Covariates and Factors Homework Answers

Crispin Jordan

12/11/2021

Question 1

Let's import the data:

```
liz <- read.table("lizard.csv", header = TRUE, sep = ',')
```

Before we continue, let's look at the data. First, we'll examine the types of variables present:

```
str(liz)
```

```
## 'data.frame':    11 obs. of  2 variables:
##  $ bite          : num  4.3 4.6 4.3 5 5 5.2 5.3 5.4 4.8 4.7 ...
##  $ territory.area: num  14 16 18.8 19.8 22.5 28 29.8 34.8 30.8 27.4 ...
```

We have two numeric (`num`) variables: `bite` and `territory.area`.

The dataset is not big, and looks like this:

```
liz
```

```
##    bite territory.area
## 1   4.3           14.0
## 2   4.6           16.0
## 3   4.3           18.8
## 4   5.0           19.8
## 5   5.0           22.5
## 6   5.2           28.0
## 7   5.3           29.8
## 8   5.4           34.8
## 9   4.8           30.8
## 10  4.7           27.4
## 11  4.6           32.4
```

Only 11 data points. Be warned that small datasets can be more challenging to analyze than large datasets (!) because it can be more difficult to assess the assumptions.

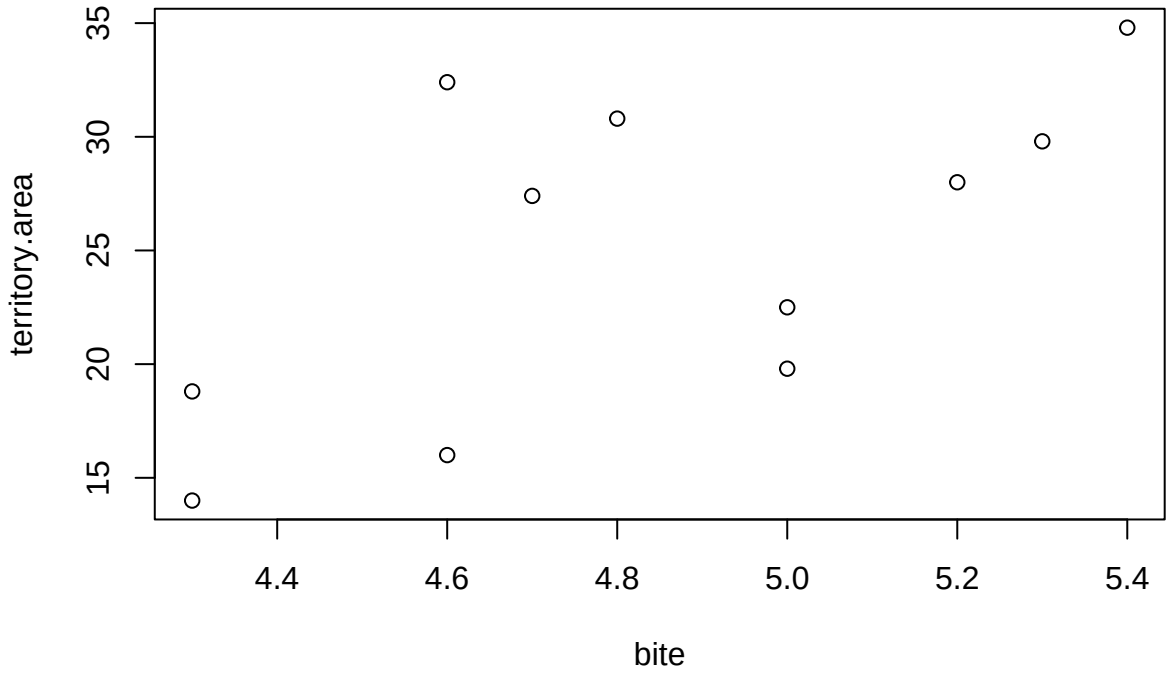
Let's look again at the question asked us to do: "Lappin and Husak (2005) tested whether the force of an individual's bite **predicted** its territory size (e.g., do lizards with stronger bites have larger territories?)." The word 'predicted' is important here for two reasons.

- First, the word **predict** implies *causation*, which is best addressed by regression. This is because regression works in a way that assumes that the dependent variable *depends* on the independent variable. If, instead, the authors had wanted to test for whether a change in one variable is *associated* with a change in the other, they may have best analyzed their data with **correlation** (which we're not considering in this problem set.) (Note the distinction here: correlation only considers whether variables are associated, but regression implies that one variable causally affects another variable.)

- Second, the word **predicted** indicates which variable should be the *dependent* vs. *independent* variable. If we're interested in whether `bite` strength predicts `territory.area`, then this implies that `territory.area` is the *dependent* variable (and will go to the left of the tilda (`~`) in our model).

Following this logic, let's start by plotting the data:

```
plot(territory.area ~ bite, data = liz)
```



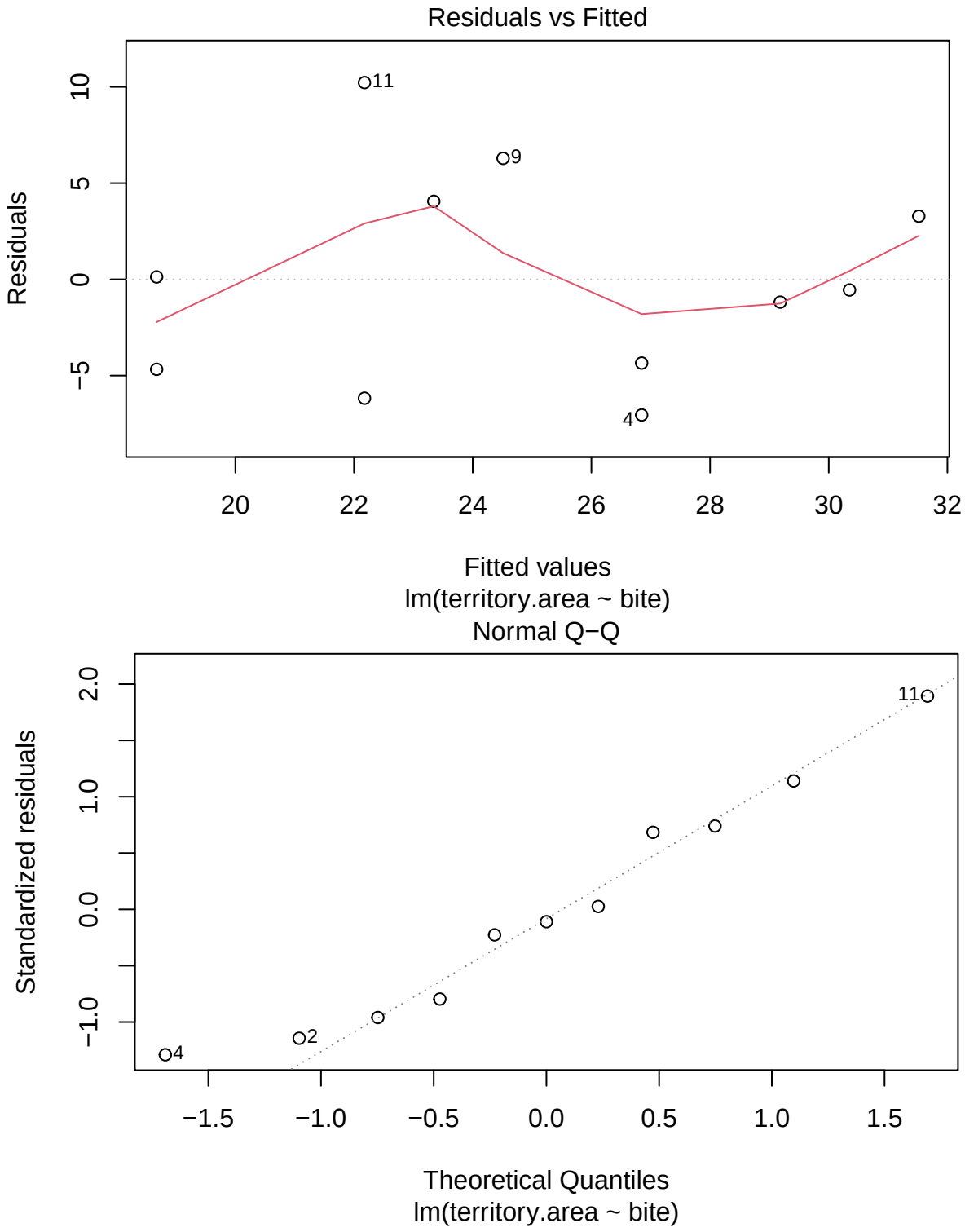
What do we see? Well, to me, it looks like there may be a weak trend for `territory.area` to increase with `bite`. There do not appear (yet) to be any major outliers (unusual data points) and it seems reasonable (so far) to assume the relationship is a straight line.

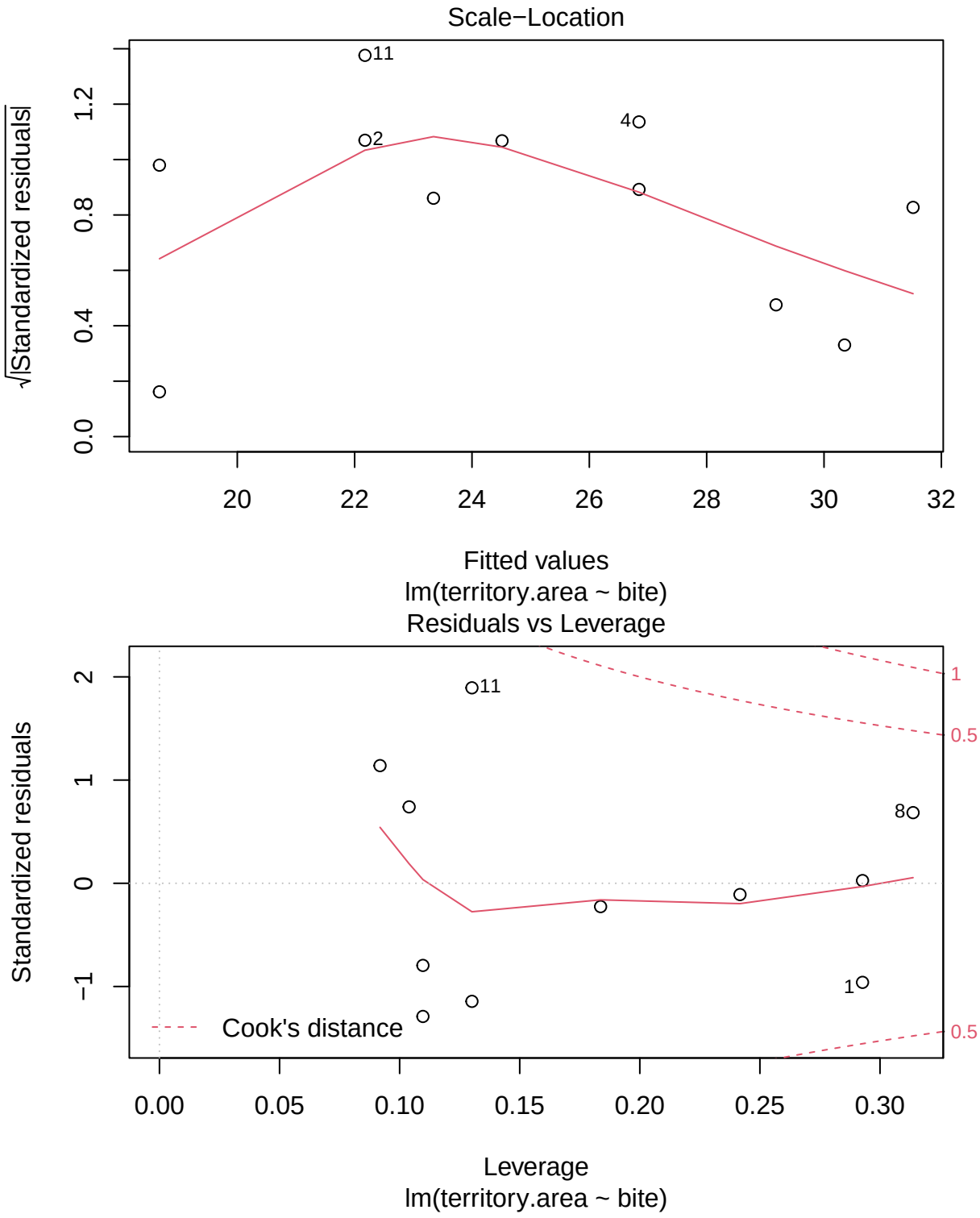
With this in mind, let's run a model of these data:

```
liz.lm <- lm(territory.area ~ bite, data = liz)
```

As always, let's start by checking the assumptions:

```
plot(liz.lm)
```


Residuals vs Fitted
Residuals
10
5
0
−5
11
9
4
20
22
24
26
28
30
32
Fitted values
lm(territory.area ~ bite)
Normal Q−Q
Standardized residuals
2.0
1.0
0.0
−1.0
11
2
4
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
Theoretical Quantiles
lm(territory.area ~ bite)



We'll assume the authors who collected these data did so in a way that meets the assumptions of random selection and independence (having looked at the original paper (https://doi.org/10.1086/432564), the authors' do well to meet the assumptions.). Let's examine the residuals to test the remaining assumptions.

The second plot suggests the residuals are normally distributed. However, the first and third plots raise worries about equal variance. In response, I tried transforming the data in various ways (various transformation of the x-axis, y-axis or both), and also tried fitting a polynomial function (i.e., a curve; I not shown you how to

do this yet), but none of these efforts improved the residuals much. (*It turns out that the original authors used correlation, not regression, to analyze these data. I disagree with their approach due to their goal to 'predict', but this does explain why they did not appear to encounter issues with residuals*).

What do we do? Well, it is worth noting that residuals as poor as these can easily arise even when the data **do** meet the assumptions, simply due to small sample size. The code at the end of this question simulates data with similar qualities as the present data: the simulated data do meet the assumptions of the test, but they often appear not to due to small sample size. This highlights the problems of working with small datasets: it can be much harder to assess the assumptions. Does this mean that we can stop worrying about whether the data meet the assumptions? No, not really - the residual plots are worrying with respect to equal variance. But, we're going to continue analyzing these data purely for the learning experience and remember that if we were the authors of these data (and analyzed them with regression, not correlation) we'd warn our readers of concerns over possibly violated assumptions and how this may increase the possibility of a Type 1 error.

Let's examine the p-value. We can do this two ways. First, we'll use `anova()` to obtain only a p-value of the slope:

```
anova(liz.lm)
```

```
## Analysis of Variance Table
##
## Response: territory.area
##           Df Sum Sq Mean Sq F value  Pr(>F)
## bite       1 194.37 194.374  5.8012 0.03934 *
## Residuals  9 301.55  33.506
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value equals 0.03934, providing 'moderate' or 'suggestive' evidence for a relationship between `bite` and `territory.area`.

We can also obtain the -value for the slope from the summary output of our model:

```
summary(liz.lm)
```

```
##
## Call:
## lm(formula = territory.area ~ bite, data = liz)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.0472 -4.5101 -0.5504  3.6689 10.2237
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -31.539     23.513  -1.341   0.2127
## bite          11.677      4.848   2.409   0.0393 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.788 on 9 degrees of freedom
## Multiple R-squared:  0.3919, Adjusted R-squared:  0.3244
## F-statistic: 5.801 on 1 and 9 DF,  p-value: 0.03934
```

Notice that we obtain the same p-value for `bite`. But we also obtain estimates of the slope (11.677; SE = 4.848) and y-intercept (-31.539; SE = 23.513).

In a results section, we could report the moderate evidence for a relationship between `bite` and `territory.area` (regression; F (1,9) = 5.801, p = 0.039), as well as the equation of the line (with SE's) provided in the paragraph, above. If course, we'd also warn our readers about possible increased probability of Type 1 error due to possibly violated assumptions.

Below, please find some code to simulate data that do meet the assumptions of a regression and that closely match the qualities of the data we just analyzed. If you run this code, yourself, several times you will likely come across worrisome residual plots fairly easily, despite the fact that the generated data do meet the assumptions of regression.

A short script to simulate data and residuals for lizards:

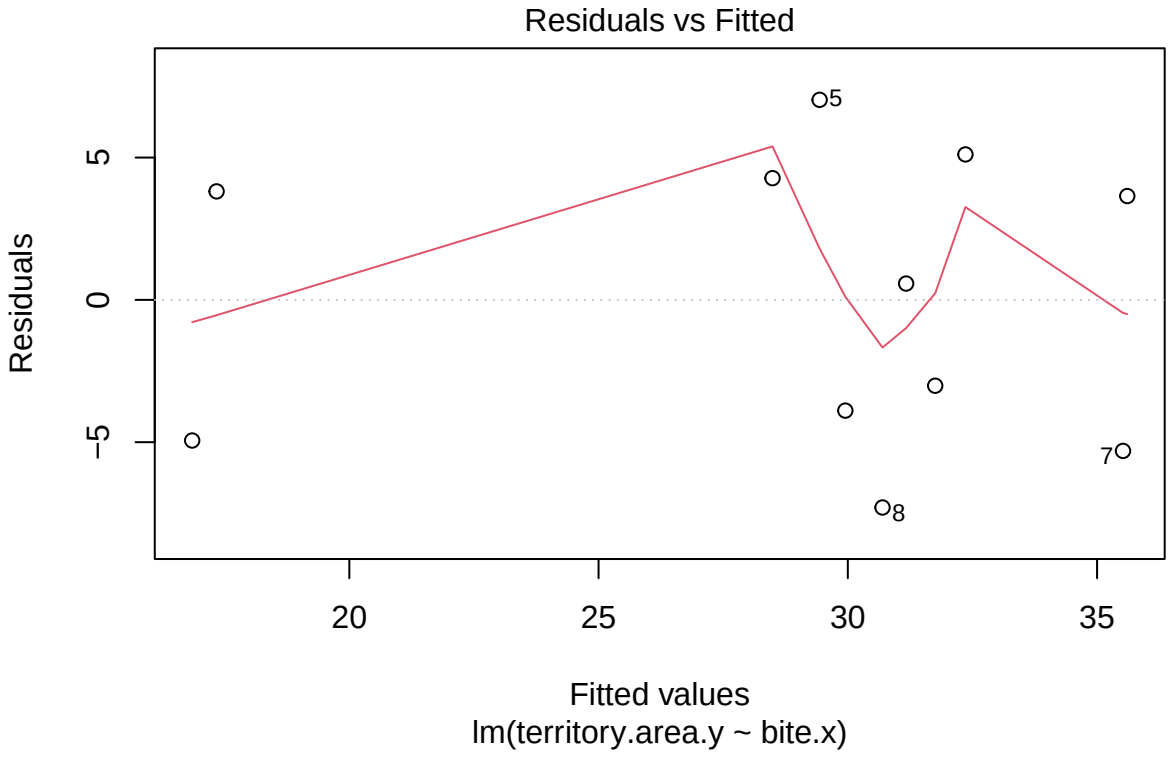
```
#Plot `bite` data to asses its' distribution:
hist(liz$bite)
```

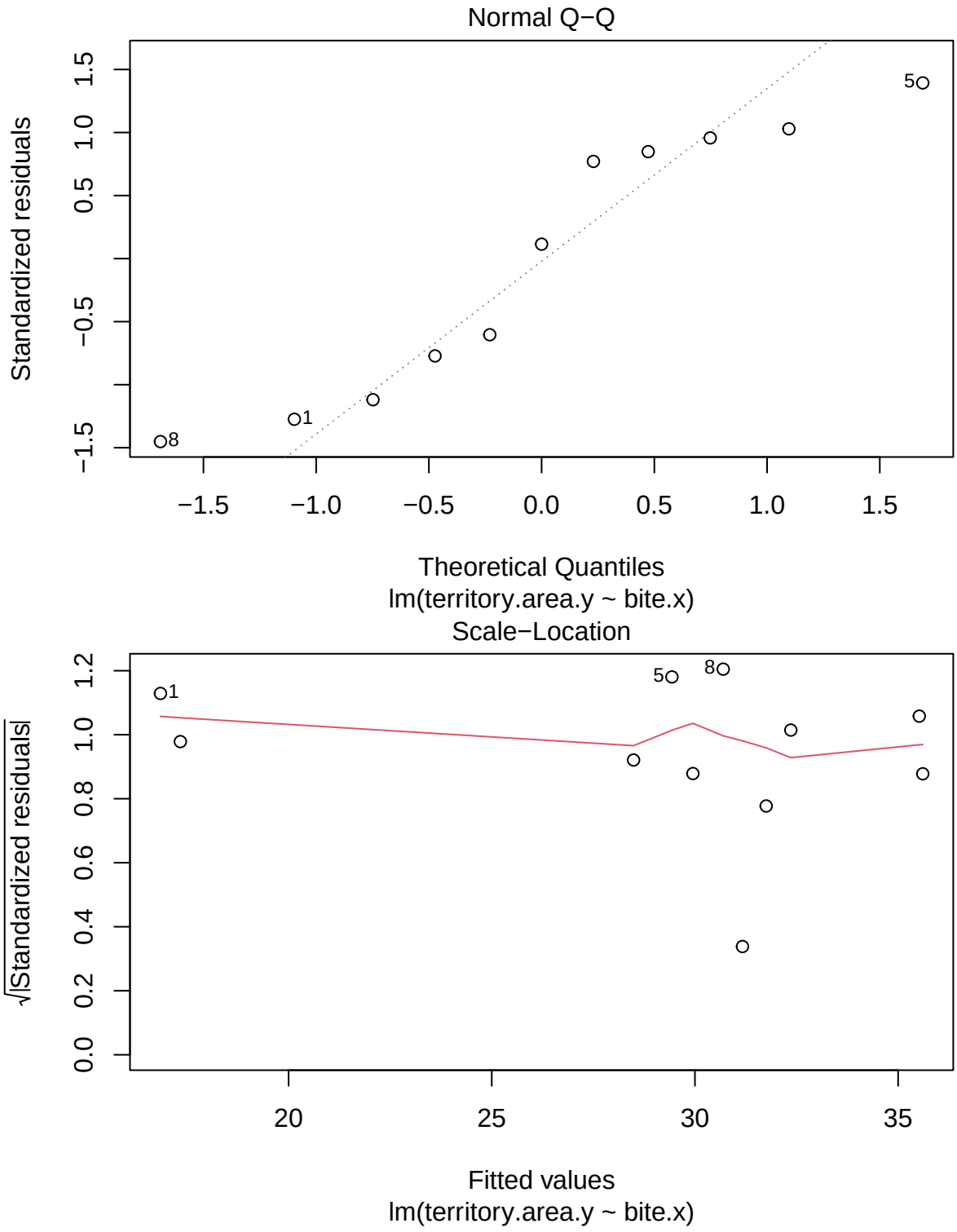

```
#`bite` appears to be uniformly distributed: select random values
#from a uniform distribution with the same min and max as observed:
bite.x <- runif(11, min = min(liz$bite), max = max(liz$bite))
#NOTE:  we get similar results if we assume 'bite' is normally
#disctributed:  bite.x <- rnorm(11,mean(liz$bite), sd(liz$bite))

#Determine intercept, slope, and standard deviation of residuals
#from this output:
summary(liz.lm)
```

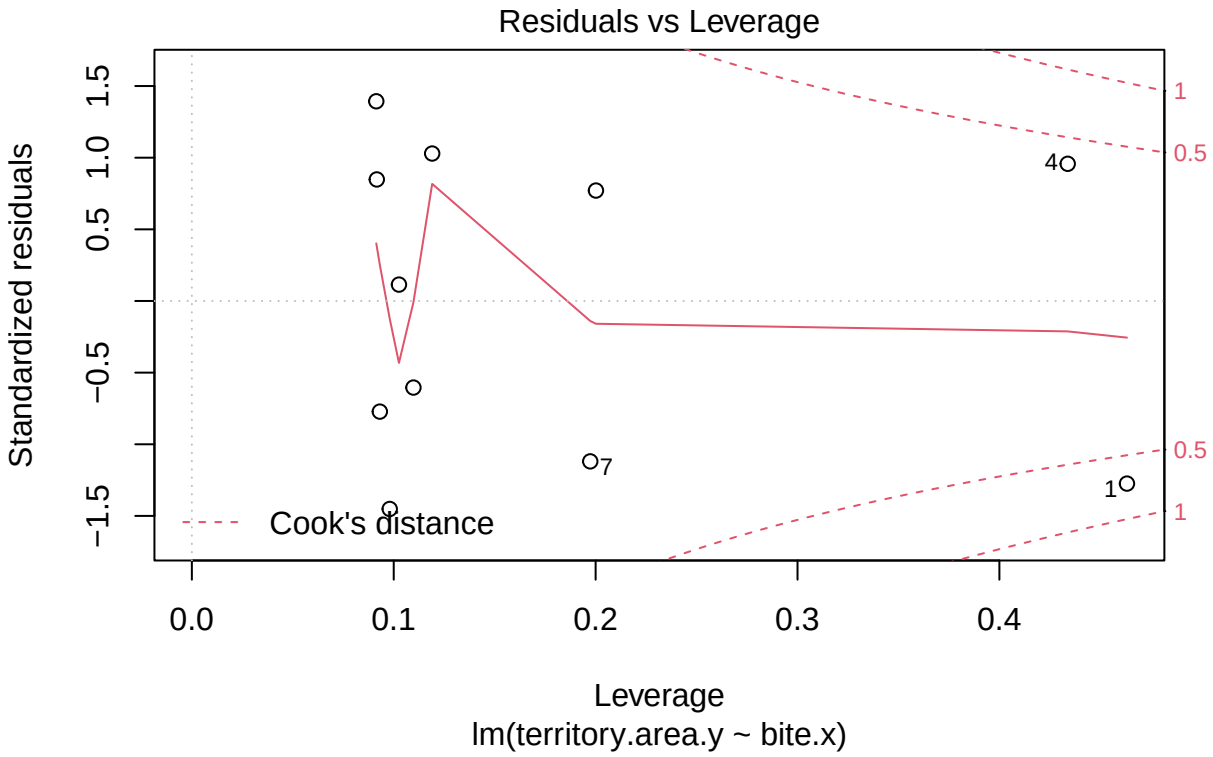
```
##
## Call:
## lm(formula = territory.area ~ bite, data = liz)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -7.0472 -4.5101 -0.5504  3.6689 10.2237
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -31.539     23.513  -1.341   0.2127
## bite          11.677      4.848   2.409   0.0393 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.788 on 9 degrees of freedom
## Multiple R-squared:  0.3919, Adjusted R-squared:  0.3244
## F-statistic: 5.801 on 1 and 9 DF,  p-value: 0.03934
```

```r
#Simulate residual variance (11 observations), mean of 0 and sd
#equal to sd of residuals in model, `liz.lm` (see above):
y.res <- rnorm(11,0,5.788)
#Use `bite.x` values and observed equation of line in original
#data to calculate y-values (`territory.area`) without residual
#variation:
territory.area.y <- bite.x*11.677 -31.539
#Add residual variation to y-values (`territory.area`):
territory.area.y <- territory.area.y + y.res
#Model the data:
bite.sim.lm <- lm(territory.area.y ~ bite.x)
#Plot residuals
plot(bite.sim.lm)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(territory.area.y ~ bite.x)
Scale−Location
√|Standardized residuals|
Fitted values
lm(territory.area.y ~ bite.x)



Question 2

Let's start by importing the data and examining the data:

```
brains <- read.table("brains.csv", header = TRUE, sep = ',')
```

Now, let's look at the data and also determine the types of variables involved:

```
brains
```

```
##    lnMass lnBrain     species
## 1    4.00    7.19      recent
## 2    3.96    7.23      recent
## 3    3.95    7.26      recent
## 4    4.04    7.23      recent
## 5    4.11    7.22      recent
## 6    4.10    7.24      recent
## 7    4.05    7.27      recent
## 8    4.03    7.26      recent
## 9    4.03    7.27      recent
## 10   4.05    7.35      recent
## 11   4.09    7.35      recent
## 12   4.12    7.35      recent
## 13   4.14    7.34      recent
## 14   4.17    7.33      recent
## 15   4.19    7.31      recent
## 16   4.21    7.27      recent
## 17   4.26    7.28      recent
## 18   4.28    7.32      recent
## 19   4.37    7.33      recent
## 20   4.27    7.33      recent
```

```
## 21   4.21    7.33      recent
## 22   4.17    7.36      recent
## 23   4.21    7.38      recent
## 24   4.25    7.35      recent
## 25   4.26    7.37      recent
## 26   4.26    7.39      recent
## 27   4.23    7.42      recent
## 28   4.17    7.44      recent
## 29   4.39    7.54      recent
## 30   4.43    7.48      recent
## 31   4.15    7.15 neanderthal
## 32   4.21    7.17 neanderthal
## 33   4.27    7.21 neanderthal
## 34   4.23    7.35 neanderthal
## 35   4.25    7.47 neanderthal
## 36   4.35    7.40 neanderthal
## 37   4.39    7.38 neanderthal
## 38   4.43    7.35 neanderthal
## 39   4.44    7.43 neanderthal
```

```
str(brains)
```

```
## 'data.frame':    39 obs. of  3 variables:
##  $ lnMass : num  4 3.96 3.95 4.04 4.11 4.1 4.05 4.03 4.03 4.05 ...
##  $ lnBrain: num  7.19 7.23 7.26 7.23 7.22 7.24 7.27 7.26 7.27 7.35 ...
##  $ species: chr  "recent" "recent" "recent" "recent" ...
```

We see that most of the data come from `recent` humans, and only 9 data points come from `neanderthal`. We also see three variables: `lnMass` (body size), `lnBrain` (brain size), and `species`. The two measures of size have already been log-transformed (indicated by the 'ln' at the start of the names). We see that `species` is currently type `chr` (character)m but we want it to be type `Factor`. Therefore, let's concert it:

```
brains$species <- factor(brains$species)
str(brains)
```

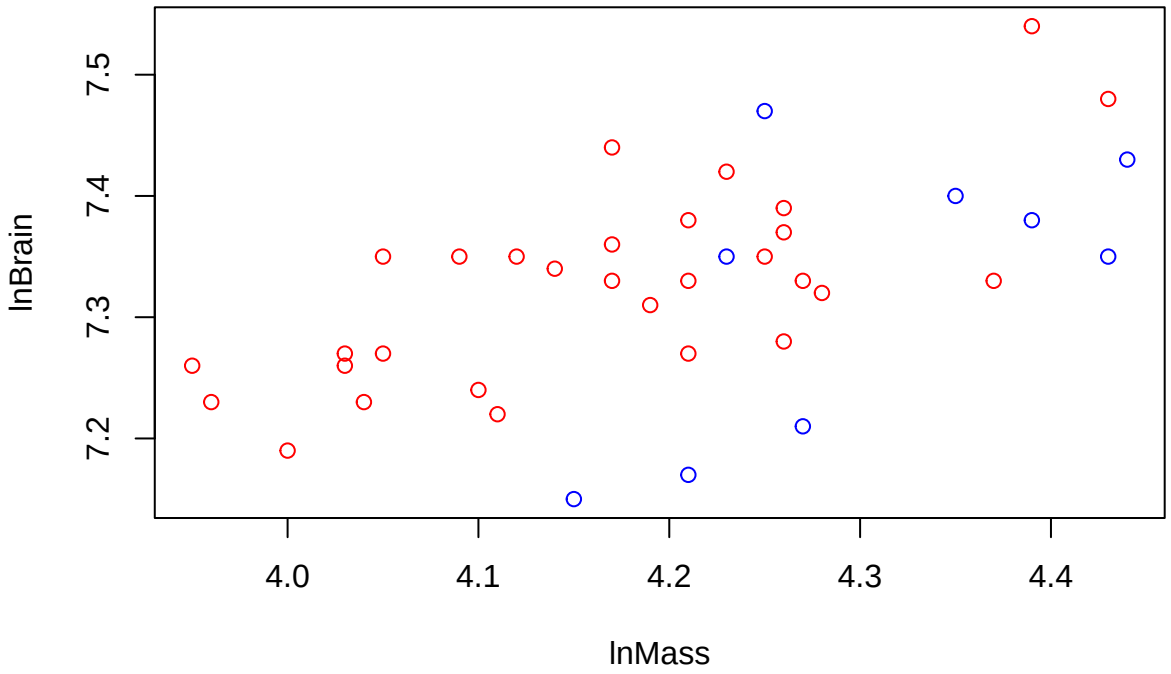
```
## 'data.frame':    39 obs. of  3 variables:
##  $ lnMass : num  4 3.96 3.95 4.04 4.11 4.1 4.05 4.03 4.03 4.05 ...
##  $ lnBrain: num  7.19 7.23 7.26 7.23 7.22 7.24 7.27 7.26 7.27 7.35 ...
##  $ species: Factor w/ 2 levels "neanderthal",..: 2 2 2 2 2 2 2 2 2 2 ...
```

Excellent! Now `species` is a Factor. Our variables are in the form we like.

Before continuing, let's think about our goal for this analysis: we wan to "know whether, after accounting for differences in body size, Neanderthals' brains were of a different size than modern humans?". This implies several things. First, we're foremost interested in whether brain size (`lnBrain`) differs between `species`; in other words, we want to know whether `species` affects `lnBrain`, or, in other words (again), we want to know whether `lnBrain` depends on `species`. Therefore, we know that `lnBrain` is the dependent variable. We're also told that we want to account for body size (`lnMass`) when making this comparison. Why would we do this? Well, consider the humans around you: bigger people tend to have bigger heads, which may lead to variation in brain size simply due to differences in body size. If recent humans and Neanderthals tended to have different body sizes, then differences in bosy size would be an un-interesting reason for differences in brain size. Therefore, we want to know whether, after having controlled for body size (`lnMass`), body size differs between recent humans vs. Neanderthals. For this reason, we'll include `lnMass` as an independent variable in our model: this will allow us to 'control' for body size differences when we compare brain size between recent humans and Neanderthals. In other words, a model like this can effectively allow us to compare brain size of recent humans vs. Neanderthals for individuals of similar size.

Now that we understand our goal, let's start by plotting the data:

```
plot(lnBrain ~ lnMass, data = brains, col=ifelse(species=="recent", "red", "blue"))
```



What do we see?

- As expected, we see that brain size tends to increase with body size.
- This relationship appears to be true for both recent humans (red) and Neanderthals (blue).
- Neanderthals tend to be larger than recent humans.
- The distributions of `lnMass` overlap nicely for recent humans and Neanderthals; this is important we because we need this overlap to run the model we'll use, below.

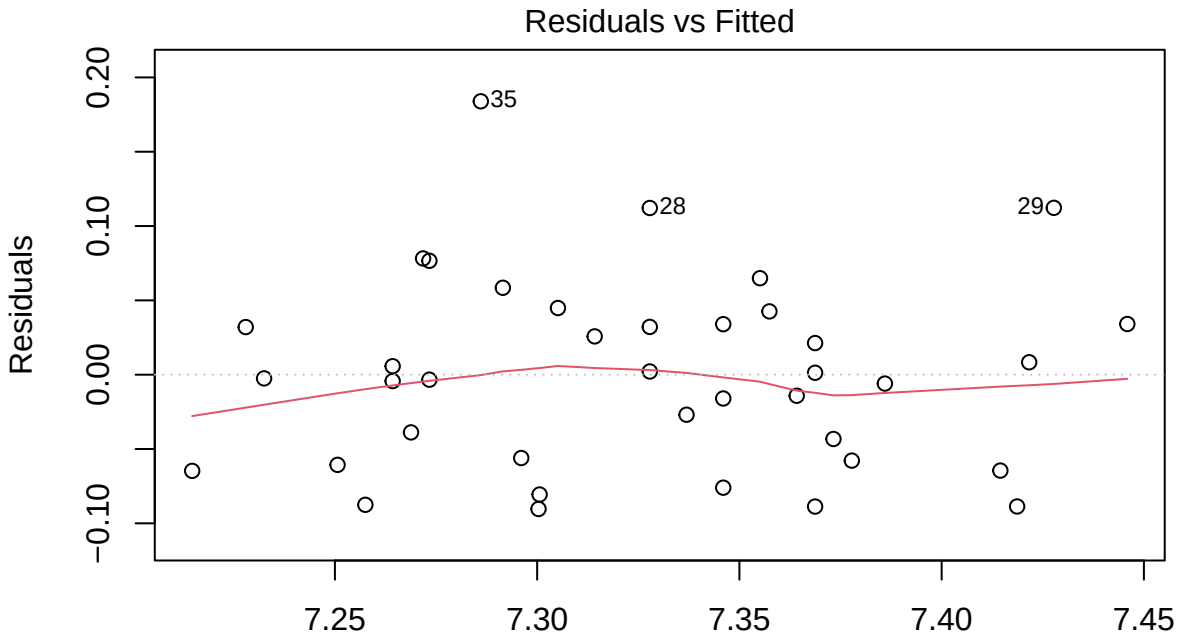
Do recent humans tend to have higher brain size (per unit body mass) than Neanderthals? That's a bit hard to tell from the plot. Let's run our model. The model, below, includes an interaction between `species` and `lnMass`: this allows the model to fit lines with different slopes for the two species.

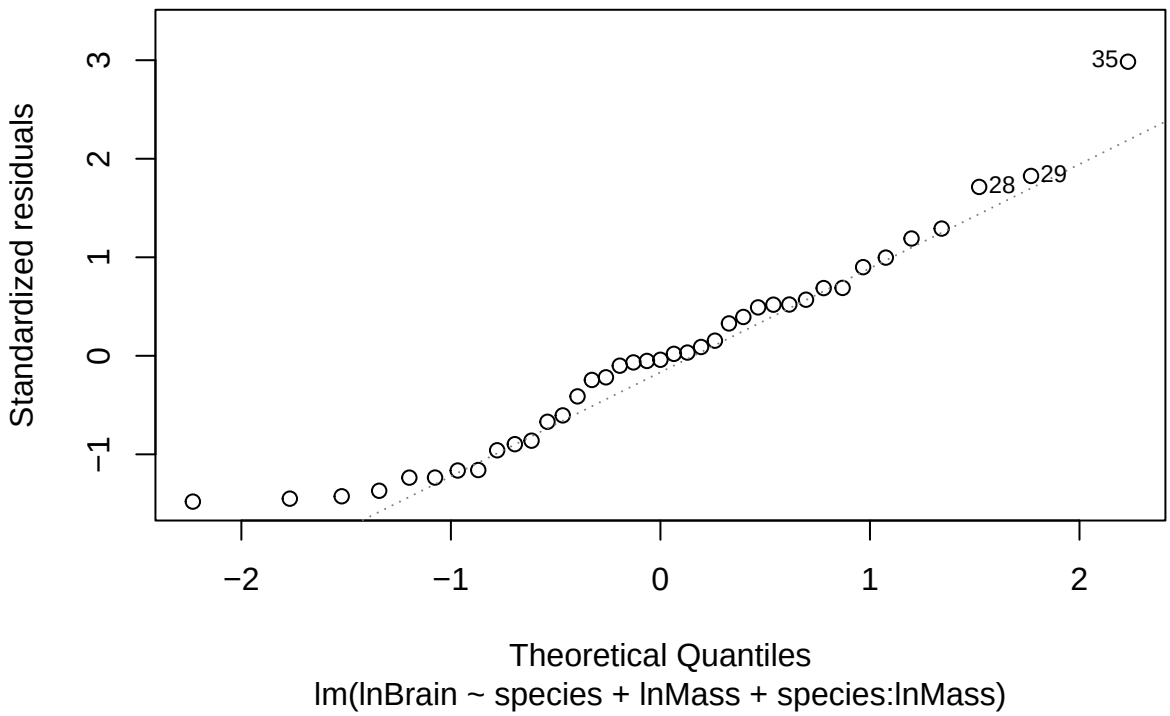
```
brains.lm <- lm(lnBrain ~ species + lnMass + species:lnMass, data = brains, contrasts = list(species = c
```

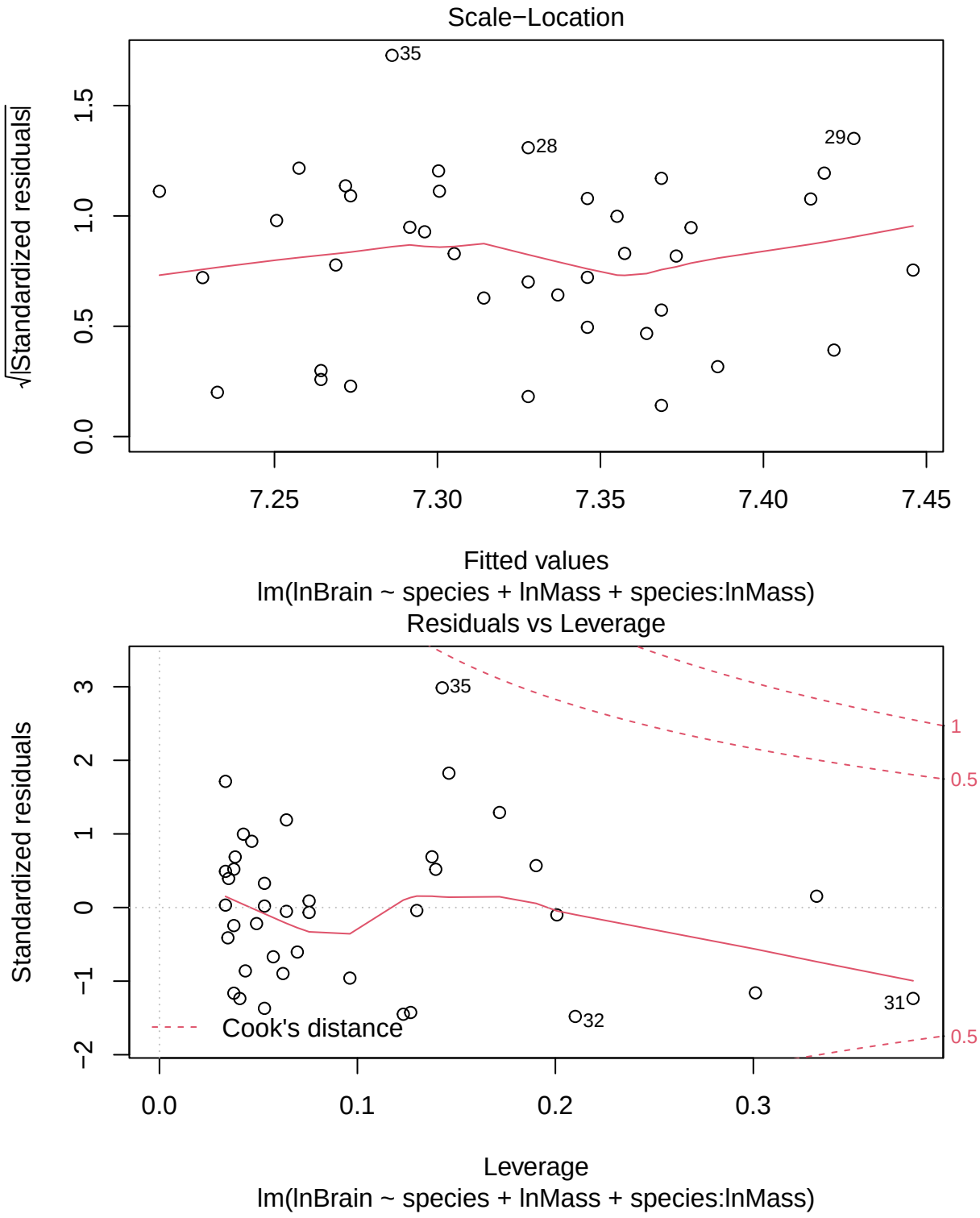
(Notice that we specified contrasts for the Factor, `species`. When analyzing models with both covariate(s) and Factor, we need to calculate p-values as if we have unbalanced data (i.e., we'll use the `Anova()` function, below.))

Let's examine the residuals:

```
plot(brains.lm)
```


Residuals vs Fitted
Residuals
0.20
0.10
0.00
−0.10
35
28
29
7.25
7.30
7.35
7.40
7.45
Fitted values
lm(lnBrain ~ species + lnMass + species:lnMass)


Normal Q−Q
Standardized residuals
3
2
1
0
−1
35
28
29
−2
−1
0
1
2
Theoretical Quantiles
lm(lnBrain ~ species + lnMass + species:lnMass)



Unlike what we saw in Question 1, these residuals indicate the data nicely meet the assumptions: the first and third plot indicate the data meet the assumptions of equal variance, and the second plot indicates the data meet the assumption of normality (not perfectly, but certainly well enough to avoid worry).

Let's check our p-values. We'll use the `Anova()` function, which is found in the `car` library, to specify type 3 sums of squares:

```
library(car)
```

```
## Loading required package: carData
```

```
Anova(brains.lm, type = 3)
```

```
## Anova Table (Type III tests)
##
## Response: lnBrain
##                 Sum Sq Df F value    Pr(>F)
## (Intercept)    0.36840  1 83.1685 8.926e-11 ***
## species        0.00547  1  1.2358    0.2739
## lnMass         0.09810  1 22.1469 3.886e-05 ***
## species:lnMass 0.00485  1  1.0938    0.3028
## Residuals      0.15503 35
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As with any model that includes an interaction, we focus our attention in the interaction: how we proceed depends on whether we have evidence for an interaction. If we find evidence for an interaction, we focus on understanding the interaction; in the present example, we would determine the lines that describe the relationship between `lnMass` and `lnBrain` for each level of `species` (`recent` and `neanderthal`). Note that this approach would not directly answer our original question of whether `lnBrain` differed between `species` (while accounting for `lnMass`). But, that's OK because if we find evidence for an interaction between `species` and `lnMass` then this will mean that there is no straight-forward answer to our original question because the size of the difference in `lnMass` between species would depend on which value of `lnMass` we focus on; therefore, providing equations for the line for each level of `species` will provide the most complete answer that's possible. On the other hand, if we do not find evidence for an interaction, we can focus instead on interpreting the 'main' effects (`species` and `lnMass`) to answer our original question directly.

We see a high p-value for the interaction (`p = 0.3028`), providing little evidence for an interaction. Let's also plot the lines associated with each level of `species`. We can obtain the y-intercept and slope for each species from the `summary()` output of our model. Recall, however, that we included the option, `contrasts = list(species = contr.sum)`, which will change the interpretation of the coefficients. Therefore, we need to re-run our model without this option to obtain coefficients (i.e., values of y-intercepts and slopes) that we can interpret appropriately:

```
brains.lm.2 <- lm(lnBrain ~ species + lnMass + species:lnMass, data = brains)
```

Now, let's look at the `summary()` output:

```
summary(brains.lm.2)
```

```
##
## Call:
## lm(formula = lnBrain ~ species + lnMass + species:lnMass, data = brains)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.090341 -0.056928 -0.002495  0.034044  0.183930
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)            4.2535     0.9769   4.354 0.000111 ***
## speciesrecent          1.1809     1.0623   1.112 0.273862
## lnMass                 0.7135     0.2270   3.143 0.003395 **
## speciesrecent:lnMass  -0.2595     0.2481  -1.046 0.302790
```

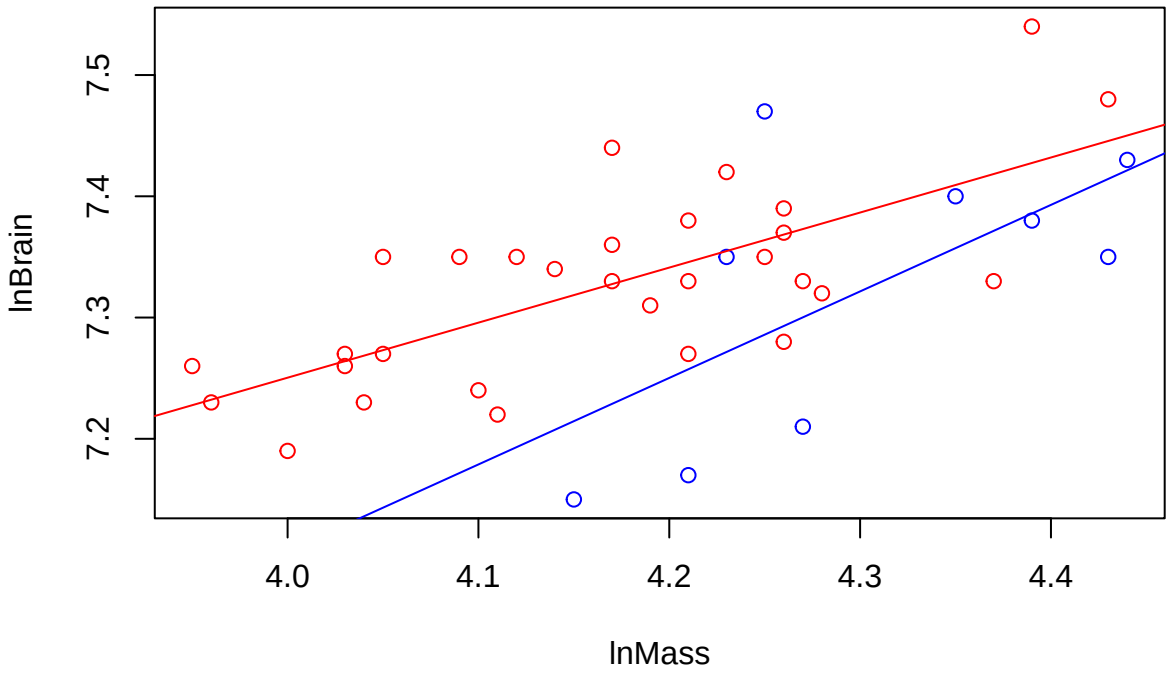
```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06655 on 35 degrees of freedom
## Multiple R-squared:  0.4653, Adjusted R-squared:  0.4195
## F-statistic: 10.15 on 3 and 35 DF,  p-value: 5.908e-05
```

The first two rows correspond to values that we need to calculate the y-intercepts of each species. The first row (`(Intercept)`) indicates the y-intercept for `neanderthal` (notice that `neanderthal` is not listed in the output, but `recent` is). Therefore, the y-intercept of `neanderthal` equals 4.2535. The second row (`speciesrecent`) indicates the *difference* between the y-intercept of `neanderthal` and that of `recent`, which equals 1.1809. Therefore, the y-intercept of `recent` equals `4.2535 + 1.1809 = 5.4344`

As you may guess, the latter two rows provide information about the slopes. `lnMass` indicates the slope for `neanderthal` (0.7135), and `speciesrecent:lnMass` indicates the difference between the slopes of `neanderthal` and `recent` (-0.2595). Therefore, the slope for `recent` equals `0.7135 - 0.2595 = 0.454`.

With these values in mind, we can add lines to our plot, above, as follows:

```
plot(lnBrain ~ lnMass, data = brains, col=ifelse(species=="recent", "red", "blue"))
#add line for neanderthal:
abline(4.2535,0.7135, col = "blue")
#add line for recent:
abline(5.4344,0.454, col = "red")
```



Note that slopes *appear* to be appreciably different (but this apparent difference might not be real; it might be due to sampling error). Recall that the p-value for the interaction is high, however, providing no evidence for 'real' differences between these slopes. This situation will have arisen be due to large uncertainty (sampling error) in the estimate of the difference between these slopes: basically we're in a situation where we have no statistical evidence for differences in slopes, but the raw estimates of the slopes are appreciably different. So, what do we do? Given that our p-value for the interaction is so large, we'll proceed with the assumption that there is no interaction. But, we'll proceed cautiously: we'll proceed with **an analysis that aims to estimate the difference in mean value between the two species**, and we'll use two approaches. If the two approaches give the same general answers then we can feel confident in our conclusions. The two approaches include 1) keeping the interaction term in the model and 2) dropping the interaction term from

the model. **Whether to drop an interaction term from a model when we find little statistical evidence for the interaction is a big question, and falls under the topic of 'model selection', which we will address in the future.** Note that, in our present example, we're going to examine results from both of these models for a specific purpose: **to determine whether including the interaction appreciably affects our estimate of differences between species.** If we obtain similar answers from the two approaches then we can feel confident in our conclusions, regardless of the possible effect of an interaction.

Approach 1: Model with interaction

We'll begin by estimating the difference in `lnMass` between levels of `species` for a model that includes the interaction, `species:lnMass`.

This is the model we performed just above, where we saved the model output in the object, `brains.lm.2` (similarly we could use the output in `brains.lm`). So let's proceed by giving this object to `emmeans` to estimate the effect of `species`. (Remember that we have to open the `emmeans` library, too.)

```
library(emmeans)
brains.emmeans <- emmeans(brains.lm.2,"species")
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

Note that `emmeans` provides the warning, `NOTE: Results may be misleading due to involvement in interactions`. This is worth noting! It is also the exact reason why we will be analyzing models with and without the interaction in order to determine whether an interaction affects our conclusions.

Note that we may not be interested in knowing the average value of `lnBrain` for each `species` because it appears that `lnBrain` depends on body size (`lnMass`); but, we can obtain such estimates of we wish (just enter `brains.emmeans`).

Let's focus on an estimate of the difference in `lnBrain` between levels of `species`, `neanderthal` and `recent`. We can obtain this estimate with the `pairs()` function (found in the `emmeans` library):

```
pairs(brains.emmeans)
```

```
##  contrast             estimate     SE df t.ratio p.value
##  neanderthal - recent  -0.0916 0.0348 35 -2.634  0.0125
```

This output provides several insights:

- It provides the `estimate` for the contrast, `neanderthal - recent`. This `estimate` equals, `-0.0916` (with SE = 0.0348). The fact that this estimate is negative indicates that the mean `lnBrain` is larger for `recent` than `neanderthal` (i.e., we subtracted a larger number from a smaller one). Therefore, this result implies that, when accounting for body size (`lnMass`), brain size (`lnBrain`) is larger for `recent` than `neanderthal` by 0.0916 (SE = 0.0348), remembering that these measurements are on the log-scale.
- Note that the p-value for this contrast is smaller than that from the output from, `Anova(brains.lm, type = 3)`, above. Now, the p-value equals 0.0125, providing moderate or 'suggestive' evidence for a difference in `lnBrain` between `recent` and `neanderthal` (when accounting for `lnMass`). This is the p-value we would report. (Note that the differences in p-values between the two sets of output arises due to a phenomenon called, 'variance inflation', which we will likely discuss in the future.)
- We would report all of the output provided here (`estimate`, `SE`, `df`, `t.ratio`, `p.value`).

Finally, remember that we can obtain 95% Confidence Intervals for our results, above:

```
confint(pairs(brains.emmeans))
```

```
##  contrast             estimate     SE df lower.CL upper.CL
##  neanderthal - recent  -0.0916 0.0348 35   -0.162   -0.021
##
## Confidence level used: 0.95
```

These results indicate that the `contrast`, above, has 95% CI's of -0.162 to -0.021. Therefore, when accounting for `lnMass`, mean `lnBrain` of `recent` is plausibly greater than that of `neanderthal` by 0.021 to 0.162 (again, remember that we're on the log scale). We would report these results.

In a moment, we'll compare these results to those from a model that does not include an interaction. Before we do so, however, let's examine the slopes for each level of `species`. We do so using the function `emtrends` (in the `emmeans` library), like this:

```
brains.trends <- emtrends(brains.lm.2, "species", var = "lnMass")
brains.trends
```

```
##  species     lnMass.trend    SE df lower.CL upper.CL
##  neanderthal        0.714 0.227 35    0.253    1.174
##  recent             0.454 0.100 35    0.251    0.657
##
## Confidence level used: 0.95
```

This code allows us to calculate the slope for the relationship between `lnMass` and `lnBrain` separately for each level of `species`.

Note a few things from this output:

- The slopes for both `neanderthal` and `recent` are both positive, implying that `lnBrain` tends to increase with `lnMass` in both species (we also find SE's for these slope estimates).
- The output also provides 95% CI's for each slope. Notice that the 95% CI's do not include the value, zero. This implies that 'zero' is not a very plausible slope. This, in turn, implies that we have `p < 0.05` with respect to the slope for both levels of `species`.

Approach 2: Model without interaction

Now, we will repeat the process, above, but for a model in which we remove the interaction. As the overall process is similar, we'll just jump into the code ans look at the final output:

```
#Model without interaction:
brains.lm.noInteract <- lm(lnBrain ~ species + lnMass,
                           data = brains)
brains.noInt.emmeans <- emmeans(brains.lm.noInteract, "species")
pairs(brains.noInt.emmeans)
```

```
##  contrast             estimate     SE df t.ratio p.value
##  neanderthal - recent  -0.0703 0.0282 36 -2.491  0.0175
```

```
confint(pairs(brains.noInt.emmeans))
```

```
##  contrast             estimate     SE df lower.CL upper.CL
##  neanderthal - recent  -0.0703 0.0282 36   -0.128  -0.0131
##
## Confidence level used: 0.95
```

What do we notice when we compare this output to that from the model that included the interaction?

- `emmeans` did not provide the warning that we saw, above, related to including an interaction in the model.
- the contrast estimate from the model without the interaction (-0.0703; SE = 0.0282) is fairly similar to that from the model with the interaction included (-0.0916; SE = 0.0348). Therefore, the results are fairly consistent so far.
- the p-values are also similar for this contrast: p = 0.0175 vs. p = 0.0125. Again, not much difference between the models.
- the 95% CI's for the contrast in the model without the interaction (-0.128 to -0.0131) are similar to those from the model that included the interaction term, (-0.162 to -0.021).

Overall, the conclusions are very similar. Which model you use to report your results is an open question: however, when you do report your results, it would be wise to mention that you ran both model types and that the results led to similar conclusions; you could (should?) even report the results from the alternative model in Online Supplementary Materials if you are publishing your results. This leads to greater transparency.

2. keep interaction in - note that p-value for comparison between recent and Neanderthal can be different from what we obtained above, in output from `Anova()` (`p = 0.2739` for `species` effect).

A final note on the process we have taken: Note that, in practice, we should have a defined protocol for how to model the data before we set out to analyze it, and we should stick to that protocol, regardless of whether the outcome supports your favourite hypothesis or not. (Of course, our protocol may need to change if unexpected circumstances arise, such as the data do not conform to the assumptions of our anticipated model.) Why do I mention this? Because in these practice problem answers I've analyzed the results using two models: **I do not want to give the impression that it is wise to analyze the data multiple ways and then select the result that you 'prefer'.** From another perspective, however, please note that it is absolutely fine, and may be encouraged, to analyze the data multiple ways, especially **if your goal is to ensure that your conclusions are robust to different forms of analysis**. Our analysis, here, followed this spirit: We analyzed the same dataset in multiple ways to determine whether the approach to the analysis affects our conclusions. Importantly, if we had found that our approaches had yielded different conclusions, we absolutely should report the results and conclusions from both analysis approaches to ensure transparency. i.e., we absolutely do not want to simply select the results that we 'prefer' and report those.

Question 3

Part (a)

This question uses a dataset already in **R**, the `Puromycin` dataset. Let's obtain these data and look at them:

```
data(Puromycin)
str(Puromycin)
```

```
## 'data.frame':    23 obs. of  3 variables:
##  $ conc : num  0.02 0.02 0.06 0.06 0.11 0.11 0.22 0.22 0.56 0.56 ...
##  $ rate : num  76 47 97 107 123 139 159 152 191 201 ...
##  $ state: Factor w/ 2 levels "treated","untreated": 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "reference")= chr "A1.3, p. 269"
```

R's help option describes this dataset as, "The Puromycin data frame has 23 rows and 3 columns of the reaction velocity versus substrate concentration in an enzymatic reaction involving untreated cells or cells treated with Puromycin."

Based on this description, we infer that `rate` is the *dependent* variable; we want to know how `conc` (i.e., substrate 'concentration', a numeric variable) and `state`, a Factor with 2 levels, `treated` and `untreated`, affect reaction rate.

Based on this background, we can address three questions with these data:

1. Does `state` affect reaction rate ('rate")?
2. Does substrate concentration (`conc`) affect reaction rate?
3. Does the way in which `state` affect `rate` depend on `conc`? i.e., do `state` and `conc` interact in their effects on `rate`?

Before we proceed, let's look at the whole dataset, just to get a sense of how much data we have:

```
Puromycin
```

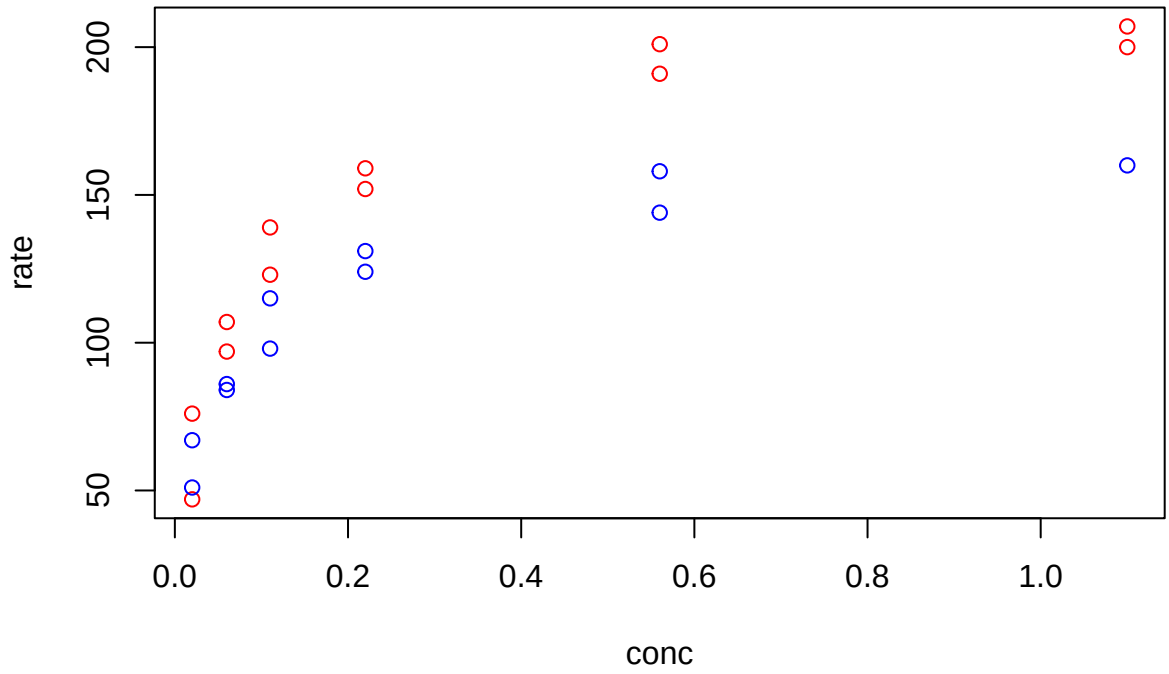
```
##    conc rate     state
## 1  0.02   76   treated
## 2  0.02   47   treated
## 3  0.06   97   treated
```

```
## 4  0.06  107   treated
## 5  0.11  123   treated
## 6  0.11  139   treated
## 7  0.22  159   treated
## 8  0.22  152   treated
## 9  0.56  191   treated
## 10 0.56  201   treated
## 11 1.10  207   treated
## 12 1.10  200   treated
## 13 0.02   67 untreated
## 14 0.02   51 untreated
## 15 0.06   84 untreated
## 16 0.06   86 untreated
## 17 0.11   98 untreated
## 18 0.11  115 untreated
## 19 0.22  131 untreated
## 20 0.22  124 untreated
## 21 0.56  144 untreated
## 22 0.56  158 untreated
## 23 1.10  160 untreated
```

Parts (b) and (c)

Let's plot the data:

```
plot(rate ~ conc, data = Puromycin, col=ifelse(state=="treated", "red", "blue"))
```

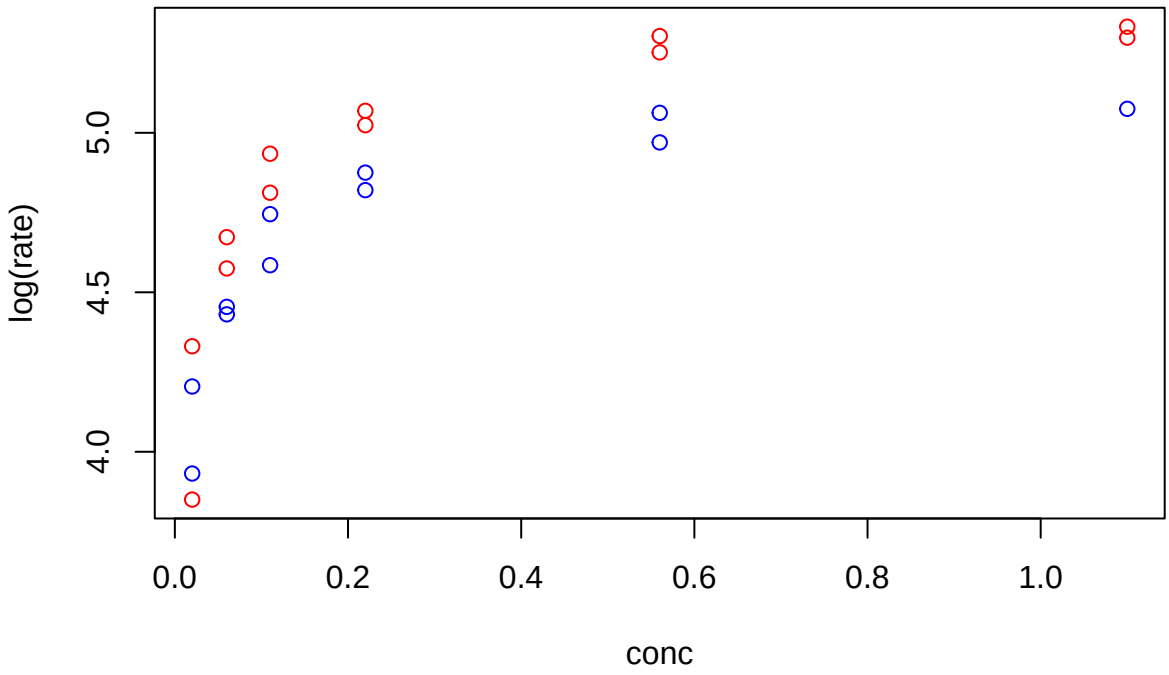


In this plot, we've made the `treated` data red, and `untreated` data blue.

These data are clearly not straight lines. Let's try to straighten them by transforming the y- and/or x-axes:

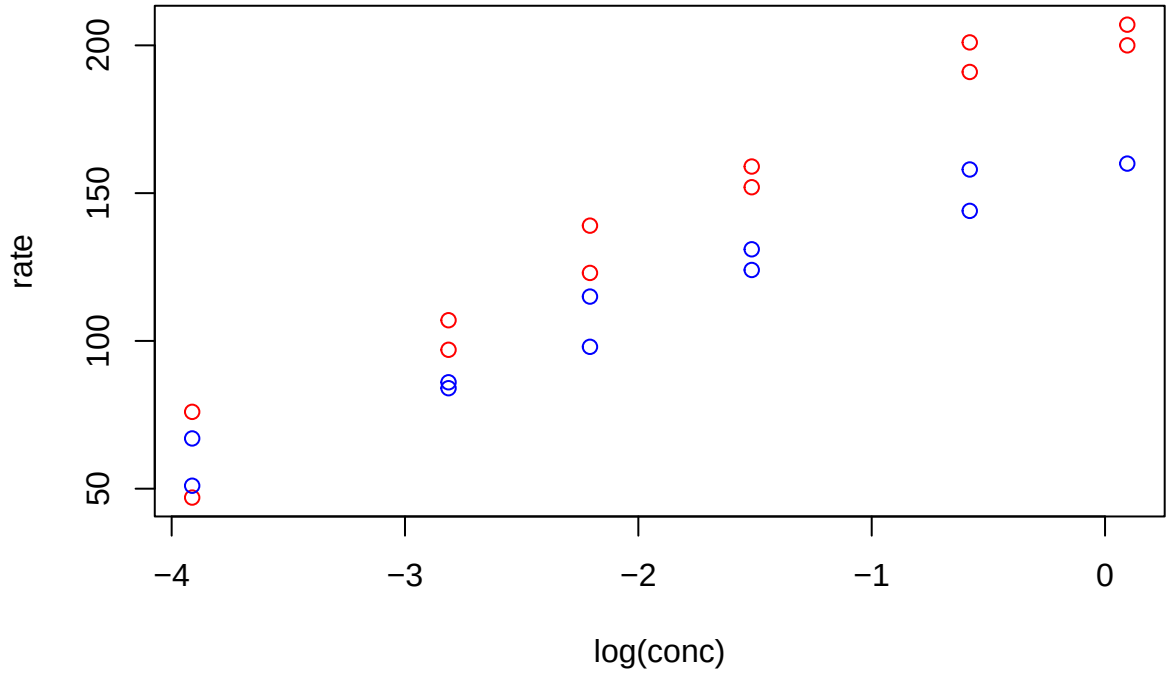
Transform the y-axis only:

```
plot(log(rate) ~ conc, data = Puromycin, col=ifelse(state=="treated", "red", "blue"))
```



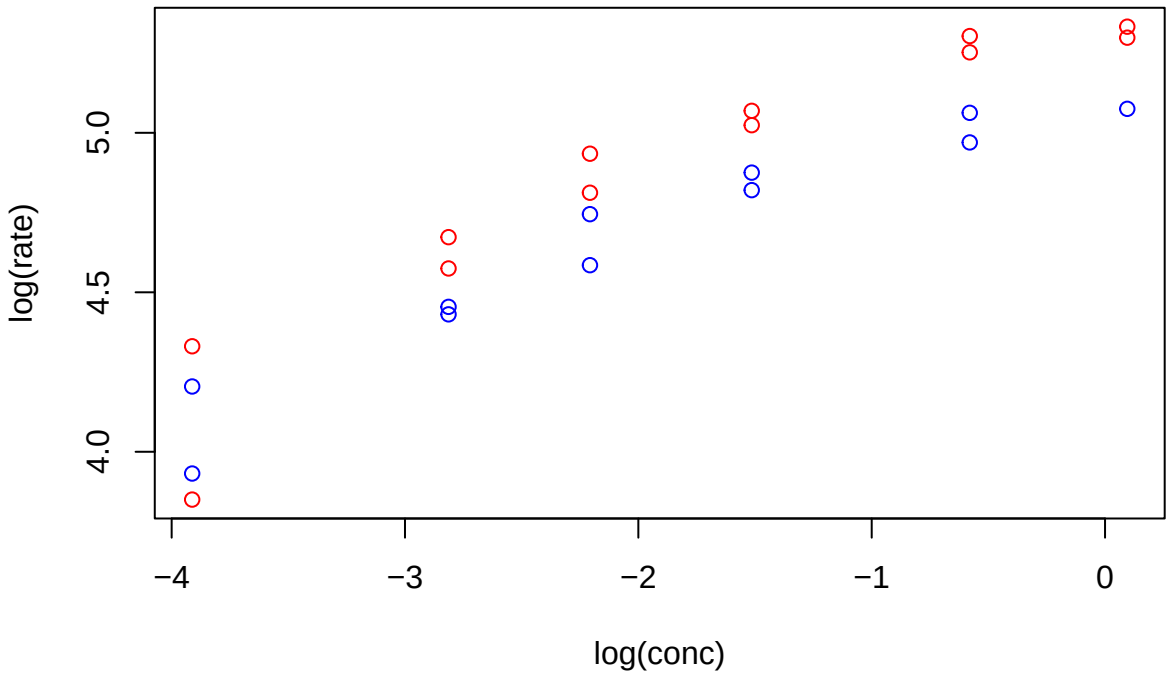
Transform the x-axis only:

```
plot(rate ~ log(conc), data = Puromycin, col=ifelse(state=="treated", "red", "blue"))
```



Transform both the y- and x-axis:

```
plot(log(rate) ~ log(conc), data = Puromycin, col=ifelse(state=="treated", "red", "blue"))
```



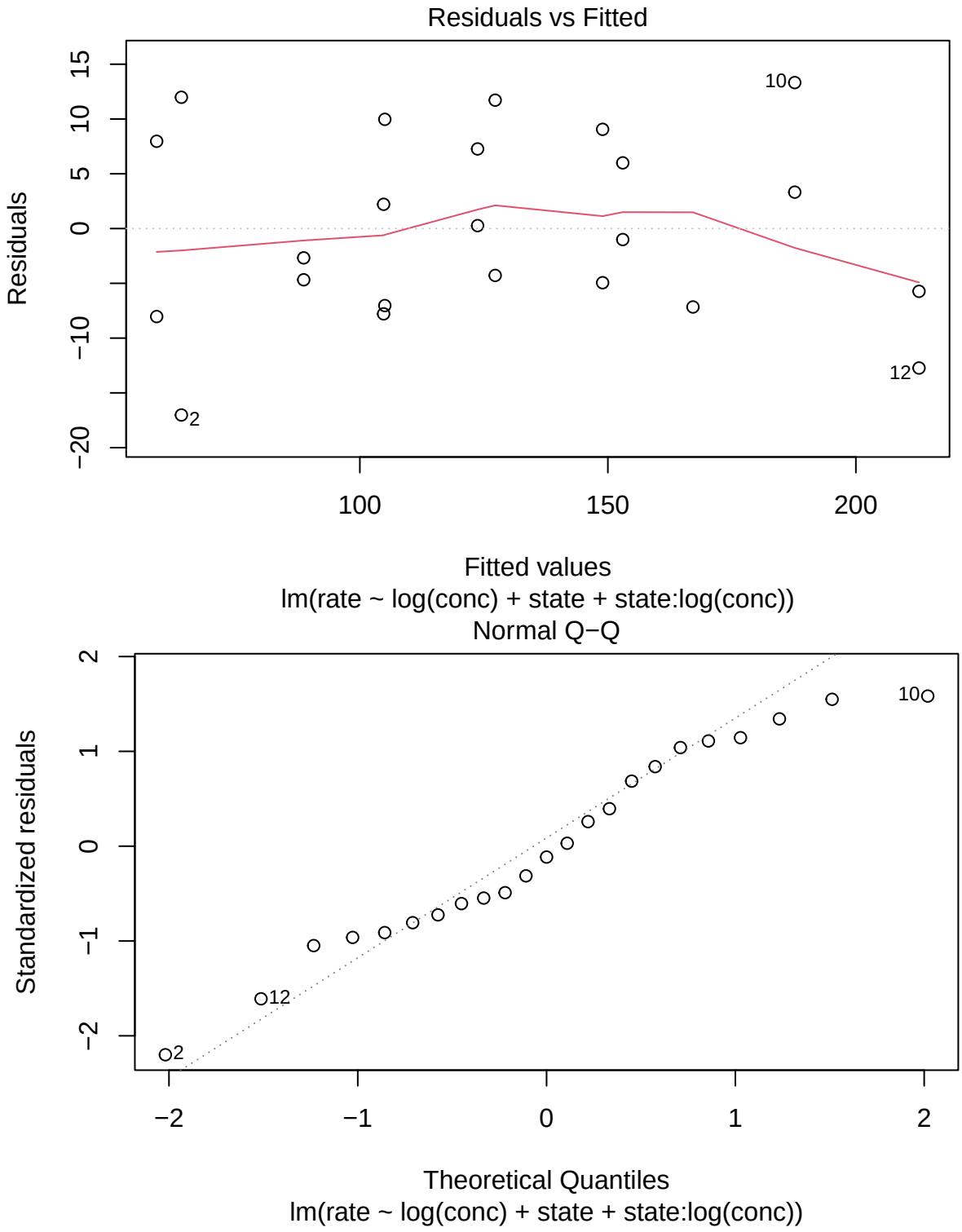
To my eye, the data are most linear when we transform the x-axis only. Therefore, let's model the data with log-transformed values of `conc`. Note that this highlights that we can transform *independent* variables, as well as *dependent* variables.

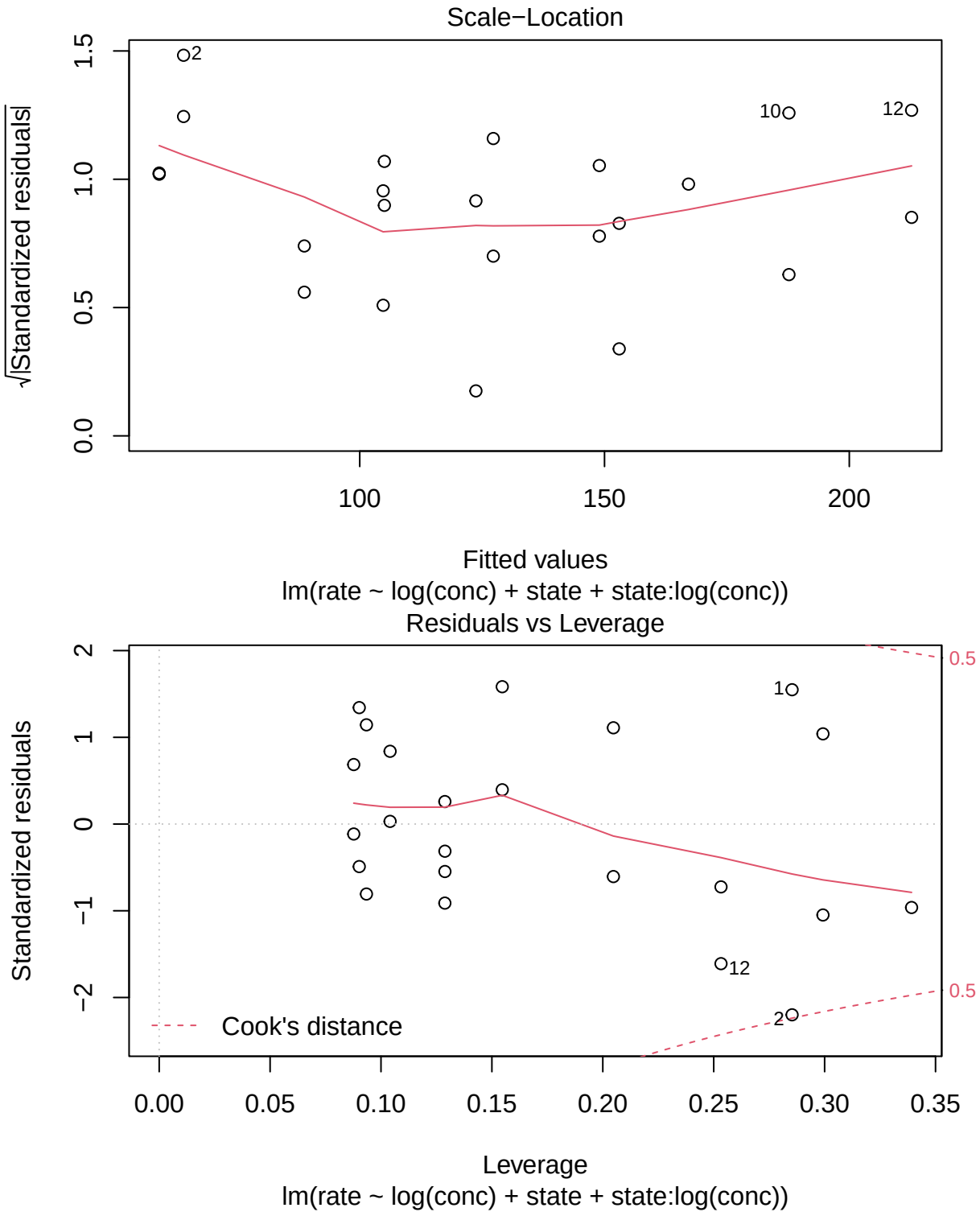
Let's form our model:

```
pur.lm <- lm(rate ~ log(conc) + state + state:log(conc), data = Puromycin)
```

Now, let's plot the residuals to check the assumptions. (As we know nothing about how these data were collected, we'll assume the data meet the assumptions of randomization and independence.)

```
plot(pur.lm)
```


Residuals vs Fitted
Residuals
15
10
5
0
−10
−20
100
150
200
10
2
12
Fitted values
lm(rate ~ log(conc) + state + state:log(conc))
Normal Q−Q
Standardized residuals
2
1
0
−1
−2
−2
−1
0
1
2
10
12
2
Theoretical Quantiles
lm(rate ~ log(conc) + state + state:log(conc))



What do these plopts tell us?

- Plot 1: Notice the downward curvature in these points: the points almost have a rainbow shape. The fact that these residuals do not form a nice 'cloud' (with no discernible shape) implies that we've not managed to fit an appropriate line through the data. i.e., we're attempting to fit a straight line through (albeit transformed) data, and this approach does not adequately capture the shape of the trends in the data. Perhaps this is not surprising: enzyme kinetics can lead to 'saturating' functions (i.e., they reach

an asymptote), which made it harder to model data such as these with `lm()`. (Alternative analysis methods could include, for example, non-linear regression, which we have not yet taught).
- Plot 2: the residuals are not too far off from a normal distribution (good!).
- Plot 3: The curvature in this plot suggests that the data violate the assumptions of equal (homogeneous) variance.

Overall, these residual plots cause us to worry about our model. Perhaps `lm()` is not the appropriate approach for these data, and we should use an alternative approach. This happens in data analysis, and we should always bear such possibilities in mind. However, given that we've not learned any other approaches yet, **we'll continue using `lm()`, but we'll do so with full awareness that this approach brings serious concern. We are only continuing our analysis using `lm()` in order to gain further practice.**

Now, above, we forgot to include `contrasts` to allow us to calculate p-values using type 3 sum of squares. Let's re-formulate our model, and then obtain appropriate p-values:

```
pur.lm.con <- lm(rate ~ log(conc) + state + state:log(conc), data = Puromycin,
                 contrasts = list(state = contr.sum))
library(car)
Anova(pur.lm.con, type = 3)
```

```
## Anova Table (Type III tests)
##
## Response: rate
##                 Sum Sq Df  F value    Pr(>F)
## (Intercept)     252232  1 3011.731 < 2.2e-16 ***
## log(conc)        39881  1  476.194 6.491e-15 ***
## state             3592  1   42.891 2.854e-06 ***
## log(conc):state    996  1   11.892  0.002692 **
## Residuals         1591 19
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We find strong evidence ($p = 0.002692$) for an interaction between `state` and `log(conc)`, implying that the relationship between `log(conc)` and `rate` depends on whether we consider the `treated` or `untreated` groups of `state`. (Remember to not take these results seriously, as we know our data violate the assumptions of the model).

In this case, we may be most interested to know the slope for each relationship. (The intercept will be biologically uninteresting, because we know that the reaction rate (`rate`) will equal zero when `conc` equals zero, at least when we've fit an appropriate line to the data, *which we have not managed to do.*) We can obtain the slopes, their SE's and 95% CI's using `emtrends`:

```
#Remember to open the emmeans library, which we did in Question 2...
emtrends(pur.lm.con, "state", var = "log(conc)")
```

```
##  state     log(conc).trend   SE df lower.CL upper.CL
##  treated              37.1 1.97 19     33.0     41.2
##  untreated            27.0 2.18 19     22.4     31.5
##
## Confidence level used: 0.95
```

These are the results we'd report, along with the output from `Anova()`, above (along with plots of the data, etc).