Lab 9 - Linear and Multivariate Models

Contents

Lab Objectives:

- Examining residuals in regressions
- Dealing with outliers in regressions
- Multivariate linear models

Checking assumptions from a regression

Blue tit cap color

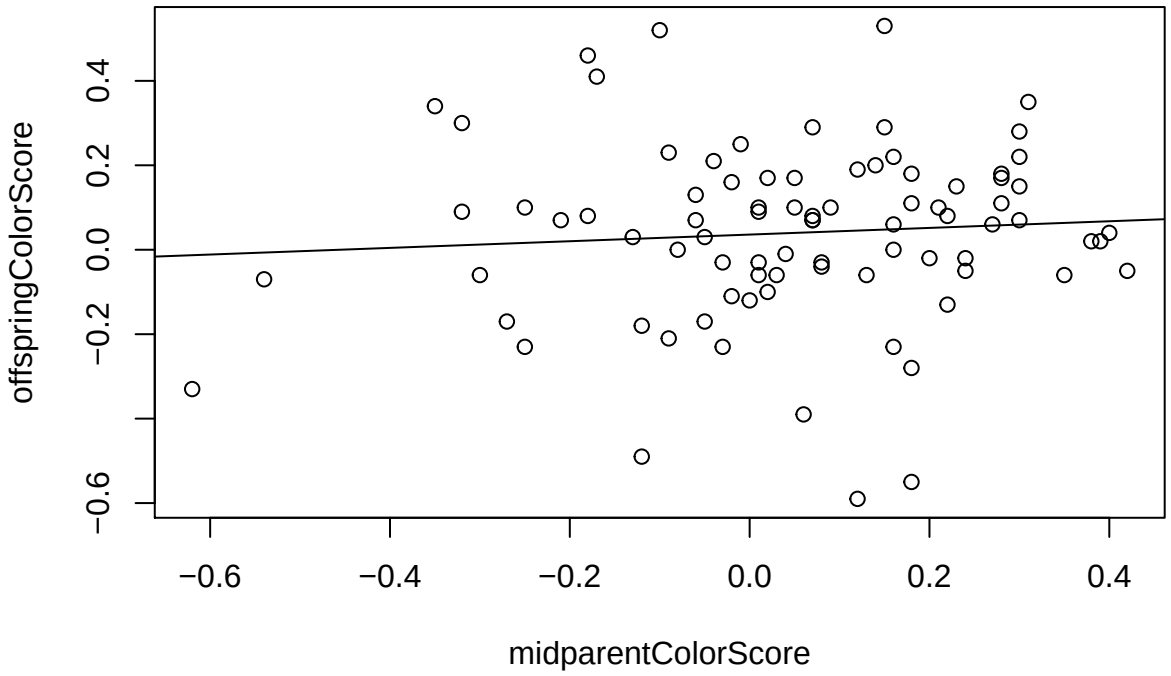
Create a residual plot to check assumptions. The data are cap color of offspring and parents in a sample of the blue tit.

Read and inspect the data.

```
d <- read.csv("capcolour.csv")
```

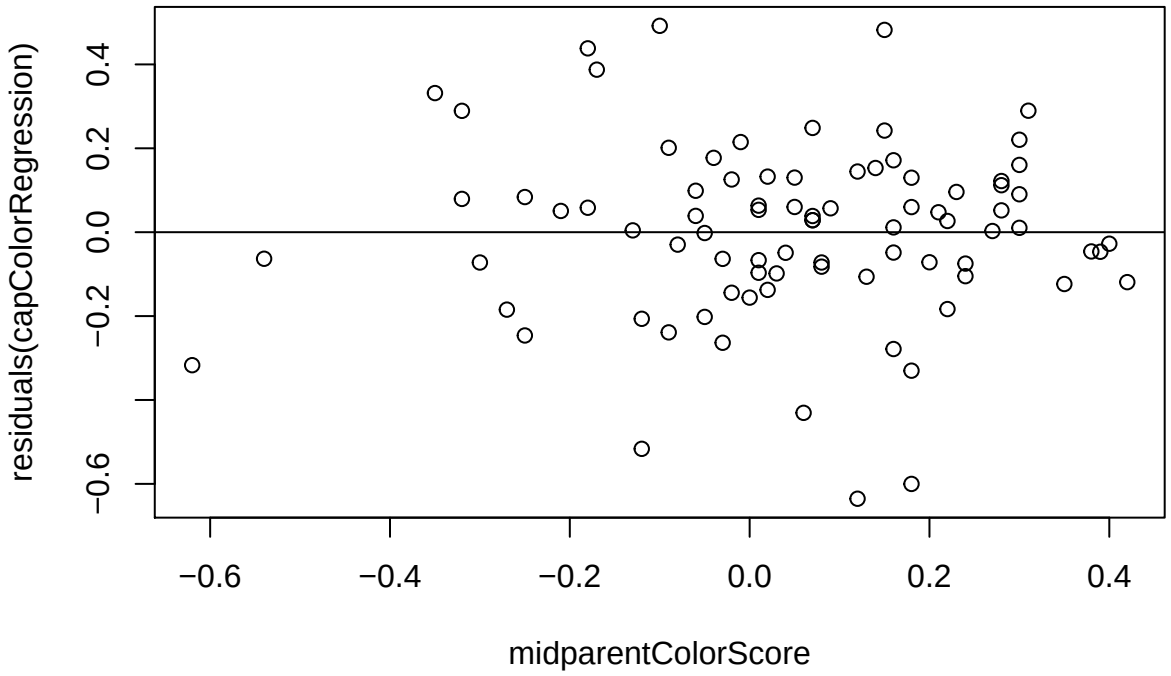
Basic scatter plot with regression line.

```
plot(offspringColorScore ~ midparentColorScore, data = d)
capColorRegression <- lm(offspringColorScore ~ midparentColorScore, data = d)
abline(capColorRegression)
```



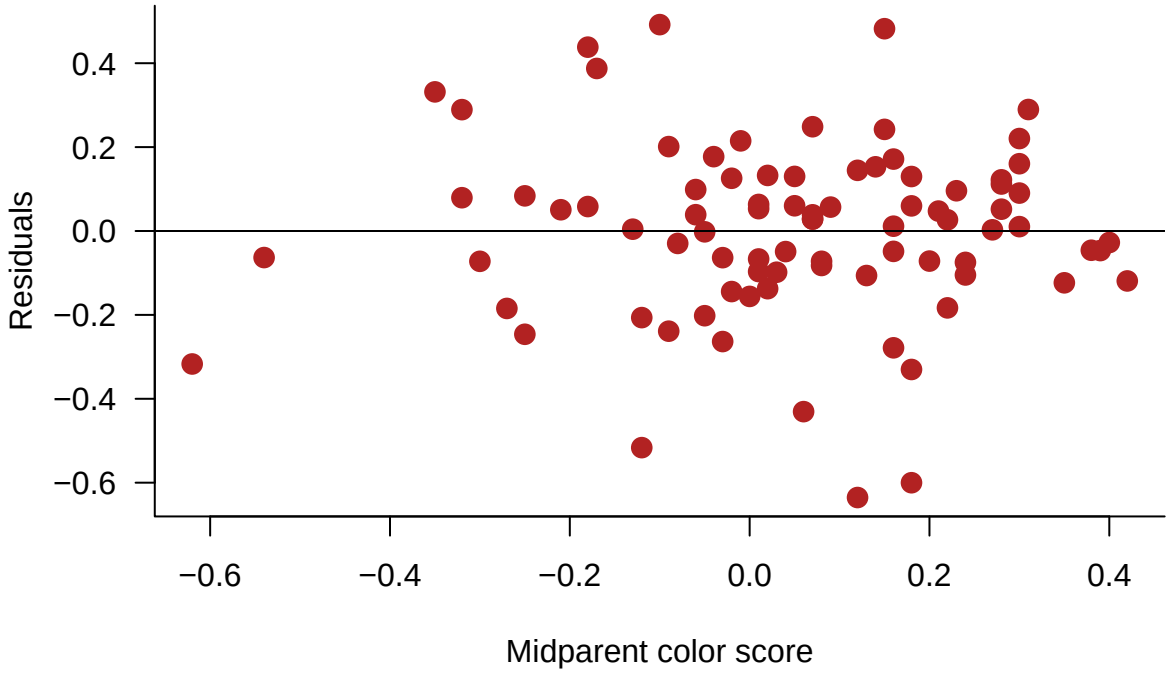
Residual plot. Residuals are calculated from the model object created from the previous lm command, using the residuals command.

```
plot(residuals(capColorRegression) ~ midparentColorScore, data = d)
abline(c(0, 0))
```



Commands for a fancier residual plot using more options:

```
plot(residuals(capColorRegression) ~ midparentColorScore, data = d, pch = 16,
    col = "firebrick", las = 1, cex = 1.5, bty = "l", xlab = "Midparent color score",
    ylab = "Residuals")
abline(c(0, 0))
```



Dealing with Outliers

Dark-eye juncos

Examine the effect on an outlier on linear regression. The data are percentage of white in the tail feathers of the dark-eyed junco at sites at different latitudes.

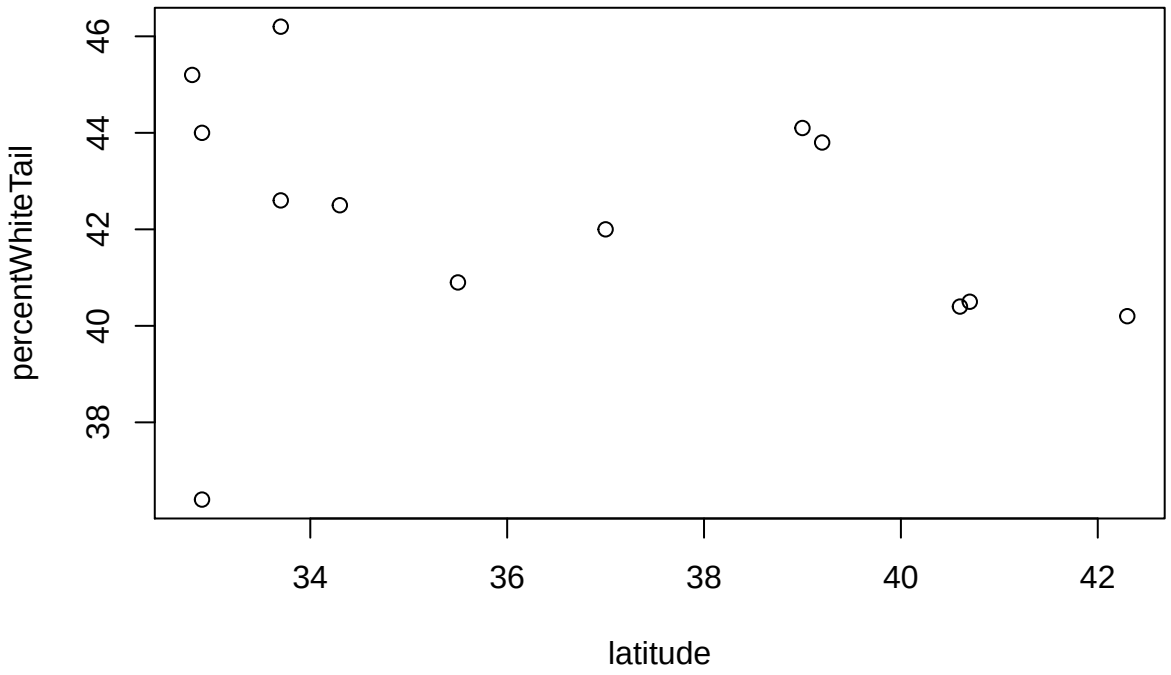
Read and inspect data:

```
junco <- read.csv("junco.csv")
head(junco)
```

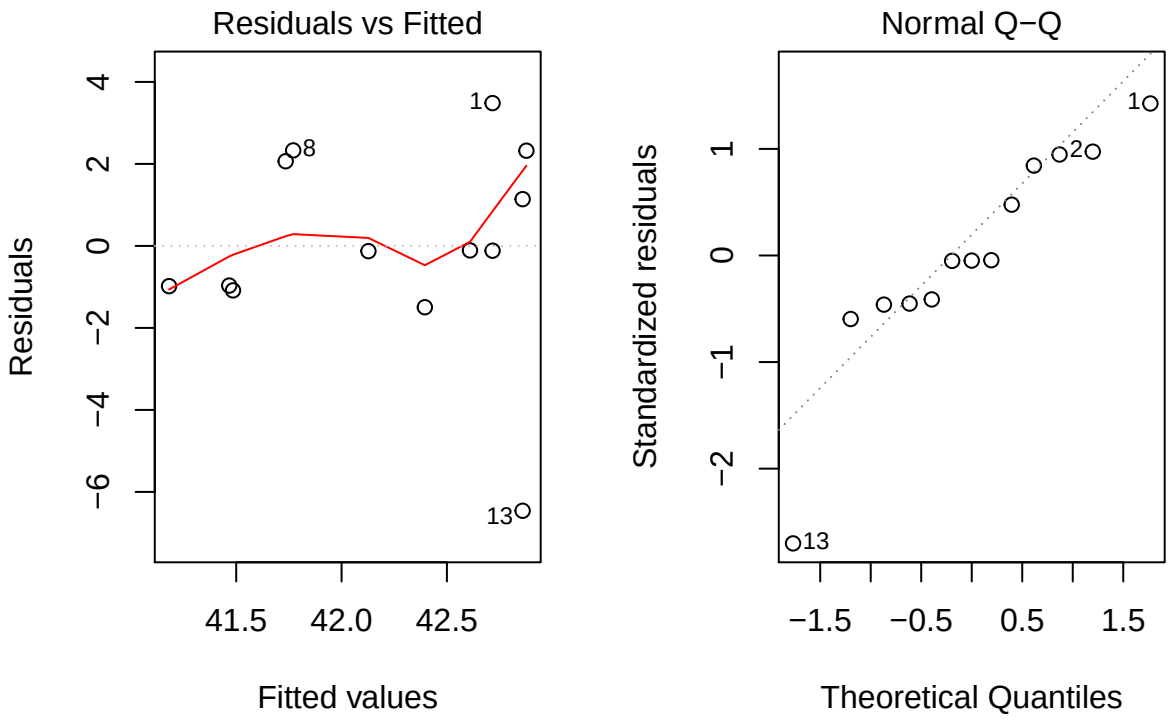
```
##   X latitude percentWhiteTail
## 1 1     33.7             46.2
## 2 2     32.8             45.2
## 3 3     32.9             44.0
## 4 4     33.7             42.6
## 5 5     34.3             42.5
## 6 6     35.5             40.9
```

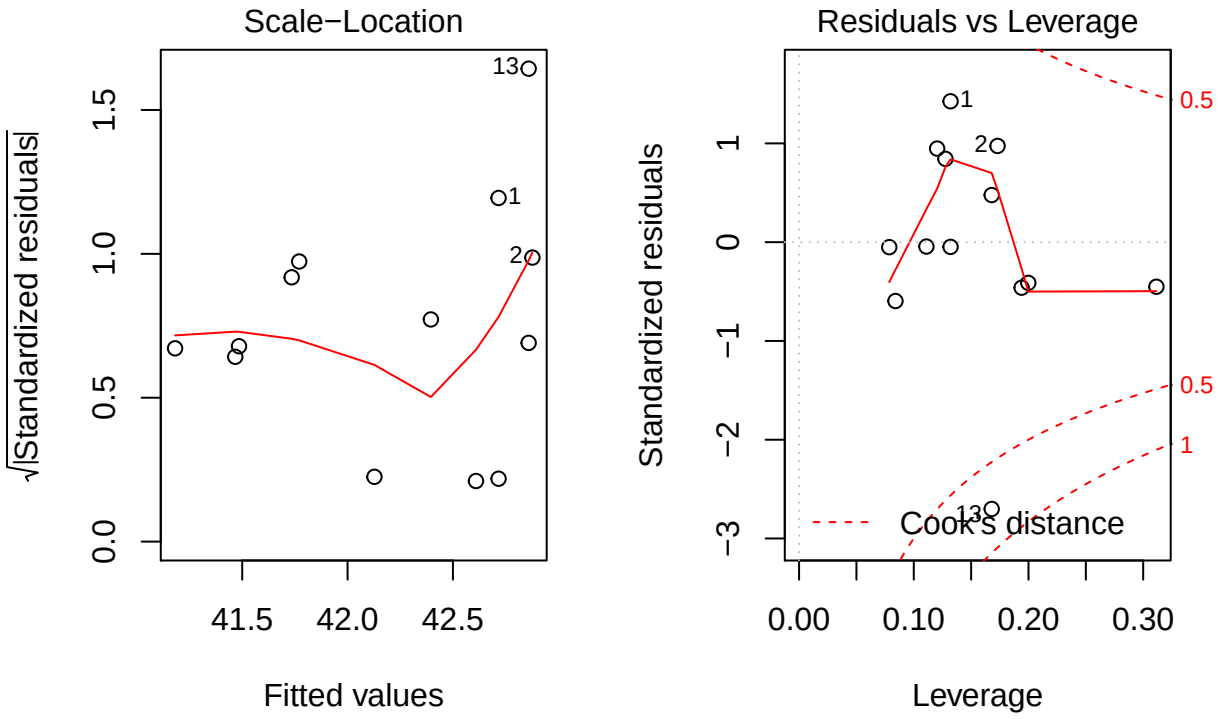
Scatter plot of all the data.

```
plot(percentWhiteTail ~ latitude, data = junco)
```




```
juncoRegression <- lm(percentWhiteTail ~ latitude, data = junco)
oldpar <- par()
par(mfrow = c(1, 2))
plot(juncoRegression)
```





From the diagnostic plots, it is apparent that one data point is influential. See the 4th plot above, where one point has an elevated "Cook's distance".

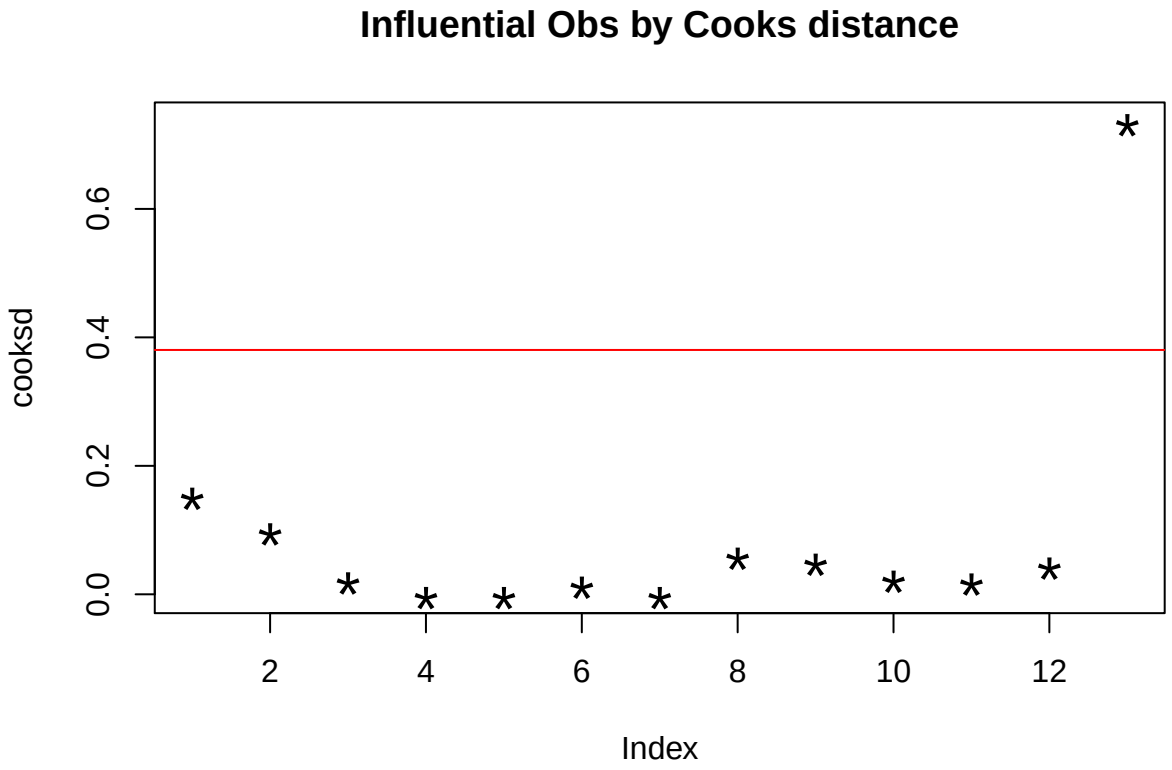
The cook's distance for each observation i measures the change in $\hat{Y}$ (fitted Y) for all observations with and without the presence of observation i, so we know how much the observation i impacted the fitted values.

Mathematically, cook's distance Di for observation i is computed as:

In general use, those observations that have a cook's distance greater than 4 times the mean cook's distances may be classified as influential. This is not a hard boundary.

Alternatively you can calculate the Cook's distance from the regression fit:

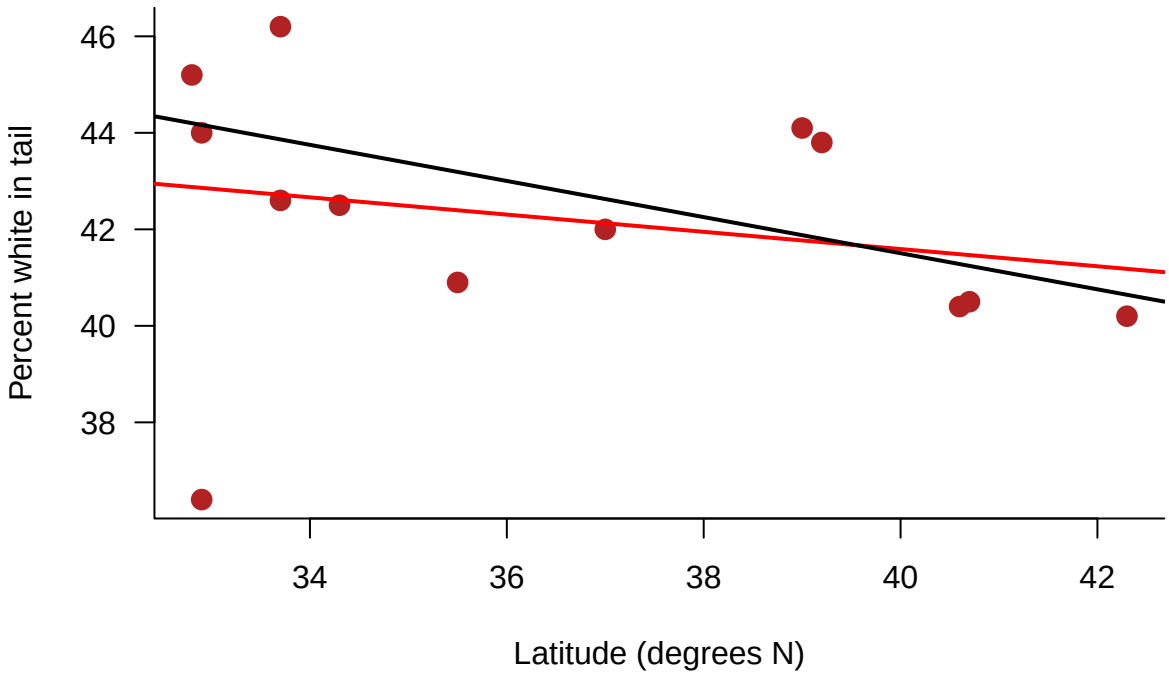
```
cooksd <- cooks.distance(juncoRegression)
plot(cooksd, pch = "*", cex = 2, main = "Influential Obs by Cooks distance")
abline(h = 4 * mean(cooksd, na.rm = T), col = "red")
text(x = 1:length(cooksd) + 1, y = cooksd, labels = ifelse(cooksd > 4 *
    mean(cooksd, na.rm = T), names(cooksd), ""), col = "red")
```



Knowing that one of the data points is influential, it could be removed.

Let's add regression lines to this scatter plot, calculated both with and without the outlier. You may need to redraw the scatter plot using the above commands before executing the following functions. The two lines below demonstrate the influence of including (red) and removing (black) the influential point:

```
plot(percentWhiteTail ~ latitude, data = junco, pch = 16, col = "firebrick",
    las = 1, cex = 1.5, bty = "l", xlab = "Latitude (degrees N)", ylab = "Percent white in tail")
juncoRegression <- lm(percentWhiteTail ~ latitude, data = junco)
abline(juncoRegression, col = "red", lwd = 2)
juncoRegressionNoOut <- lm(percentWhiteTail ~ latitude, data = junco, subset = percentWhiteTail >
    38)
abline(juncoRegressionNoOut, lwd = 2)
```



Plot residuals to check assumptions

- Whitlock and Schluter - 17.5-4

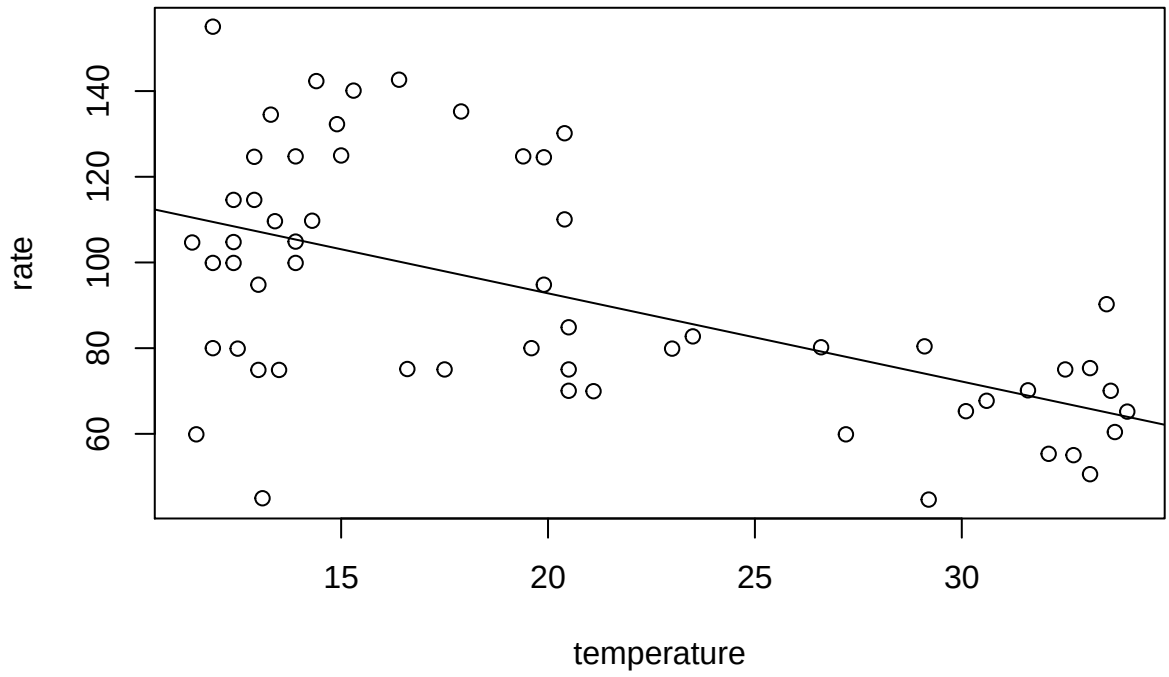
Neuron Firing Rate

Read and inspect the data. The data are temperature and the firing rate of neurons in cockroach.

```
roach <- read.csv("roach.csv")
```

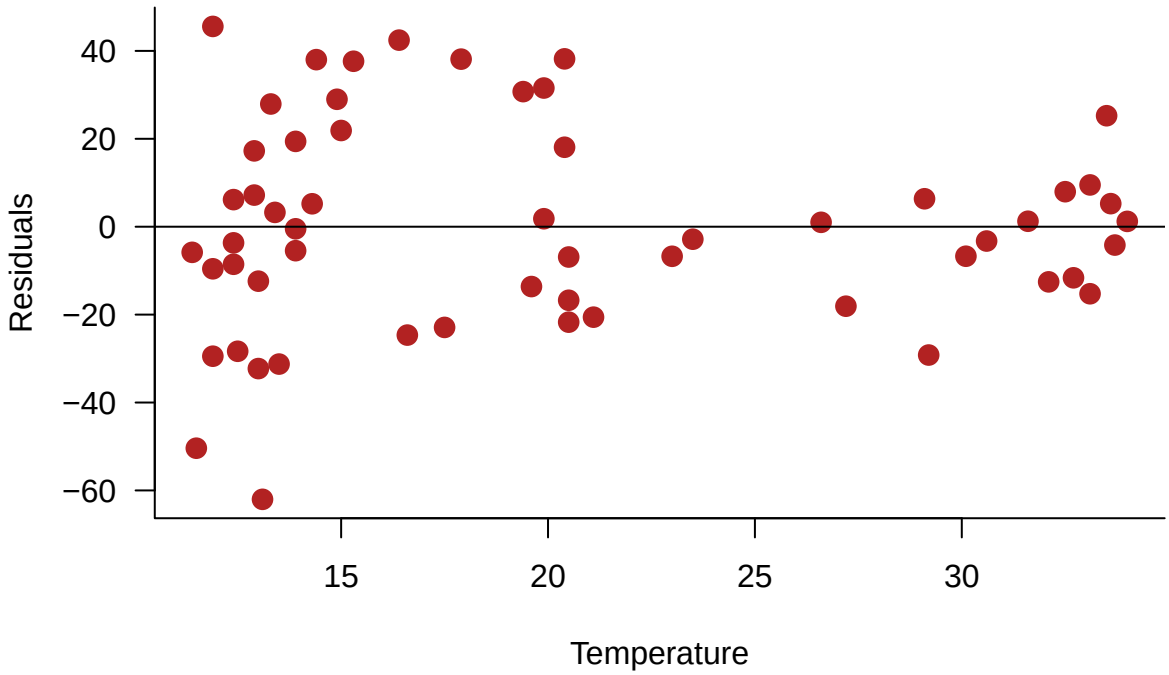
Scatter plot with regression line.

```
plot(rate ~ temperature, data = roach)
roachRegression <- lm(rate ~ temperature, data = roach)
abline(roachRegression)
```



Make a residual plot to check assumptions.

```
plot(residuals(roachRegression) ~ temperature, data = roach, pch = 16,
    col = "firebrick", las = 1, cex = 1.5, bty = "l", xlab = "Temperature",
    ylab = "Residuals")
abline(c(0, 0))
```



Transformation to improve regression fit

Iris data

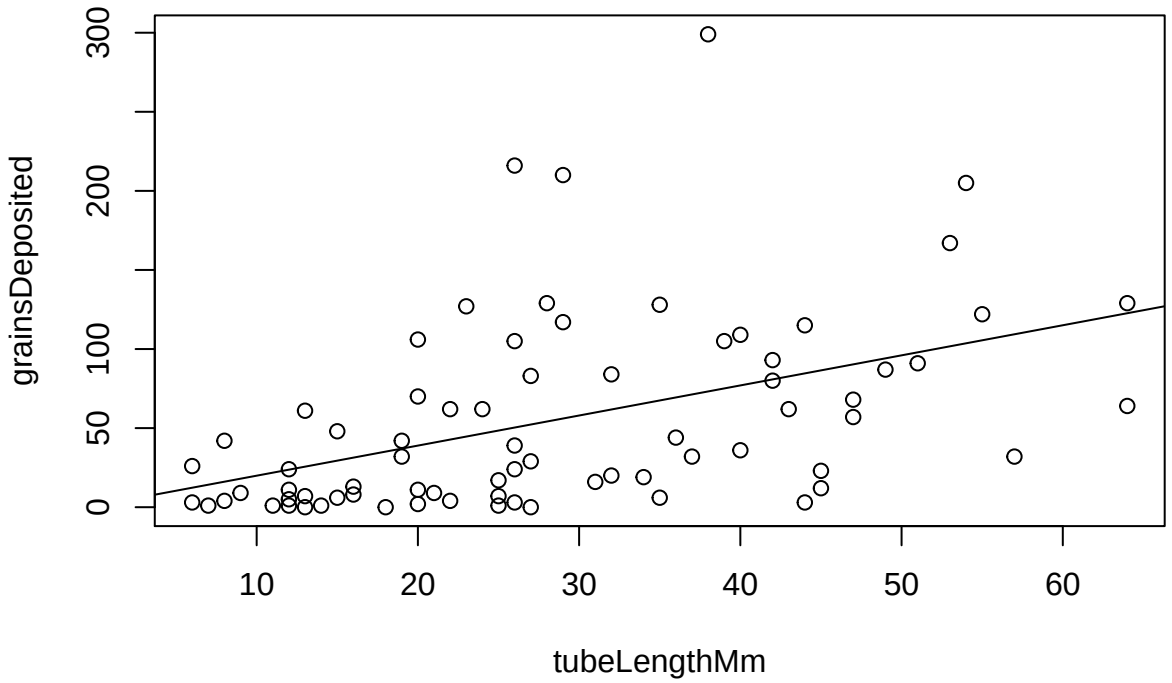
Read and inspect data.

```
iris <- read.csv("iris.csv")
str(iris)
```

```
## 'data.frame':    74 obs. of  3 variables:
##  $ X              : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ tubeLengthMm   : int  38 54 53 55 64 64 57 51 49 47 ...
##  $ grainsDeposited: int  299 205 167 122 129 64 32 91 87 68 ...
```

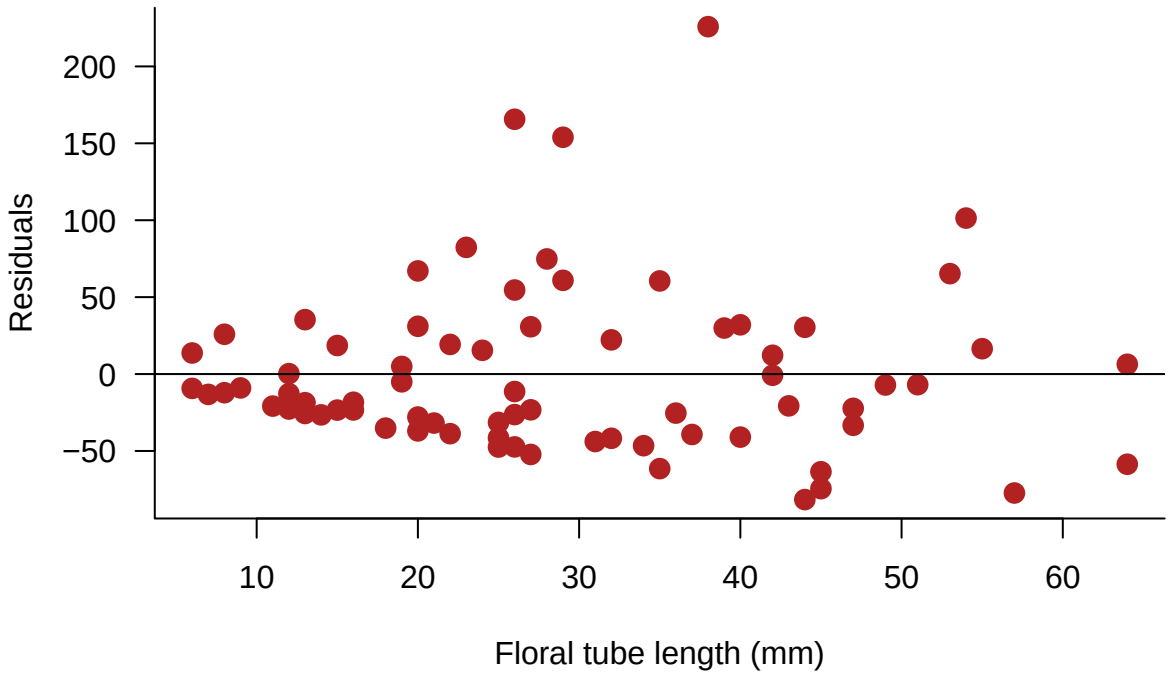
Scatter plot with regression line.

```
plot(grainsDeposited ~ tubeLengthMm, data = iris)
irisRegression <- lm(grainsDeposited ~ tubeLengthMm, data = iris)
abline(irisRegression)
```



Residual plot:

```
plot(residuals(irisRegression) ~ tubeLengthMm, data = iris, pch = 16, col = "firebrick",
    las = 1, cex = 1.5, bty = "l", xlab = "Floral tube length (mm)", ylab = "Residuals")
abline(c(0, 0))
```

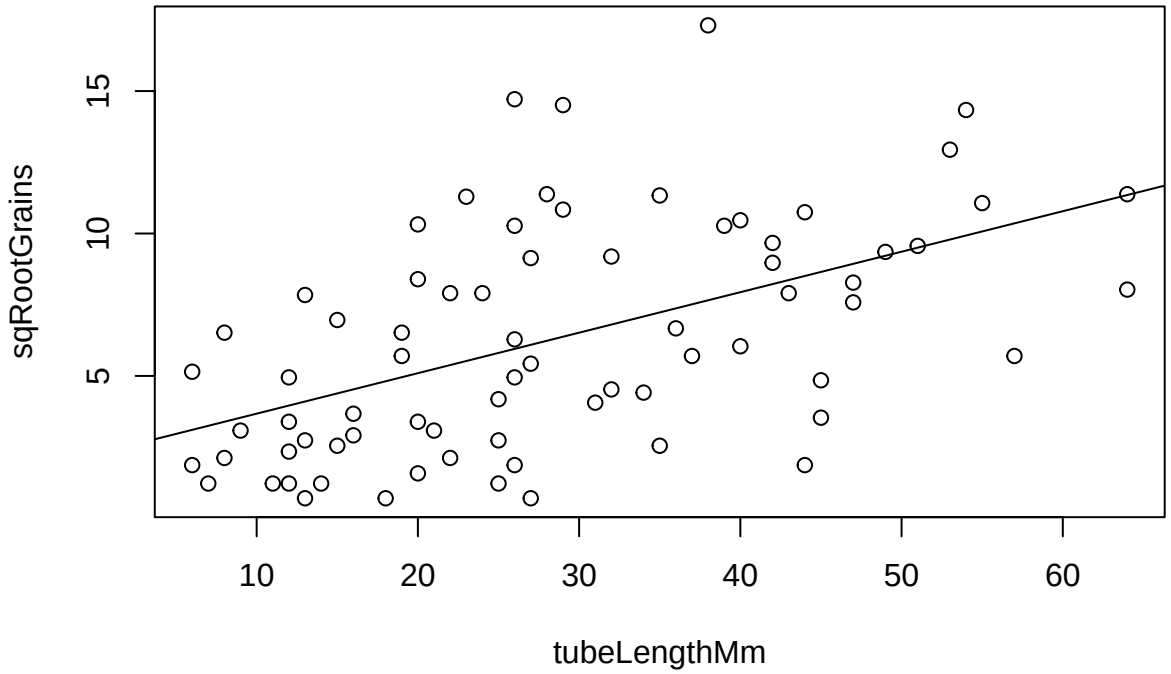


Use a square root transformation to improve the fit to assumptions of linear regression. The data are number of pollen grains received and floral tube length of an iris species.

```
iris$sqRootGrains <- sqrt(iris$grainsDeposited + 1/2)
```

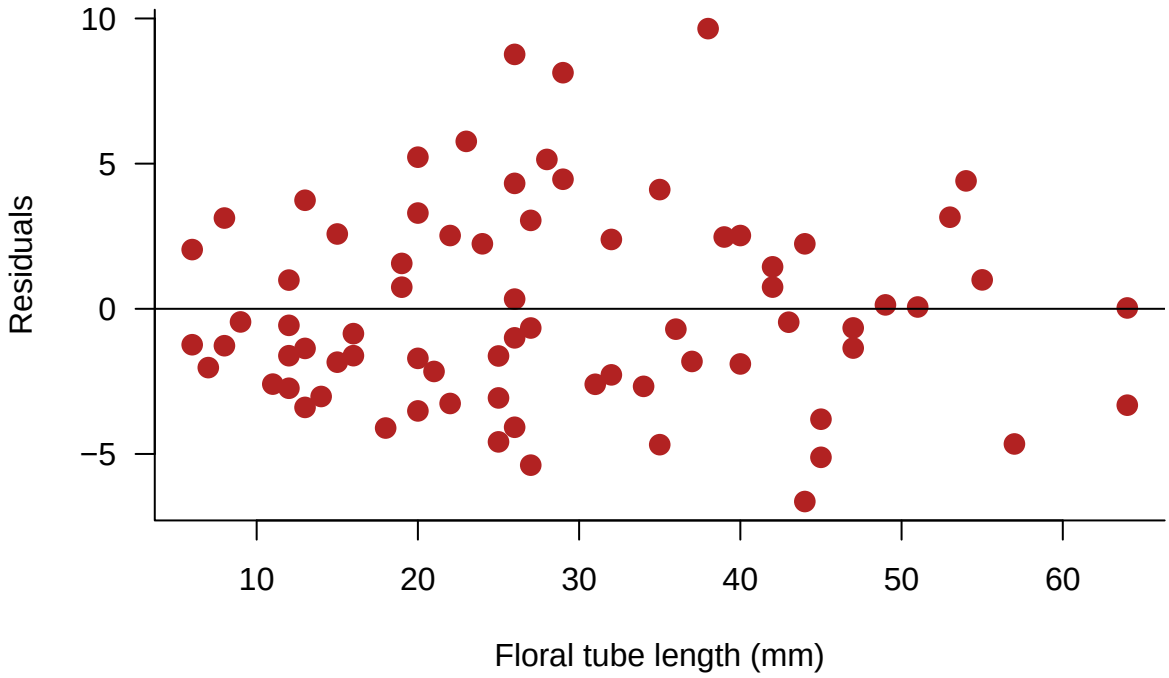
Scatter plot using transformed data, with new regression line added.

```
plot(sqRootGrains ~ tubeLengthMm, data = iris)
irisRegressionSqrt <- lm(sqRootGrains ~ tubeLengthMm, data = iris)
abline(irisRegressionSqrt)
```



Residual plot based on the transformed data.

```
plot(residuals(irisRegressionSqrt) ~ tubeLengthMm, data = iris, pch = 16,
    col = "firebrick", las = 1, cex = 1.5, bty = "l", xlab = "Floral tube length (mm)",
    ylab = "Residuals")
abline(c(0, 0))
```



ANCOVA is a Linear Model

- from Whitlock and Schluter. . .

Naked mole-rat energy expenditure

Analyze a factor while adjusting for a covariate, comparing energy expenditure of two castes of naked mole-rat while adjusting for differences in body mass using analysis of covariance ANCOVA.

Read and inspect the data.

```
d <- read.csv("molerat.csv")
head(d)
```

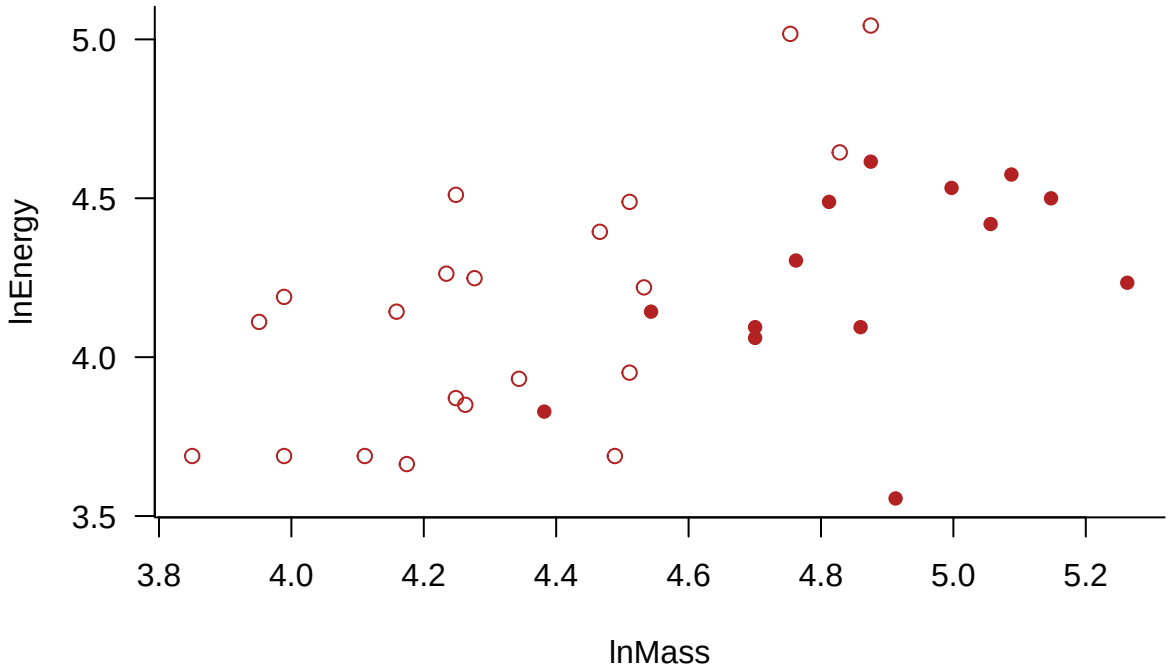
```
##   X  caste   lnMass lnEnergy
## 1 1 worker 3.850148 3.688879
## 2 2 worker 3.988984 3.688879
## 3 3 worker 4.110874 3.688879
## 4 4 worker 4.174387 3.663562
## 5 5 worker 4.248495 3.871201
## 6 6 worker 4.262680 3.850148
```

We are going to sort the data according to the value of the ln of body mass. This simplifies graphing of the model fits. The graph commands below assume that the data are sorted in this way.

```
moleRatSorted <- d[order(d$lnMass), ]
```

Scatter plot of the data.

```
plot(lnEnergy ~ lnMass, data = d, type = "n", las = 1, bty = "l")
points(lnEnergy ~ lnMass, data = subset(moleRatSorted, caste == "worker"),
    pch = 1, col = "firebrick")
points(lnEnergy ~ lnMass, data = subset(moleRatSorted, caste == "lazy"),
    pch = 16, col = "firebrick")
```



Fit models to the data, beginning with the model lacking an interaction term. Use lm because caste and mass are fixed effects. Save the predicted values in the data frame.

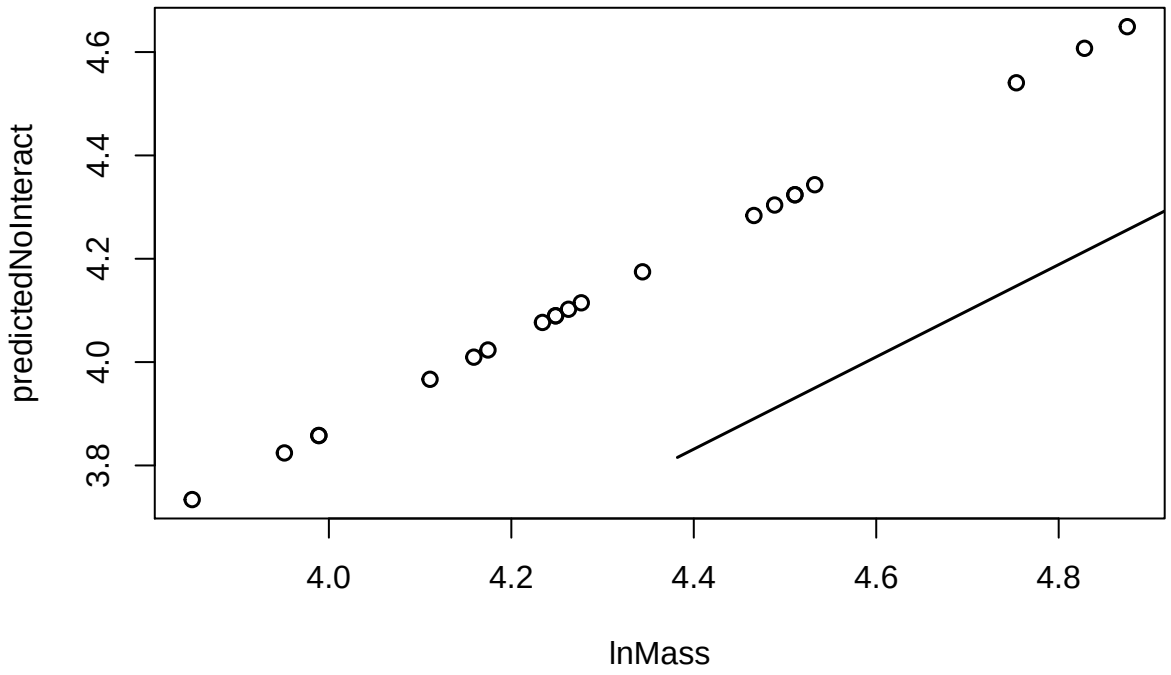
```
moleRatNoInteractModel <- lm(lnEnergy ~ lnMass + caste, data = moleRatSorted)
moleRatSorted$predictedNoInteract <- predict(moleRatNoInteractModel)
```

Fit the full model, which includes the interaction term. Again, save the predicted values in the data frame.

```
moleRatFullModel <- lm(lnEnergy ~ lnMass * caste, data = moleRatSorted)
moleRatSorted$predictedInteract <- predict(moleRatFullModel)
```

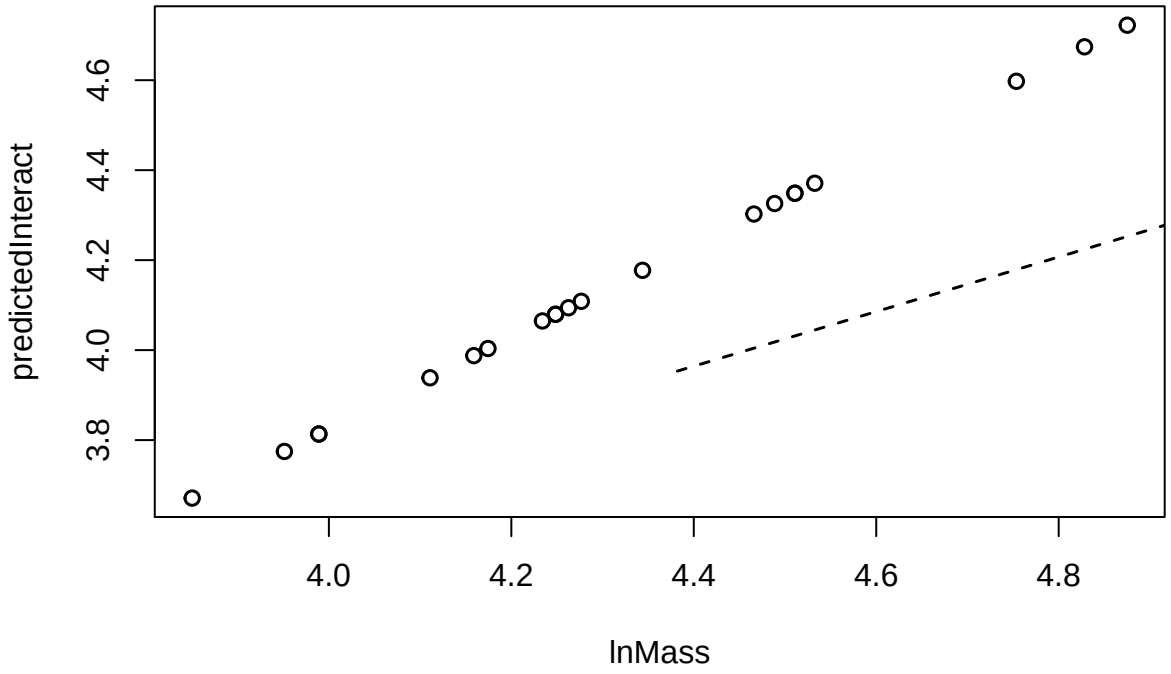
Visualize the model fits, beginning with the fit of the no-interaction model. Redraw the scatter plot (see commands above), if necessary, before issuing the following commands.

```
plot(predictedNoInteract ~ lnMass, data = subset(moleRatSorted, caste ==
    "worker"), lwd = 1.5)
lines(predictedNoInteract ~ lnMass, data = subset(moleRatSorted, caste ==
    "lazy"), lwd = 1.5)
```



Visualize the fit of the full model, including the interaction term. Redraw the scatter plot, if necessary, before issuing the following commands.

```
plot(predictedInteract ~ lnMass, data = subset(moleRatSorted, caste ==
    "worker"), lwd = 1.5, lty = 2)
lines(predictedInteract ~ lnMass, data = subset(moleRatSorted, caste ==
    "lazy"), lwd = 1.5, lty = 2)
```



Test the improvement in fit of the full model, including the interaction term. This is a test of the interaction term only.

```
anova(moleRatNoInteractModel, moleRatFullModel)
```

```
## Analysis of Variance Table
##
## Model 1: lnEnergy ~ lnMass + caste
## Model 2: lnEnergy ~ lnMass * caste
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     32 2.8145
## 2     31 2.7249  1  0.089557 1.0188 0.3206
```

Test for differences in ln body mass between castes, assuming that no interaction term is present in the mole rat population (i.e., assuming that the two castes hve equal slopes). Most commonly this is done using either "Type III" sums of squares (see footnote 5 on p 618 of the book) or "Type I" sums of squares (the default in R). The two methods do not give identical answers here because the design is not balanced (in a balanced design, each value of the x-variable would have the same number of y-observations from each group).

Test using "Type III" sums of squares. We need to include a contrasts argument for the categorical variable in the lm command. Then we need to load the car package and use its Anova command. Note that "A" is in upper case in Anova().

```
moleRatNoInteractModelTypeIII <- lm(lnEnergy ~ lnMass + caste, data = d,
    contrasts = list(caste = contr.sum))
library(car)
```

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

```
Anova(moleRatNoInteractModelTypeIII, type = "III")
```

```
## Anova Table (Type III tests)
##
## Response: lnEnergy
##              Sum Sq Df F value    Pr(>F)
## (Intercept) 0.00111  1  0.0126    0.9112
## lnMass      1.88152  1 21.3923 5.887e-05 ***
## caste       0.63747  1  7.2478    0.0112 *
## Residuals   2.81450 32
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# note 'A' in Anova is capitalized
```

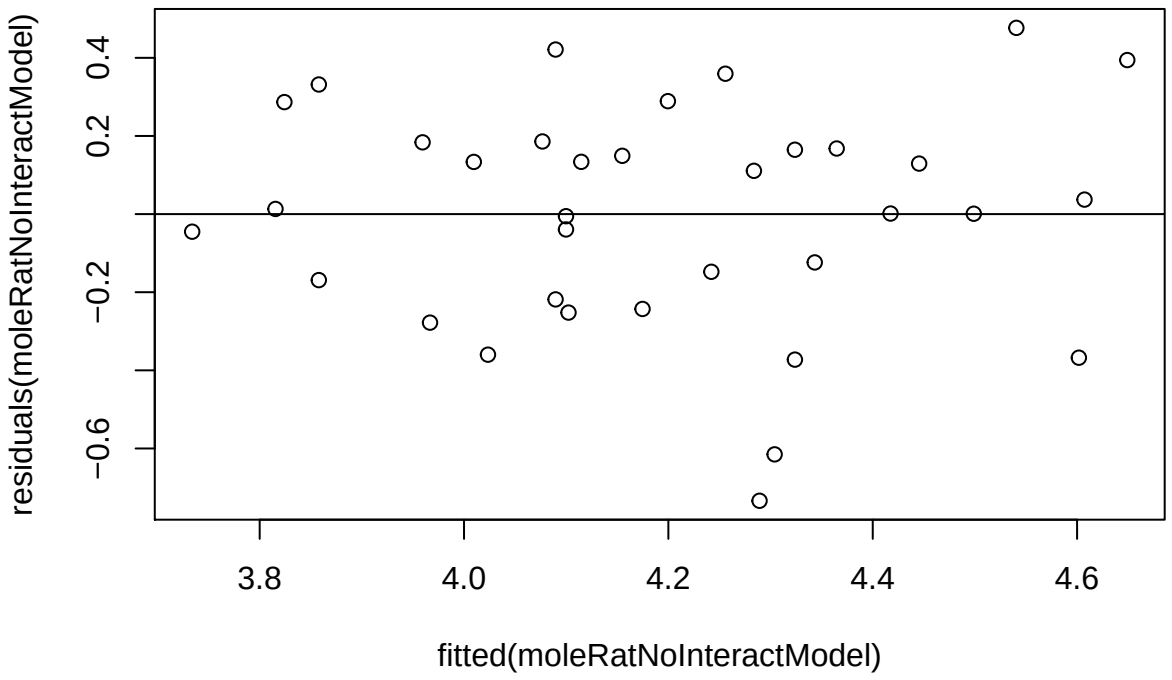
Test using "Type I" (sequential) sums of squares. Make sure that the covariate (lnMass) comes before the factor (caste) in the lm formula, as shown. Note that "a" is in lower case in anova.

```
moleRatNoInteractModel <- lm(lnEnergy ~ lnMass + caste, data = d)
anova(moleRatNoInteractModel)
```

```
## Analysis of Variance Table
##
## Response: lnEnergy
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## lnMass     1 1.31061 1.31061 14.9013 0.0005178 ***
## caste      1 0.63747 0.63747  7.2478 0.0111984 *
## Residuals 32 2.81450 0.08795
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

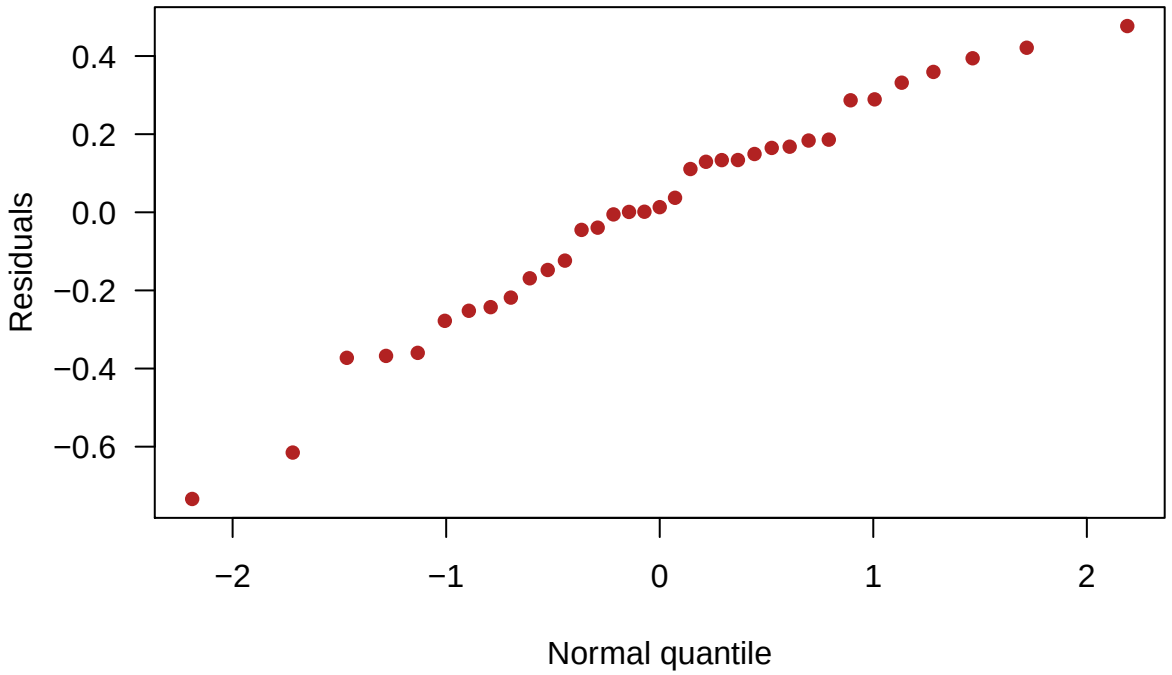
Residual plot from the linear model.

```
plot(residuals(moleRatNoInteractModel) ~ fitted(moleRatNoInteractModel))
abline(0, 0)
```



Normal quantile plot of residuals.

```
qqnorm(residuals(moleRatNoInteractModel), pch = 16, col = "firebrick",
    las = 1, ylab = "Residuals", xlab = "Normal quantile", main = "")
```



Multivariate - Polynomial Regression

Pond Productivity

- Whitlock and Schluter - 17.8-2

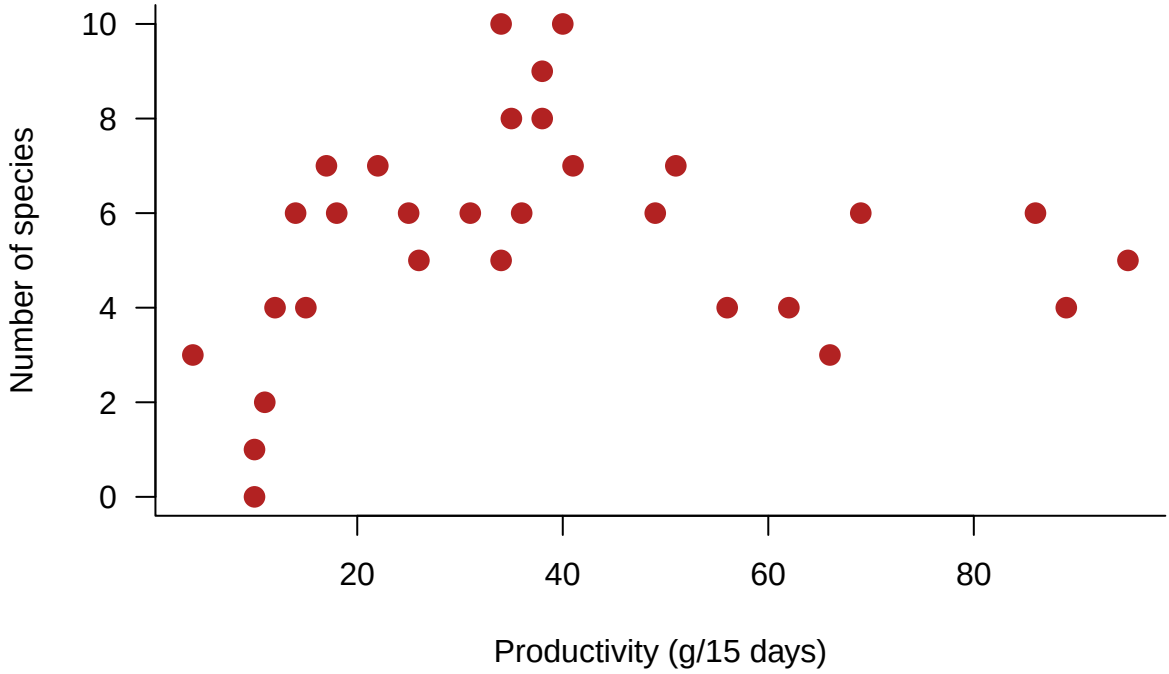
Fit a quadratic curve to the relationship between the number of plant species present in ponds and pond productivity.

Read and examine data.

```
d <- read.csv("pondproductivity.csv")
```

Scatter plot.

```
plot(species ~ productivity, data = d, pch = 16, col = "firebrick", las = 1,
    cex = 1.5, bty = "l", ylab = "Number of species", xlab = "Productivity (g/15 days)")
```



Fit a quadratic curve to the data. Here, the single variable productivity in the data frame is included in the formula both as itself and as the squared term, productivity2. To make the squared term work, we need to wrap the term with I(). The results of the model fit are saved in an R object productivityCurve.

```
productivityCurve <- lm(species ~ productivity + I(productivity^2), data = d)
```

Show estimates of the parameters of the quadratic curve (regression coefficients) are obtained as follows, along with standard errors and t-tests.

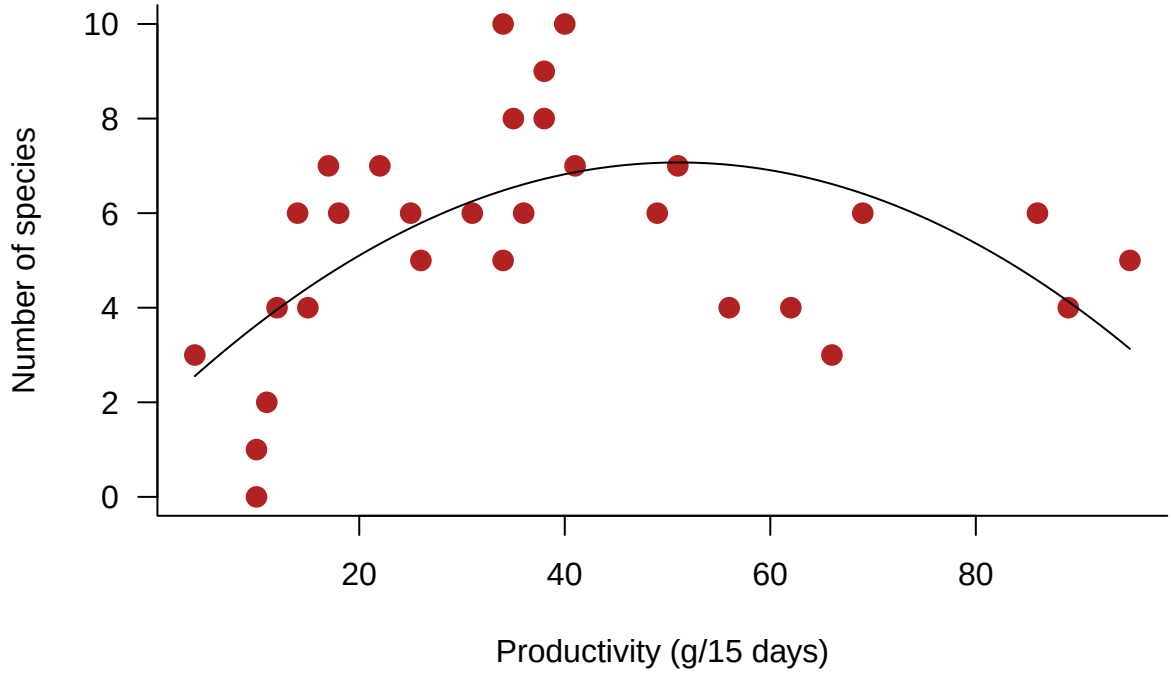
```
summary(productivityCurve)
```

```
##
## Call:
## lm(formula = species ~ productivity + I(productivity^2), data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.6330 -0.9952 -0.0158  1.4458  3.5215
```

```
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         1.7535477  1.1118289   1.577 0.126401
## productivity        0.2083549  0.0558421   3.731 0.000897 ***
## I(productivity^2) -0.0020407  0.0005677  -3.595 0.001280 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.999 on 27 degrees of freedom
## Multiple R-squared:  0.3402, Adjusted R-squared:  0.2913
## F-statistic: 6.961 on 2 and 27 DF,  p-value: 0.003648
```

Add quadratic regression curve to scatter plot. If necessary, redraw the previous scatter plot before issuing the following commands.

```
xpts <- seq(min(d$productivity), max(d$productivity), length.out = 100)
ypts <- predict(productivityCurve, new = data.frame(productivity = xpts))
plot(species ~ productivity, data = d, pch = 16, col = "firebrick", las = 1,
    cex = 1.5, bty = "l", ylab = "Number of species", xlab = "Productivity (g/15 days)")
lines(xpts, ypts)
```



Modeling and testing model fits

Prairie plant biomass

- Whitlock and Schluter fig 18.1-1

Compare the fits of the null and univariate regression models to data on the relationship between stability of plant biomass production and the initial number of plant species assigned to plots. The data are from Example 17.3.

Read and inspect data.

```
prairie <- read.csv("prairie.csv")
```

Take the log-transformation of stability.

```
prairie$logStability <- log(prairie$biomassStability)
head(prairie)
```

```
##   X nSpecies biomassStability logStability
## 1 1        1             7.47     2.010895
## 2 2        1             6.74     1.908060
## 3 3        1             6.61     1.888584
## 4 4        1             6.40     1.856298
## 5 5        1             5.67     1.735189
## 6 6        1             5.26     1.660131
```

Fit the null model to the data, which in simple linear regression is a line of 0 slope.

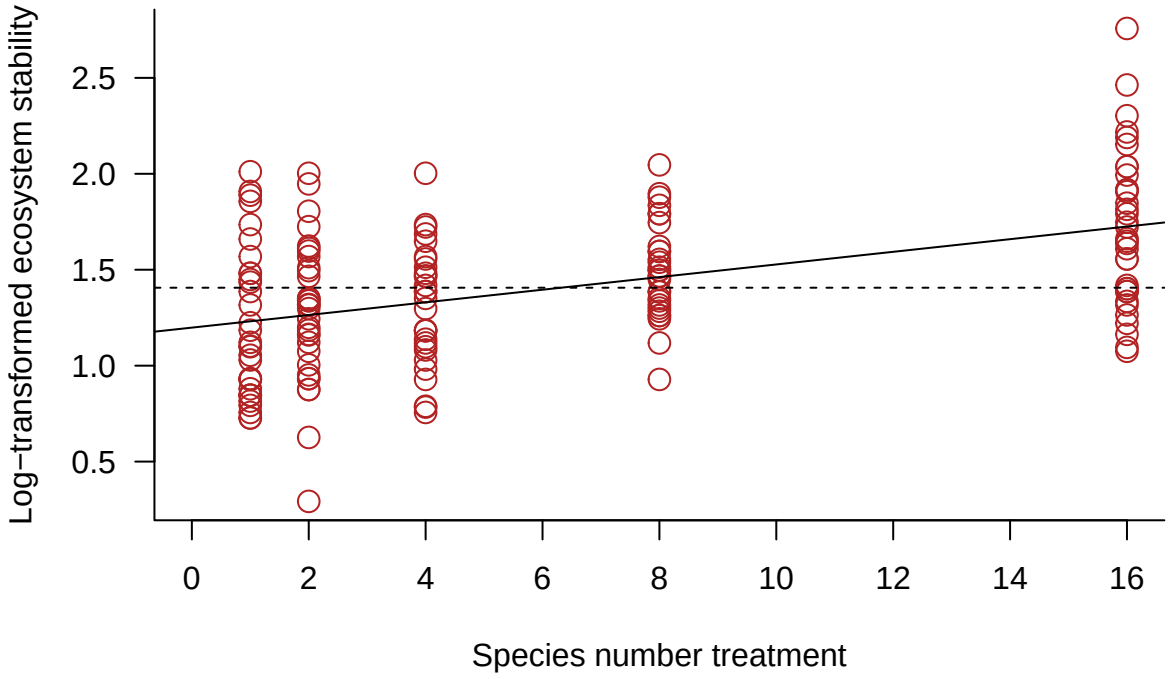
```
prairieNullModel <- lm(logStability ~ 1, data = prairie)
```

Fit the full model, which includes the treatment variable.

```
prairieRegression <- lm(logStability ~ nSpecies, data = prairie)
```

Scatter plot to compare models visually.

```
plot(logStability ~ nSpecies, data = prairie, bty = "l", col = "firebrick",
    pch = 1, las = 1, cex = 1.5, xlim = c(0, 16), xaxp = c(0, 16, 8), xlab = "Species number treatment"
    ylab = "Log-transformed ecosystem stability")
abline(prairieNullModel, lty = 2)
abline(prairieRegression)
```



The F-test of improvement in fit of the full (regression) model.

```
anova(prairieRegression)
```

```
## Analysis of Variance Table
##
## Response: logStability
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## nSpecies    1  5.5169  5.5169  45.455 2.733e-10 ***
## Residuals 159 19.2980  0.1214
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ALL EXAMPLES NEED WORK - STILL IN PROGRESS