STATS 415 Lab 2

Nick Seewald

19 January 2018

1 Today's Objectives

1. Learn how to perform (simple and multiple) linear regression in R
2. Practice reading R output for linear regression to identify key results
3. See how R works with categorical variables and interactions

2 Exploratory Data Analysis

We'll use the `Boston` dataset available in the `MASS` package. This dataset contains information on housing values for 506 neighborhoods near Boston. Recall from Lab 1 how to load packages and access datasets from them. First install the MASS package.

```
install.packages("MASS")
```

Now, we'll load the package and the dataset.

```
library(MASS)
data(Boston)
```

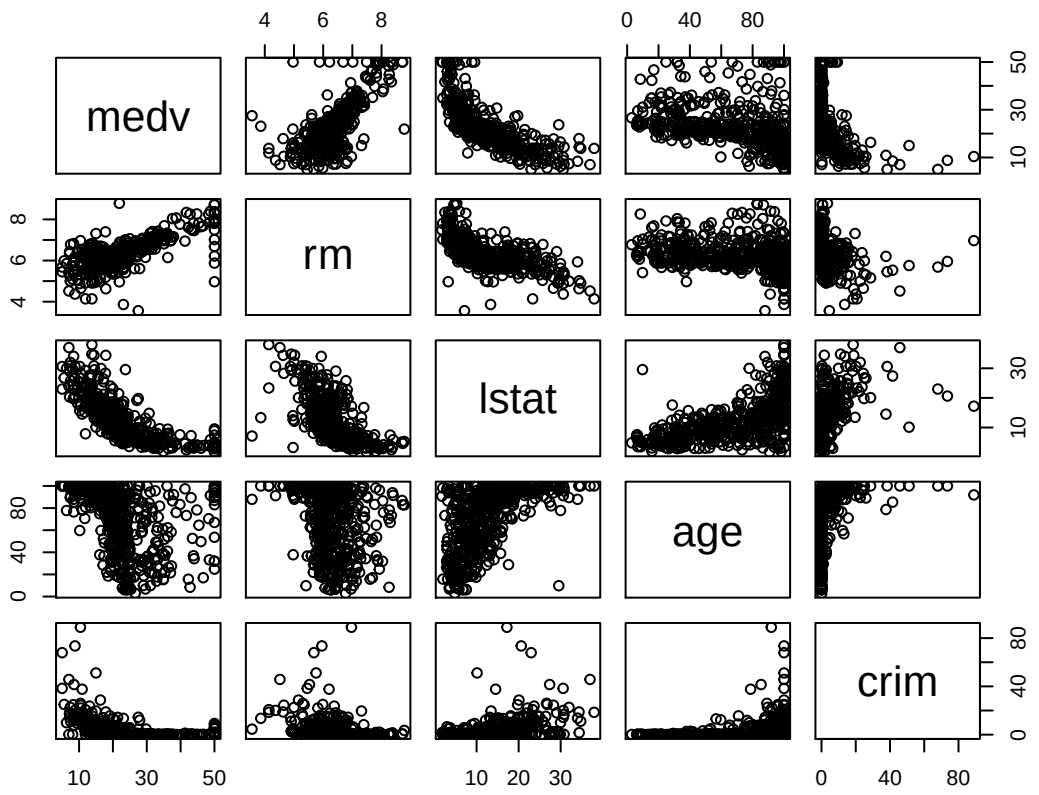
Since this dataset is from a package, we can use `?Boston` or `help(Boston)` to get information about the variables in the dataset.

Question: What is one categorical variable in the `Boston` data?

Question: What is the `rm` variable?

Let's look at a scatterplot matrix of a few continuous variables in `Boston`.

```
pairs(subset(Boston, select = c("medv", "rm", "lstat", "age", "crim")),
      main = "Scatterplot matrix for variables in Boston dataset")
```



Question: What can we say about the relationship between `medv` and `lstat`? Do you expect `lstat` to be a good predictor of `medv`?

3 Simple Linear Regression

We'll start by fitting a simple linear regression model to `medv` using `lstat` as a predictor. To do this in R, we use the `lm` function:

```
mod1 <- lm(medv ~ lstat, data = Boston)
```

Let's break this code down a bit:

- We're storing the results of the regression as an object called `mod1`.
- The first argument to `lm` is a "formula" in R. The basic syntax is `y ~ x`, where `y` is the response and `x` is a predictor. The formula `medv ~ lstat` tells R to fit the model

$$\texttt{medv}_i = \beta_0 + \beta_1 \texttt{lstat}_i + \epsilon_i. \tag{1}$$

- The `data` argument tells `lm` where to look for the variables you used in the formula.

To get the results of the regression, we can just print `mod1`, or, better yet, use `summary`:

```
mod1
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Coefficients:
## (Intercept)        lstat
##       34.55        -0.95
```

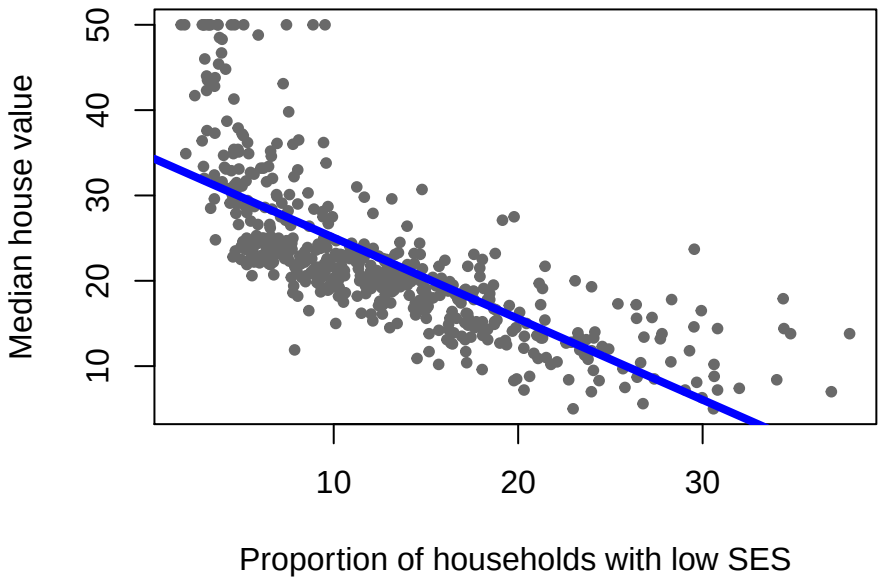
```
summary(mod1)
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41   <2e-16 ***
## lstat       -0.95005    0.03873  -24.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

Printing `mod1` shows the estimates for the βs obtained by solving the OLS problem (see optional slides from lecture), whereas `summary` gives us more information, including p-values and standard errors for the βs, as well as the R^2 statistic for the model.

What does the fitted line look like relative to the data?

```
plot(Boston$medv ~ Boston$lstat,
     xlab = "Proportion of households with low SES",
     ylab = "Median house value",
     pch = 20, col = "dimgray")
abline(a = mod1$coefficients[1], b = mod1$coefficients[2],
       col = "blue", lwd = 4)
```



Note that "SES" stands for "socio-economic status"

Question: Is `lstat` a useful predictor of `medv`? How do we tell?

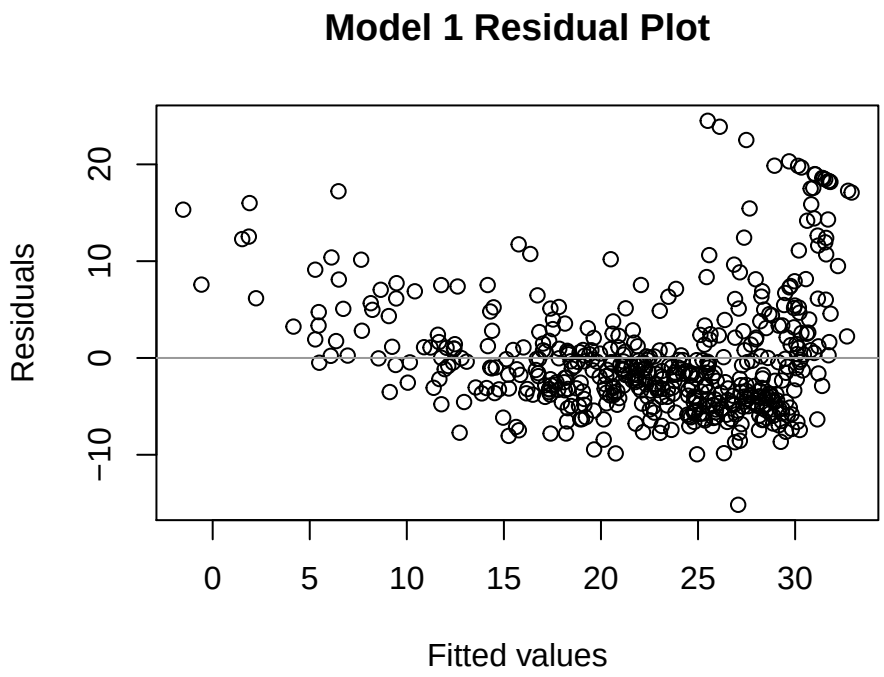
- **Hint:** Recall that we use a hypothesis testing framework to address this question: "Assume `lstat` is not useful ($\beta_1 = 0$) and see if there is enough evidence to reject this null hypothesis" (Lecture "Model Accuracy 1", slide 26)

3.1 Residual plots

A key assumption in linear regression is that the errors (ϵ_i) are independent and all have mean zero and variance σ^2. We can check this assumption by looking at a plot of the residuals vs. the fitted values. *Residuals* are estimates of the errors: $\hat{\epsilon}_i = y_i - \hat{y}_i$.

Question: If the errors are actually mean-zero and have constant variance, what kind of pattern would we expect in this plot?

```
plot(mod1$residuals ~ mod1$fitted.values, main = "Model 1 Residual Plot",
     xlab = "Fitted values", ylab = "Residuals")
abline(a = 0, b = 0, col = "gray60")
```



Question: Does this residual plot suggest any assumption violations?

4 Multiple Linear Regression

We can see from the residual plot above that maybe a quadratic term of `lstat` might be useful to include in the model. Let's add that squared term. This is now a *multiple* linear regression since it includes multiple predictors.

```
mod1sq <- lm(medv ~ lstat + I(lstat^2), data = Boston)
summary(mod1sq)
```

```
##
## Call:
## lm(formula = medv ~ lstat + I(lstat^2), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.2834  -3.8313  -0.5295   2.3095  25.4148
##
```

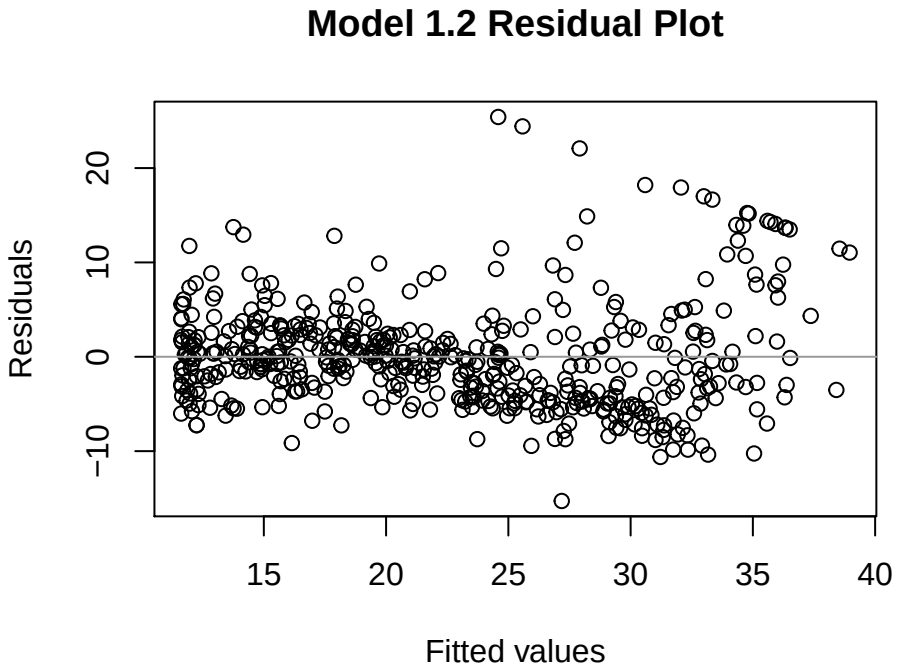
```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.862007   0.872084   49.15   <2e-16 ***
## lstat       -2.332821   0.123803  -18.84   <2e-16 ***
## I(lstat^2)   0.043547   0.003745   11.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.524 on 503 degrees of freedom
## Multiple R-squared:  0.6407, Adjusted R-squared:  0.6393
## F-statistic: 448.5 on 2 and 503 DF,  p-value: < 2.2e-16
```

Some notes on the code:

- When we add additional predictors to the model formula, we combine them with `+`
- In order to add $\texttt{lstat}^2$ to the model we either have to create a new variable in the dataset *or*, if we want to do math inside `lm`, wrap it in `I()`.

Now let's look at the new residual plot.

```
plot(mod1sq$residuals ~ mod1sq$fitted.values, main = "Model 1.2 Residual Plot",
     xlab = "Fitted values", ylab = "Residuals")
abline(a = 0, b = 0, col = "gray60")
```



Let's now fit another multiple linear regression, this time adding `age` and `rm` to model 1. We still use `lm`, but this time we add more variables to our formula argument:

```
mod2 <- lm(medv ~ lstat + age + rm, data = Boston)
summary(mod2)
```

```
##
## Call:
```

```
## lm(formula = medv ~ lstat + age + rm, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.210  -3.467  -1.053   1.957  27.500
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.175311   3.181924  -0.369    0.712
## lstat       -0.668513   0.054357 -12.298   <2e-16 ***
## age          0.009091   0.011215   0.811    0.418
## rm           5.019133   0.454306  11.048   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.542 on 502 degrees of freedom
## Multiple R-squared:  0.639,  Adjusted R-squared:  0.6369
## F-statistic: 296.2 on 3 and 502 DF,  p-value: < 2.2e-16
```

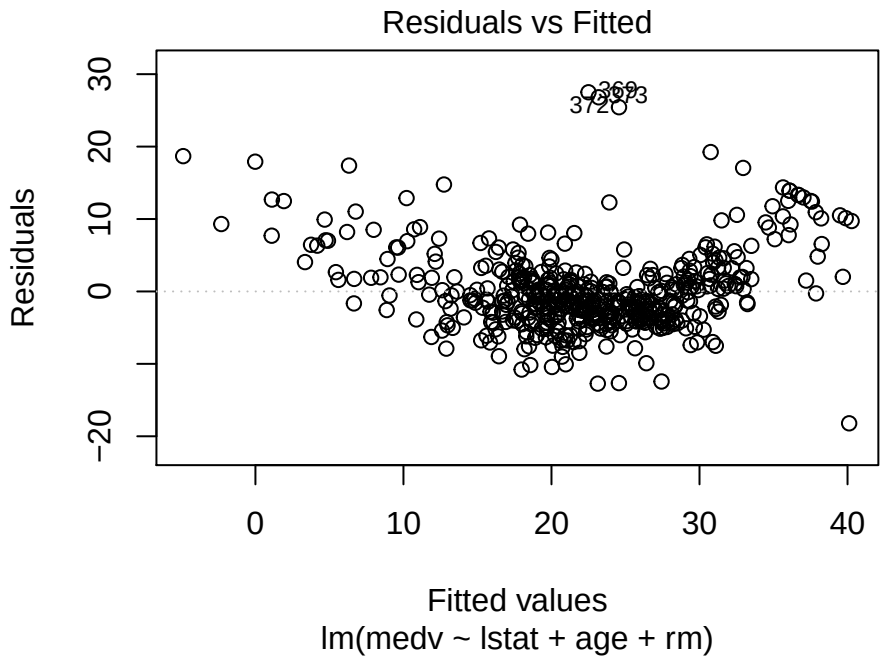
Question: What is the model this code fits?

Question: What is the estimate of the coefficient for `age`?

Question: What is the null hypothesis corresponding to the p-value for `rm`?

Let's check the residual plot for the model. We can also construct these plots using the `plot` function on an `lm` object.

```
plot(mod2, which = 1, add.smooth = F)
```



Notes:

- `which = 1` tells R to plot only the residual plot (omitting this will give you more plots that are useful, but not entirely in the scope of this course)
- `add.smooth = F` tells R not to add a "smoother" to the plot. We don't want this because we don't know what it does yet! We only want to plot things we understand.

Question: Does this plot indicate any assumption violations?

In `mod2`, how do we test the null hypothesis $H_0 : \beta_{\texttt{age}} = \beta_{\texttt{rm}} = 0$? Under this null hypothesis, the model would be $y_i = \beta_0 + \beta_{\texttt{lstat}}\texttt{lstat}_i + \epsilon_i$. We call this the *reduced* or *null* model. Notice that the null model is just `mod1` from above! We want to compare it to `mod2` using ANOVA to test the null hypothesis:

```
anova(mod1, mod2)
```

```
## Analysis of Variance Table
##
## Model 1: medv ~ lstat
## Model 2: medv ~ lstat + age + rm
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1    504 19472
## 2    502 15419  2    4053.3 65.981 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The `anova` function generates an ANOVA table like in the notes that compares the two models. The first argument is the null model, the second is the *full* model.

What if we wanted to test the null hypothesis $H_0: \beta_{\texttt{lstat}} = \beta_{\texttt{age}} = \beta_{\texttt{rm}} = 0$?

Question: What is the null model for this test?

We build this null, intercept-only model by using `1` as the only predictor in the model

```
nullModel <- lm(medv ~ 1, data = Boston)
anova(nullModel, mod2)
```

```
## Analysis of Variance Table
##
## Model 1: medv ~ 1
## Model 2: medv ~ lstat + age + rm
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1    505 42716
## 2    502 15419  3     27297 296.24 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Question: What do we conclude from the above about our null hypothesis $H_0: \beta_{\texttt{lstat}} = \beta_{\texttt{age}} = \beta_{\texttt{rm}} = 0$?

5 Categorical predictors

5.1 2-level categorical predictors

We saw from above that the `chas` variable is categorical: it has value 1 if the tract bounds the Charles River and 0 otherwise. Now let's include it as a predictor in a multiple linear regression with `medv` as the response.

```
mod3 <- lm(medv ~ lstat + rm + chas, data = Boston)
summary(mod3)
```

```
##
## Call:
## lm(formula = medv ~ lstat + rm + chas, data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21.5682  -3.3475  -0.9197   2.0177  28.2290
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.76368    3.12212  -0.245    0.807
## lstat       -0.64285    0.04299 -14.953  < 2e-16 ***
```

```
## rm           4.95581    0.43813  11.311  < 2e-16 ***
## chas         4.12048    0.95820   4.300 2.05e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.446 on 502 degrees of freedom
## Multiple R-squared:  0.6514, Adjusted R-squared:  0.6493
## F-statistic: 312.7 on 3 and 502 DF,  p-value: < 2.2e-16
```

Question: Because `chas` is "dummy coded", how do we interpret its parameter estimate?

5.2 Categorical predictors with more than two levels

For this example, we'll use the Iris dataset that's been previously discussed in class.

```
data(iris)
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
```

`Species` is a categorical variable with three levels, and it's already a factor, so no need to convert.

```
is(iris$Species)
```

```
## [1] "factor"              "integer"             "oldClass"
## [4] "numeric"             "vector"              "data.frameRowLabels"
```

```
table(iris$Species)
```

```
##
##     setosa versicolor  virginica
##         50         50         50
```

Let's run a regression with `Sepal.Length` as the response and include all the predictors in the model *except* for `Species`.

```
irismod1 <- lm(Sepal.Length ~ . - Species, data = iris)
summary(irismod1)
```

```
##
## Call:
## lm(formula = Sepal.Length ~ . - Species, data = iris)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.82816 -0.21989  0.01875  0.19709  0.84570
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.85600    0.25078   7.401 9.85e-12 ***
## Sepal.Width   0.65084    0.06665   9.765  < 2e-16 ***
## Petal.Length  0.70913    0.05672  12.502  < 2e-16 ***
```

```
## Petal.Width  -0.55648    0.12755  -4.363 2.41e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3145 on 146 degrees of freedom
## Multiple R-squared:  0.8586, Adjusted R-squared:  0.8557
## F-statistic: 295.5 on 3 and 146 DF,  p-value: < 2.2e-16
```

- Note the syntax in the formula! `y ~ .` means "use all variables in `data` that aren't `y`". If we want to plot all variables except for `x1`, say, we use *"subtraction"* (vs. "addition"): `y ~ . - x1`

Now let's add `Species` in to see how R deals with multi-level factors.

```
irismod2 <- lm(Sepal.Length ~ ., data = iris)
summary(irismod2)
```

```
##
## Call:
## lm(formula = Sepal.Length ~ ., data = iris)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.79424 -0.21874  0.00899  0.20255  0.73103
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        2.17127    0.27979   7.760 1.43e-12 ***
## Sepal.Width        0.49589    0.08607   5.761 4.87e-08 ***
## Petal.Length       0.82924    0.06853  12.101  < 2e-16 ***
## Petal.Width       -0.31516    0.15120  -2.084  0.03889 *
## Speciesversicolor -0.72356    0.24017  -3.013  0.00306 **
## Speciesvirginica  -1.02350    0.33373  -3.067  0.00258 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3068 on 144 degrees of freedom
## Multiple R-squared:  0.8673, Adjusted R-squared:  0.8627
## F-statistic: 188.3 on 5 and 144 DF,  p-value: < 2.2e-16
```

- Note that the output contains some strange variable names: `Speciesversicolor` and `Speciesvirginica`. These are dummy variables for `Species == 'versicolor'` and `Species == 'virginica'`, respectively.

Question: What about `Speciessetosa`? Why is that variable not in the output?

Question: What is the interpretation of the estimate of `Species virginica`?

What if we want to change the baseline variable? We need to reorder the levels of `Species`. `lm` *always* uses the first level of a factor as the baseline.

```
levels(iris$Species)
```

```
## [1] "setosa"     "versicolor" "virginica"
```

```
iris$Species <- factor(iris$Species, levels = c("versicolor", "virginica", "setosa"))
levels(iris$Species)
```

```
## [1] "versicolor" "virginica"  "setosa"
```

- Note that we use the `factor` function and *not* `as.factor`! We want to *create* the factor, rather than just convert something existing to factor. `as.factor` can only convert, not specify levels.
- The `levels` argument is a vector of the values we observe in `iris$Species` ordered in the way we want them.

Now let's re-run the model.

```
irismod3 <- lm(Sepal.Length ~ ., data = iris)
summary(irismod3)
```

```
##
## Call:
## lm(formula = Sepal.Length ~ ., data = iris)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.79424 -0.21874  0.00899  0.20255  0.73103
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        1.44770    0.28149   5.143 8.68e-07 ***
## Sepal.Width        0.49589    0.08607   5.761 4.87e-08 ***
## Petal.Length       0.82924    0.06853  12.101  < 2e-16 ***
## Petal.Width       -0.31516    0.15120  -2.084  0.03889 *
## Speciesvirginica -0.29994    0.11898  -2.521  0.01280 *
## Speciessetosa      0.72356    0.24017   3.013  0.00306 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3068 on 144 degrees of freedom
## Multiple R-squared:  0.8673, Adjusted R-squared:  0.8627
## F-statistic: 188.3 on 5 and 144 DF,  p-value: < 2.2e-16
```

Alternatively, we could have made dummy-coded variables.

```
iris$virginica  <- as.numeric(iris$Species == 'virginica')
iris$versicolor <- as.numeric(iris$Species == 'versicolor')
iris$setosa     <- as.numeric(iris$Species == 'setosa')
```

- The `==` tells R this is *logic* rather than assignment. It's checking to see if `iris$Species` is equal to the string in quotes.
- The logical statments (e.g., `iris$Species == 'virginica'`) evaluate to `TRUE` or `FALSE`. We use `as.numeric` to convert `TRUE` to 1 and `FALSE` to 0.
- **DO NOT** include all of the dummy variables in the model! This will cause `lm` to fail. You *must* have a baseline!

Note that the output of the linear model using the dummy coded variables `virginica` and `versicolor` is identical to the results of `irismod3`.

```r
irismod4 <- lm(Sepal.Length ~ . - Species - setosa, data = iris)
summary(irismod4)
```

```
##
## Call:
## lm(formula = Sepal.Length ~ . - Species - setosa, data = iris)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.79424 -0.21874  0.00899  0.20255  0.73103
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.17127    0.27979   7.760 1.43e-12 ***
## Sepal.Width   0.49589    0.08607   5.761 4.87e-08 ***
## Petal.Length  0.82924    0.06853  12.101  < 2e-16 ***
## Petal.Width  -0.31516    0.15120  -2.084  0.03889 *
## virginica    -1.02350    0.33373  -3.067  0.00258 **
## versicolor   -0.72356    0.24017  -3.013  0.00306 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3068 on 144 degrees of freedom
## Multiple R-squared:  0.8673, Adjusted R-squared:  0.8627
## F-statistic: 188.3 on 5 and 144 DF,  p-value: < 2.2e-16
```

6 Interactions

We'll continue using the iris data, and start by re-loading it so we have a clean copy. We can investigate whether the effect of `Petal.Width` varies by `Species` by adding an interaction between those two terms to the model. To do this, we use colons: the interaction term between `Petal.Width` and `Species` is coded as `Petal.Width:Species`:

```r
data(iris)
irismod5 <- lm(Sepal.Length ~ . + Petal.Width:Species, data = iris)
summary(irismod5)
```

```
##
## Call:
## lm(formula = Sepal.Length ~ . + Petal.Width:Species, data = iris)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.73356 -0.22783  0.00482  0.19809  0.73456
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   1.99438    0.29804   6.692 4.73e-10 ***
## Sepal.Width                   0.50510    0.08576   5.890 2.68e-08 ***
## Petal.Length                  0.87028    0.07123  12.218  < 2e-16 ***
## Petal.Width                   0.03171    0.41995   0.076  0.93992
## Speciesversicolor            -0.19338    0.35655  -0.542  0.58842
## Speciesvirginica             -1.27155    0.43169  -2.946  0.00377 **
```

null hypothesis for p: there is no difference between the versicolor and virginica

```
## Petal.Width:Speciesversicolor -0.76438    0.47719  -1.602  0.11142
## Petal.Width:Speciesvirginica  -0.26310    0.44288  -0.594  0.55341
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3047 on 142 degrees of freedom
## Multiple R-squared:  0.871,  Adjusted R-squared:  0.8646
## F-statistic: 136.9 on 7 and 142 DF,  p-value: < 2.2e-16
```

Question: How do we interpret the estimate for `Petal.Width:Speciesversicolor`? The p-value?

We can test whether the *interaction as a whole* is significant using ANOVA. Note that the null model is `irismod2` from above: it's the same as `irismod5` without the interaction.

```
anova(irismod2, irismod5)
```

```
## Analysis of Variance Table
##
## Model 1: Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width + Species
## Model 2: Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width + Species +
##     Petal.Width:Species
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    144 13.556
## 2    142 13.184  2   0.37202 2.0034 0.1387
```

Question: From the above, can we conclude that the lines should be parallel?

reject then parallel?