variables and an interaction between them effectively treats each distinct combination of categories as a category, so we can view this model as one with *one* factor formed by the six combinations of the levels of two factors. Fortunately when we are using functions like `contrast` or `emmeans` for inferences we do not need to be too concerned about the specific parameterization.

[15] If Y_i is a *proportion* so that $0 \leq Y_i \leq 1$, the division by 100 is omitted so that $\text{Var}(Y_i) \propto E(Y_i)^p[1 - E(Y_i)]^p$.

[16] Because these weights are a more complicated function of the predicted values you might find it useful to compute the weights in a couple of steps where you first save the predicted values to the data frame and then you compute the weights using the `with` function so that you do not need to write `ironretention$yhat` to reference the predicted values. Here is how that would look like for the example in class on 2/24.

```
daphniastrat$yhat <- predict(m)              # save predicted values
daphniastrat$w <- with(daphniastrat, 1/yhat) # compute weights
```

Note that you will compute the weights differently here and not use `1/yhat`.

[17] Although we are only considering integer values of p, it could be any real number (e.g., $p = 1.5$).

what you might think might be an appropriate value (or values) of p to produce appropriate weights to address any heteroscedasticity. Then using what you believe is the best value of p to address any heteroscedasticity, estimate the model using weighted least squares based on that value of p and repeat what you did in the first problem. Compare the estimates and standard errors of these results with those you did before, and comment briefly on if or how they changed.

4. Since the explanatory variables are categorical and there are multiple observations per treatment condition, another approach to dealing with the heteroscedasticity is to let the variance vary over the treatment conditions so that

$$\text{Var}(Y_i) = \begin{cases} \sigma_1^2, & \text{if concentration is 0.3 and the ion is ferric,} \\ \sigma_2^2, & \text{if the concentration is 1.2 and the ion is ferric,} \\ \sigma_3^2, & \text{if the concentration is 10.2 and the ion is ferric,} \\ \sigma_4^2, & \text{if concentration is 0.3 and the ion is ferrous,} \\ \sigma_5^2, & \text{if the concentration is 1.2 and the ion is ferrous,} \\ \sigma_6^2, & \text{if the concentration is 10.2 and the ion is ferrous.} \end{cases}$$

If these variances were known then the weights for the observations in each treatment condition would be the reciprocals of these variances (e.g., the weight for the observations in the first treatment condition would be $w_i = 1/\sigma_1^2$). These variances are not known but can be estimated from the data. Use the sample variances from the six treatment conditions (i.e., $s_1^2, s_2^2, \ldots, s_6^2$) to create weights and estimate the model using weighted least squares with these weights. Repeat what you did in the first problem and comment briefly on if and how your inferences changed.

Modeling the Biopotency of Methionine in Turkeys

Note: This problem is extra credit for students in Stat 436, but required for students in Stat 516.

This problem concerns data from a study of the effects of varying doses and sources of methionine (an essential amino acid) on the growth of young turkeys.[18] The data are not, to my knowledge, included in an existing R package, but they can be entered into a data frame called `turkeys` as follows.

```
turkeys <- data.frame(
  weight = c(674, 764, 795, 796, 826, 782, 834, 836, 830),
  pens = c(10, 5, 2, 2, 5, 5, 2, 2, 5),
  dosea = c(c(0, 0.12, 0.22, 0.32, 0.44), rep(0, 4)),
  doseb = c(rep(0, 5), c(0.12, 0.22, 0.32, 0.44))
)
turkeys
```

```
  weight pens dosea doseb
1    674   10  0.00  0.00
2    764    5  0.12  0.00
3    795    2  0.22  0.00
4    796    2  0.32  0.00
5    826    5  0.44  0.00
6    782    5  0.00  0.12
7    834    2  0.00  0.22
8    836    2  0.00  0.32
9    830    5  0.00  0.44
```

Each observation of `weight` is the average weight (in grams) of a given number of `pens`, where each pen contained 15 turkeys. The turkeys in a given pen were given a dose of one of two sources of methionine, with the doses recorded as the percent of the total diet. Note that 10 pens were not given any additional

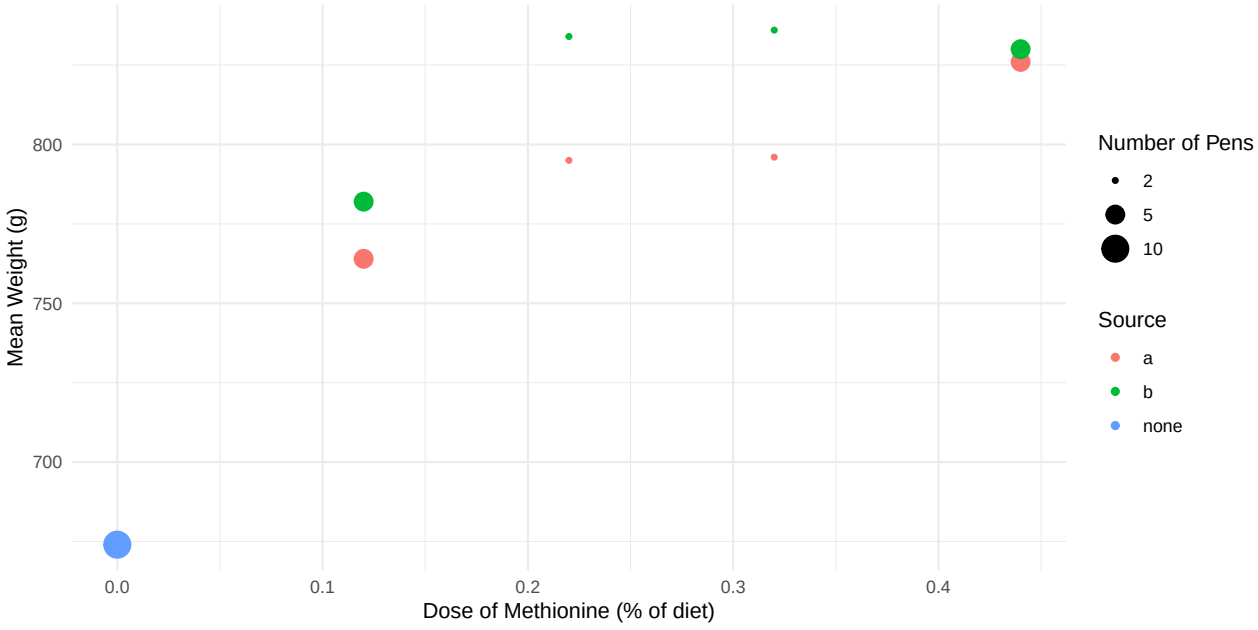
[18] Source: Noll, S. L., Waibel, P. E., Cook, R. D., & Witmer, J. A. (1984). Biopotency of methionine sources for young turkeys. *Poultry Science*, *63*, 2458–2470.

methionine. The researchers varied the number of pens for a given dose as part of an optimal experimental design. To plot these data it is useful to create a couple of new variables.

```
library(dplyr)
turkeys <- turkeys %>%
  mutate(dose = dosea + doseb) %>%
  mutate(source = case_when(
    dose == 0 ~ "none",
    dosea > 0 ~ "a",
    doseb > 0 ~ "b")
  )
turkeys
```

```
  weight pens dosea doseb dose source
1    674   10  0.00  0.00 0.00   none
2    764    5  0.12  0.00 0.12      a
3    795    2  0.22  0.00 0.22      a
4    796    2  0.32  0.00 0.32      a
5    826    5  0.44  0.00 0.44      a
6    782    5  0.00  0.12 0.12      b
7    834    2  0.00  0.22 0.22      b
8    836    2  0.00  0.32 0.32      b
9    830    5  0.00  0.44 0.44      b
```

```
library(ggplot2)
p <- ggplot(turkeys, aes(x = dose, y = weight, color = source)) +
  theme_minimal() + geom_point(aes(size = pens)) +
  scale_size(breaks = c(2, 5, 10)) +
  labs(x = "Dose of Methionine (% of diet)", y = "Mean Weight (g)",
    color = "Source", size = "Number of Pens")
plot(p)
```



Here you will consider modeling expected mean weight as a function of dose and source of methionine, while

also accounting the number of pens used for mean weight. The basic model you will be using has the form

$$E(W) = \alpha + (\gamma - \alpha)2^{-d/\lambda},$$

where W is the (mean) weight and d is dose of methionine. Here α is the asymptote representing the expected weight approached as d is increases, γ is the expected weight when no methionine is given, and λ is a "half-life" parameter interpreted as the dose of methionine that results in an expected weight half way between γ and α so that if $d = \lambda$ then $E(W) = (\alpha + \gamma)/2$. But here you will be extending this model to account for modeling different sources of methionine. The assumption will be that if the biopotency of methionine varies depending on the source, then this will be reflected in differences in the value of λ.

Because the number of pens used to compute the average weight varied by dose, the variance of the average weight will be inversely proportional to the number of pens — i.e., $\text{Var}(Y_i) \propto 1/n_i$ where n_i is the number of pens averaged for the i-th observation (similar to the example from lecture on 2/17). So in what follows use the number of pens to specify weights to estimate the model parameters using *weighted* least squares.

1. Consider the regression model

$$E(W_i) = \begin{cases} \gamma, & \text{if no methionine was given,} \\ \alpha + (\gamma - \alpha)2^{-d_i/\lambda_a}, & \text{if methionine was given from source } a, \\ \alpha + (\gamma - \alpha)2^{-d_i/\lambda_b}, & \text{if methionine was given from source } b. \end{cases}$$

   Note that if the dose is zero then the expected weight is γ, since $\alpha + (\gamma - \alpha)2^{-d/\lambda} = \gamma$ if $d = 0$. Estimate this model using the `nls` function and report the parameter estimates by giving the output from `summary`.[19] I would recommend using the `case_when` function from the **dplyr** package to specify this model in `nls`. Plot the model with the raw data. Finally, use the `lincon` function to determine if there is a statistically significant difference between λ_a and λ_b, meaning that the biopotency of methionine does depend on the source (assume a significance level of $\alpha = 0.05$).

2. Another way to write the model that avoids having to explicitly write the model case-wise is

$$E(W_i) = \alpha + (\gamma - \alpha)2^{-(a_i/\lambda_a + b_i/\lambda_b)},$$

   where a_i and b_i are the doses of methionine from sources a and b, respectively (i.e., these are the variables `dosea` and `doseb` in the data frame). Estimate this model using the `nls` function and report the parameter estimates using the output from `summary`. You should get the same results as you obtained in the previous problem.

[19] The models you are estimated here may be somewhat sensitive to specifying good starting values. If you have difficulties I would recommend that you start by *not* estimating λ_a and λ_b but instead insert values in the model based on guessing their values from the plot (similar to how I guessed the value of the h parameter for the `ToothGrowth` data from lecture). Then use the estimates of α and γ you get as starting values for those parameters, and your guesses of λ_a and λ_b as starting values for those parameters, and proceed to estimate the model with all four unknown parameters.