Lab #5 - Predictive Regression I

Econ 224

September 11th, 2018

Introduction

This lab provides a crash course on least squares regression in R. In the interest of time we'll work with a very simple, but somewhat boring, dataset that requires very little explanation. In our next lab and on the problem set you'll use what you've learned here to look at much more interesting examples!

The `mtcars` Dataset

The built-in R dataset `mtcars` contains information on 32 models of automobile from 1973-74 as reported in *Motor Trend Magazine*. For more information on the variables, see the R help file `?mtcars`. Note that `mtcars` is a dataframe rather than a tibble. Just to keep things simple I won't convert it to a tibble. But don't worry: everything I demonstrate in this tutorial will work just as well with a tibble as with a dataframe. A tibble *is* a dataframe even though a datafram is not a tibble. (C.f. a square is a rectangle, but a rectangle is not a square.) Here are the first few rows of the `mtcars`:

```
head(mtcars)
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Our goal will be to predict `mpg` (fuel efficiency in miles/gallon) using the other variables such as `cyl` (# of cylinders), `disp` (engine displacement in cubic inches), `hp` (horsepower), and `wt` (weight in thousands of pounds).

The `lm` Command

The command for least squares regression in R is `lm` which stands for *linear model*. The basic syntax is as follows: `lm([Y variable] ~ [1st predictor] + ... + [pth predictor], [dataframe])`. For example, to predict `mpg` using `disp` and `hp` we would run the command

```
lm(mpg ~ disp, mtcars)
```

```

Call:
lm(formula = mpg ~ disp, data = mtcars)

Coefficients:
(Intercept)         disp
   29.59985     -0.04122
```

Exercise #1

Carry out a regression predicting `mpg` using `disp`, `hp`, `cyl` and `wt`

```
lm(mpg ~ disp + hp + cyl + wt, mtcars)
```

```
Call:
lm(formula = mpg ~ disp + hp + cyl + wt, data = mtcars)

Coefficients:
(Intercept)         disp           hp          cyl           wt
   40.82854      0.01160     -0.02054     -1.29332     -3.85390
```

Getting More Information from `lm`

If we simply run `lm` as above, R will display only the estimated regression coefficients: $\widehat{\beta}_0, \widehat{\beta}_1, \ldots, \widehat{\beta}_p$ along with the command used to run the regression: `Call`. To get more information, we need to *store* the results of our regression.

```
reg1 <- lm(mpg ~ disp + hp, mtcars)
```

If you run the preceding line of code in the R console, it won't produce any output. But if you check your R environment after running it, you'll see a new `List` object: `reg1`. To see what's inside this list, we can use the command `str`:

```
str(reg1)
```

```
List of 12
 $ coefficients : Named num [1:3] 30.7359 -0.0303 -0.0248
  ..- attr(*, "names")= chr [1:3] "(Intercept)" "disp" "hp"
 $ residuals    : Named num [1:32] -2.15 -2.15 -2.35 1.23 3.24 ...
  ..- attr(*, "names")= chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
 $ effects      : Named num [1:32] -113.65 -28.44 5.8 1.1 3.01 ...
  ..- attr(*, "names")= chr [1:32] "(Intercept)" "disp" "hp" "" ...
 $ rank         : int 3
 $ fitted.values: Named num [1:32] 23.1 23.1 25.1 20.2 15.5 ...
  ..- attr(*, "names")= chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
 $ assign       : int [1:3] 0 1 2
 $ qr           :List of 5
  ..$ qr   : num [1:32, 1:3] -5.657 0.177 0.177 0.177 0.177 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
  .. .. ..$ : chr [1:3] "(Intercept)" "disp" "hp"
  .. ..- attr(*, "assign")= int [1:3] 0 1 2
  ..$ qraux: num [1:3] 1.18 1.09 1.01
  ..$ pivot: int [1:3] 1 2 3
  ..$ tol  : num 1e-07
  ..$ rank : int 3
  ..- attr(*, "class")= chr "qr"
 $ df.residual  : int 29
```

```
 $ xlevels      : Named list()
 $ call         : language lm(formula = mpg ~ disp + hp, data = mtcars)
 $ terms        :Classes 'terms', 'formula'  language mpg ~ disp + hp
  .. ..- attr(*, "variables")= language list(mpg, disp, hp)
  .. ..- attr(*, "factors")= int [1:3, 1:2] 0 1 0 0 0 1
  .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. ..$ : chr [1:3] "mpg" "disp" "hp"
  .. .. .. ..$ : chr [1:2] "disp" "hp"
  .. ..- attr(*, "term.labels")= chr [1:2] "disp" "hp"
  .. ..- attr(*, "order")= int [1:2] 1 1
  .. ..- attr(*, "intercept")= int 1
  .. ..- attr(*, "response")= int 1
  .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  .. ..- attr(*, "predvars")= language list(mpg, disp, hp)
  .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric" "numeric"
  .. .. ..- attr(*, "names")= chr [1:3] "mpg" "disp" "hp"
 $ model        :'data.frame':  32 obs. of  3 variables:
  ..$ mpg : num [1:32] 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
  ..$ disp: num [1:32] 160 160 108 258 360 ...
  ..$ hp  : num [1:32] 110 110 93 110 175 105 245 62 95 123 ...
  ..- attr(*, "terms")=Classes 'terms', 'formula'  language mpg ~ disp + hp
  .. .. ..- attr(*, "variables")= language list(mpg, disp, hp)
  .. .. ..- attr(*, "factors")= int [1:3, 1:2] 0 1 0 0 0 1
  .. .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. .. ..$ : chr [1:3] "mpg" "disp" "hp"
  .. .. .. .. ..$ : chr [1:2] "disp" "hp"
  .. .. ..- attr(*, "term.labels")= chr [1:2] "disp" "hp"
  .. .. ..- attr(*, "order")= int [1:2] 1 1
  .. .. ..- attr(*, "intercept")= int 1
  .. .. ..- attr(*, "response")= int 1
  .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  .. .. ..- attr(*, "predvars")= language list(mpg, disp, hp)
  .. .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric" "numeric"
  .. .. .. ..- attr(*, "names")= chr [1:3] "mpg" "disp" "hp"
 - attr(*, "class")= chr "lm"
```

Don't panic: you don't need to know what all of these list elements are. The important thing to understand is that `lm` returns a *list* from which we can extract important information about the regression we have run. To extract the regression coefficient estimates, we use `coef`

```r
coef(reg1)
```

```
(Intercept)        disp          hp 
30.73590425 -0.03034628 -0.02484008 
```

To extract the regression residuals, we use `resid`

```r
resid(reg1)
```

```
          Mazda RX4       Mazda RX4 Wag          Datsun 710 
         -2.1480911          -2.1480911          -2.3483788 
     Hornet 4 Drive   Hornet Sportabout             Valiant 
```

```
          1.2258440            3.2357695           -3.1997835
         Duster 360            Merc 240D             Merc 230
          0.5745752           -0.3440204           -1.3033408
           Merc 280            Merc 280C           Merc 450SE
         -3.3945383           -4.7945383           -1.4951865
         Merc 450SL          Merc 450SLC   Cadillac Fleetwood
         -0.5951865           -2.6951865           -0.9202450
Lincoln Continental    Chrysler Imperial             Fiat 128
         -1.0359995            3.0296761            5.6917931
        Honda Civic       Toyota Corolla        Toyota Corona
          3.2529931            6.9363213           -3.1818286
   Dodge Challenger          AMC Javelin           Camaro Z28
         -1.8597761           -2.5846240           -0.7288875
   Pontiac Firebird            Fiat X1-9        Porsche 914-2
          4.9496206            0.6008970            1.1752002
       Lotus Europa       Ford Pantera L         Ferrari Dino
          5.3569558            2.2734203           -2.2886799
      Maserati Bora           Volvo 142E
          1.7197522           -2.9564359
```

and to extract the *fitted values* i.e. the predicted values of Y, we use `fitted.values`

```r
fitted.values(reg1)
```

```
          Mazda RX4        Mazda RX4 Wag           Datsun 710
           23.14809             23.14809             25.14838
     Hornet 4 Drive    Hornet Sportabout              Valiant
           20.17416             15.46423             21.29978
         Duster 360            Merc 240D             Merc 230
           13.72542             24.74402             24.10334
           Merc 280            Merc 280C           Merc 450SE
           22.59454             22.59454             17.89519
         Merc 450SL          Merc 450SLC   Cadillac Fleetwood
           17.89519             17.89519             11.32025
Lincoln Continental    Chrysler Imperial             Fiat 128
           11.43600             11.67032             26.70821
        Honda Civic       Toyota Corolla        Toyota Corona
           27.14701             26.96368             24.68183
   Dodge Challenger          AMC Javelin           Camaro Z28
           17.35978             17.78462             14.02889
   Pontiac Firebird            Fiat X1-9        Porsche 914-2
           14.25038             26.69910             24.82480
       Lotus Europa       Ford Pantera L         Ferrari Dino
           25.04304             13.52658             21.98868
      Maserati Bora           Volvo 142E
           13.28025             24.35644
```

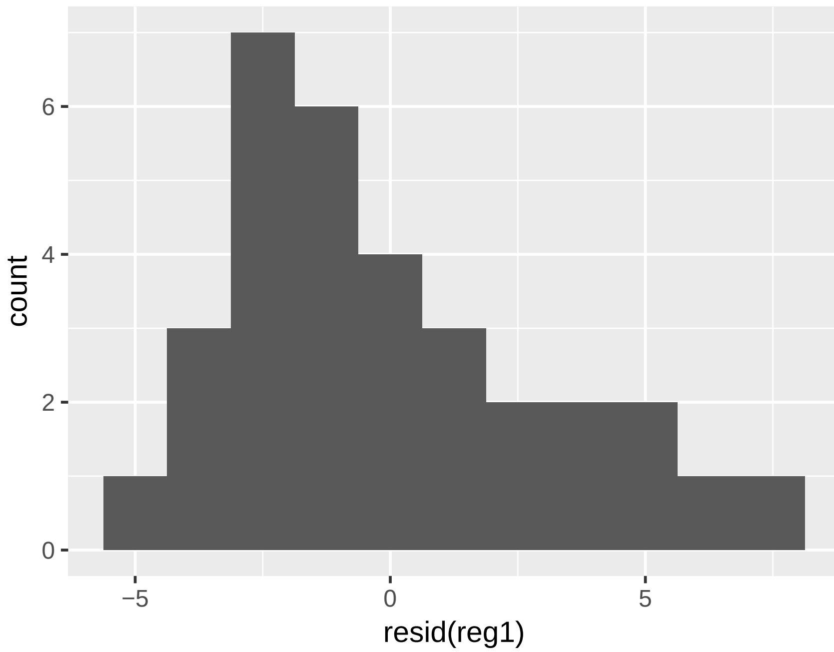
Exercise # 2

1. Plot a histogram of the residuals from `reg1` using `ggplot` with a bin width of 1.25. Is there anything noteworthy about this plot?
2. Calculate the residuals "by hand" by subtracting the fitted values from `reg1` from the column `mpg` in mtcars. Use the R function `all.equal` to check that this gives the same result as `resid`.

Solution to Exercise #2

1. There seems to be some right skewness in the residuals.

```
library(ggplot2)
ggplot() +
  geom_histogram(aes(x = resid(reg1)), binwidth = 1.25)
```



2. They give exactly the same result:

```
all.equal(resid(reg1), mtcars$mpg - fitted.values(reg1))
```

```
[1] TRUE
```

Summarizing Regression Output

To view the usual summary of regression output, we use the `summary` command:

```
summary(reg1)
```

```

Call:
lm(formula = mpg ~ disp + hp, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.7945 -2.3036 -0.8246  1.8582  6.9363 
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 30.735904   1.331566  23.083  < 2e-16 ***
disp        -0.030346   0.007405  -4.098 0.000306 ***
hp          -0.024840   0.013385  -1.856 0.073679 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.127 on 29 degrees of freedom
Multiple R-squared:  0.7482,    Adjusted R-squared:  0.7309
F-statistic: 43.09 on 2 and 29 DF,  p-value: 2.062e-09
```

Among other things, `summary` shows us the coefficient estimates and associated standard errors. It also displays the t-value (Estimate / SE) and associated p-value for a test of the null hypothesis $H_0: \beta = 0$ versus $H_1: \beta \neq 0$. Farther down in the output, `summary` provides the residual standard error and R-squared. It turns out the summary command *itself* returns a list. In particular,

```r
str(summary(reg1))
```

```
List of 11
 $ call         : language lm(formula = mpg ~ disp + hp, data = mtcars)
 $ terms        :Classes 'terms', 'formula'  language mpg ~ disp + hp
  .. ..- attr(*, "variables")= language list(mpg, disp, hp)
  .. ..- attr(*, "factors")= int [1:3, 1:2] 0 1 0 0 0 1
  .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. ..$ : chr [1:3] "mpg" "disp" "hp"
  .. .. .. ..$ : chr [1:2] "disp" "hp"
  .. ..- attr(*, "term.labels")= chr [1:2] "disp" "hp"
  .. ..- attr(*, "order")= int [1:2] 1 1
  .. ..- attr(*, "intercept")= int 1
  .. ..- attr(*, "response")= int 1
  .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  .. ..- attr(*, "predvars")= language list(mpg, disp, hp)
  .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric" "numeric"
  .. .. ..- attr(*, "names")= chr [1:3] "mpg" "disp" "hp"
 $ residuals    : Named num [1:32] -2.15 -2.15 -2.35 1.23 3.24 ...
  ..- attr(*, "names")= chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
 $ coefficients : num [1:3, 1:4] 30.7359 -0.0303 -0.0248 1.3316 0.0074 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:3] "(Intercept)" "disp" "hp"
  .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
 $ aliased      : Named logi [1:3] FALSE FALSE FALSE
  ..- attr(*, "names")= chr [1:3] "(Intercept)" "disp" "hp"
 $ sigma        : num 3.13
 $ df           : int [1:3] 3 29 3
 $ r.squared    : num 0.748
 $ adj.r.squared: num 0.731
 $ fstatistic   : Named num [1:3] 43.1 2 29
  ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
 $ cov.unscaled : num [1:3, 1:3] 1.81e-01 -1.18e-04 -8.38e-04 -1.18e-04 5.61e-06 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:3] "(Intercept)" "disp" "hp"
  .. ..$ : chr [1:3] "(Intercept)" "disp" "hp"
 - attr(*, "class")= chr "summary.lm"
```

This fact can come in handy when you want to *extract* some of the values from the regression summary table to use for some other purpose. For example, we can display *only* the R-squared as follows: We could do this as follows:

```
summary(reg1)$r.squared
```

```
[1] 0.7482402
```

and only the F-statistic with its associated degrees of freedom as follows:

```
summary(reg1)$fstatistic
```

```
   value    numdf    dendf 
43.09458  2.00000 29.00000 
```

Exercise #3

1. Use `summary` to display the results of the regression you ran in Exercise #1 above.
2. Figure out how to extract and display *only* the regression standard error from the results of `summary` in part 1 of this exercise.
3. Calculate the regression standard error for the regression from part 1 of this exercise "by hand" and make sure that your answer matches part 2. Hint: use `resid`

Solution to Exercise #3

1. Store the result of `lm` and use `summary`:

```
myreg <- lm(mpg ~ disp + hp + cyl + wt, mtcars)
summary(myreg)
```

```

Call:
lm(formula = mpg ~ disp + hp + cyl + wt, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.0562 -1.4636 -0.4281  1.2854  5.8269 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 40.82854    2.75747  14.807 1.76e-14 ***
disp         0.01160    0.01173   0.989 0.331386    
hp          -0.02054    0.01215  -1.691 0.102379    
cyl         -1.29332    0.65588  -1.972 0.058947 .  
wt          -3.85390    1.01547  -3.795 0.000759 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.513 on 27 degrees of freedom
Multiple R-squared:  0.8486,	Adjusted R-squared:  0.8262 
F-statistic: 37.84 on 4 and 27 DF,  p-value: 1.061e-10
```

2. The appropriate list item is called `sigma`

```
summary(myreg)$sigma
```

```
[1] 2.51252
```

3. Let $\widehat{\epsilon}_1, \ldots, \widehat{\epsilon}_n$ denote the residuals. Then the standard error of the regression is given by

$$\sqrt{\frac{\sum_{i=1}^{n} \widehat{\epsilon}_i^2}{n-p-1}}$$

where p is the number of X-variables in the regression. We can implement this in R using `resid` and compare it to the results calculated automatically by `summary` as follows:

```
ehat <- resid(myreg)
n <- length(ehat)
p <- length(coef(myreg)) - 1
sqrt(sum(ehat^2) / (n - p - 1))
```

```
[1] 2.51252
```

```
summary(myreg)$sigma
```

```
[1] 2.51252
```

Regression Without an Intercept

More than 99% of the time, it makes sense for us to include an intercept β_0 in a linear regression. To see why, consider the meaning of β_0: this is the value of Y that we would predict if $X_1 = X_2 = \ldots = X_p = 0$. Unless we have some very strong *a priori* knowledge, there is no reason to suppose that the mean of Y should be zero when all of the predictors are zero. In some very special cases, however, we *do* have such special knowledge. To *force* the intercept in a regression to be zero we use the syntax `-1`, for example

```
summary(lm(mpg ~ disp - 1, mtcars))
```

```

Call:
lm(formula = mpg ~ disp - 1, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-17.471  -2.900   7.034  14.799  29.702 

Coefficients:
     Estimate Std. Error t value Pr(>|t|)    
disp 0.059049   0.009765   6.047 1.07e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.42 on 31 degrees of freedom
Multiple R-squared:  0.5412,    Adjusted R-squared:  0.5264 
F-statistic: 36.57 on 1 and 31 DF,  p-value: 1.073e-06
```

Exercise #4

What do you get if you run the regression `lm(mpg ~ 1, mtcars)`?

Solution to Exercise #4

This calculates the sample mean of `mpg`

```
lm(mpg ~ 1, mtcars)
```

```

Call:
lm(formula = mpg ~ 1, data = mtcars)

Coefficients:
(Intercept)
      20.09
```

```
with(mtcars, mean(mpg))
```

```
[1] 20.09062
```

F-tests

Suppose we want to test the *joint* null hypothesis $H_0: \beta_1 = \beta_2 = \cdots = \beta_p = 0$ versus the alternative that at least one of these coefficients is non-zero. This is equivalent to testing the null hypothesis that none of the predictors $X_1, \cdots, X_p$ is helpful in predicting Y. This test is automatically carried out by `summary`. Consider a regression that uses `disp`, `hp`, `wt` and `cyl` to predict `mpg`

```
reg2 <- lm(mpg ~ disp + hp + wt + cyl, mtcars)
summary(reg2)
```

```

Call:
lm(formula = mpg ~ disp + hp + wt + cyl, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-4.0562 -1.4636 -0.4281  1.2854  5.8269

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 40.82854    2.75747  14.807 1.76e-14 ***
disp         0.01160    0.01173   0.989 0.331386
hp          -0.02054    0.01215  -1.691 0.102379
wt          -3.85390    1.01547  -3.795 0.000759 ***
cyl         -1.29332    0.65588  -1.972 0.058947 .
---
```

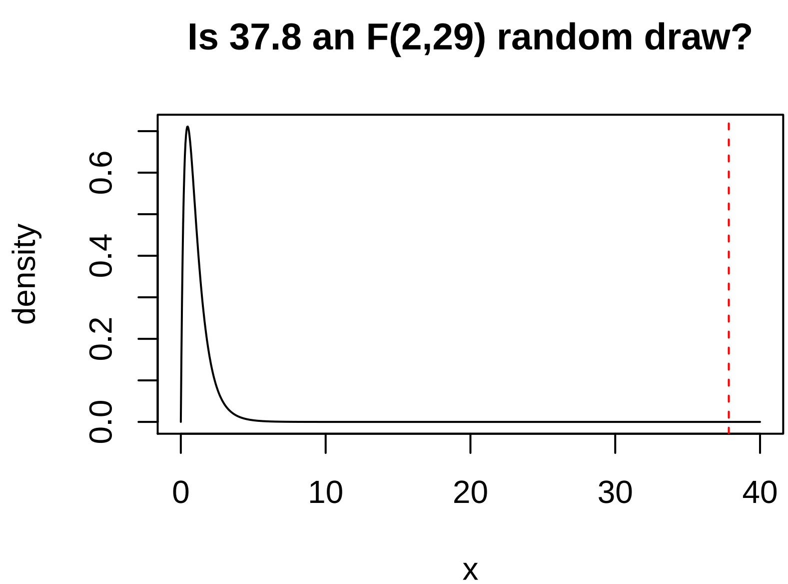
```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.513 on 27 degrees of freedom
Multiple R-squared:  0.8486,    Adjusted R-squared:  0.8262
F-statistic: 37.84 on 4 and 27 DF,  p-value: 1.061e-10
```

At the very bottom of the `summary` output is a line that begins `F-statistic`. This line contains the results of the F-test of the joint null described above. In this case the p-value is miniscule: there is very strong evidence that at least one of the predictors is helpful in predicting `mpg`. To get a better understanding of what's involved here, we can calculate the p-value "by hand" as follows:

```
summary(reg2)$fstatistic
```

```
   value    numdf    dendf 
37.84413  4.00000 27.00000 
```

```
curve(expr = df(x, 4, 27), 0, 40, n = 1001,
      ylab = 'density',
      main = 'Is 37.8 an F(2,29) random draw?')
abline(v = 37.84413, lty = 2, col = 'red')
```




```
1 - pf(37.84413, 4, 27)
```

```
[1] 1.061086e-10
```

Under the null hypothesis, the F-statistic is a draw from an F random variable with numerator degrees of freedom 2 and denominator degrees of freedom 29. (If you are unfamiliar with the F-distribution see my Tutorial Friends of the Normal Distribution.) So is is plausible that the value 37.8 came from an $F(2, 29)$ distribution? From my plot of the corresponding density function, the answer is clearly *no*. We have very strong evidence against the null hypothesis.

Sometimes we only want to test the null that a *subset* of our predictors is unhelpful for predicting Y. For example, in `reg2` we might ask whether `wt` and `cyl` provide *extra* information for predicting `mpg` after we have already included disp and hp in our model. To carry out this test, we use the function linearHypothesis from the package car. Make sure to install this package before proceeding. Note the syntax: linearHypothesis([lm object], c('[first restriction]', ..., '[last restriction]')

```
library(car)
```

```
Loading required package: carData
```

```
linearHypothesis(reg2, c('wt = 0', 'cyl = 0'))
```

```
Linear hypothesis test

Hypothesis:
wt = 0
cyl = 0

Model 1: restricted model
Model 2: mpg ~ disp + hp + wt + cyl

  Res.Df    RSS Df Sum of Sq     F  Pr(>F)
1     29 283.49
2     27 170.44  2    113.05 8.954 0.00104 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The two key numbers to look for in the output are F, the value of the F-statistic, and Pr(>F), the p-value. The other values are the inputs used to calculate F. (See Equation 3.24 in ISL.) In this instance we strongly reject the null hypothesis that wt and cyl are irrelevant for predicting mpg after controlling for disp and hp.

Exercise #5

Generate two vectors of independent standard normal draws x and z. Each vector should contain as many elements as there are rows in mtcars. Use the command set.seed(1234) before making your random draws so that they are replicable. (By first setting the seed to a fixed number, you ensure that the same random draws will be made any time that you re-run this code chunk.) Carry out a new regression, reg3, that *augments* reg2 by adding the predictors x and z. Then carry out an F-test the null hypothesis that x and z are irrelevant for predicting mpg after controlling for disp, hp, wt, and cyl. Interpret your findings. Do the results of the test make sense?

Solution to Exercise #5

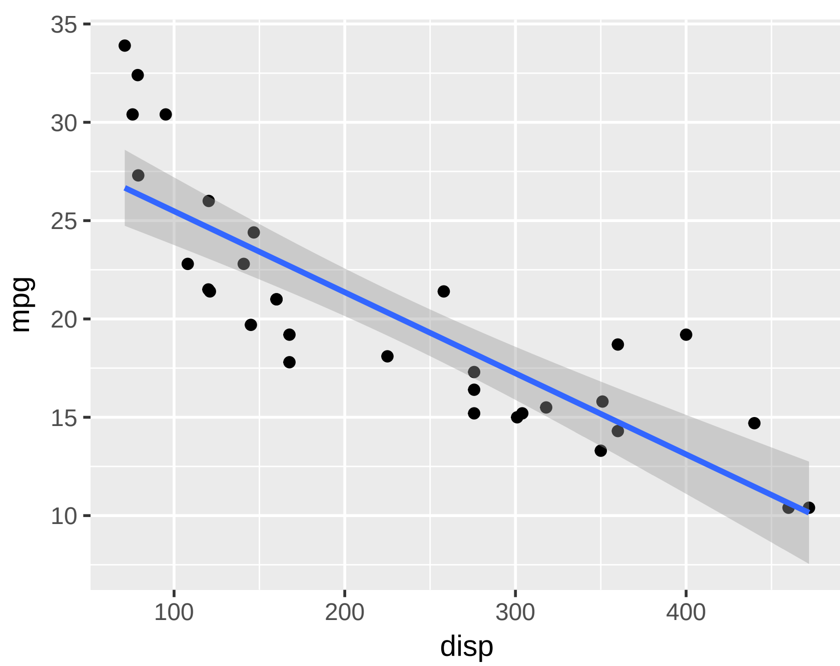
```
set.seed(1234)
n <- nrow(mtcars)
x <- rnorm(n)
z <- rnorm(n)
reg3 <- lm(mpg ~ disp + hp + wt + cyl + x + z, mtcars)
linearHypothesis(reg3, c('x = 0', 'z = 0'))
```

```
Linear hypothesis test

Hypothesis:
x = 0
z = 0

Model 1: restricted model
Model 2: mpg ~ disp + hp + wt + cyl + x + z

  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     27 170.44
2     25 169.95  2    0.4917 0.0362 0.9645
```

Plotting the Regression Line

To get an idea of whether our regression model looks reasonable, it's always a good idea to make some plots. When we have a single predictor X, it is common to plot the raw X and Y observations along with the regression line. It's easy to do this using `ggplot`. Suppose we wanted to predict `mpg` using `disp`. Here's the `ggplot` way to plot the data and regression line:

```
ggplot(mtcars, aes(x = disp, y = mpg)) +
  geom_point() +
  geom_smooth(method = 'lm')
```



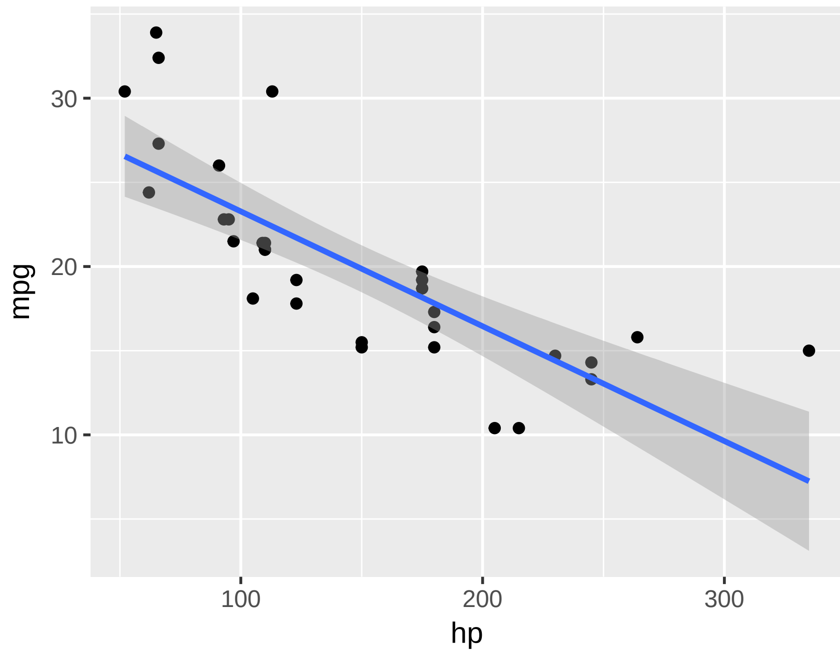
Notice that I specified `aes` inside of `ggplot`. This ensures that both `geom_point` and `geom_smooth` "know" which variable is `x` and which variable is `y`. Notice moreover, that the `ggplot` way of doing this includes *error bounds* for the regression line. This is a handy way of visualizing the uncertainty in the line we've fit.

Exercise #6

Make a ggplot with hp on the x-axis and mpg on the y-axis that includes the regression line for predicting mpg from hp.

Solution to Exercise #6

```
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point() +
  geom_smooth(method = 'lm')
```



Polynomial Regression

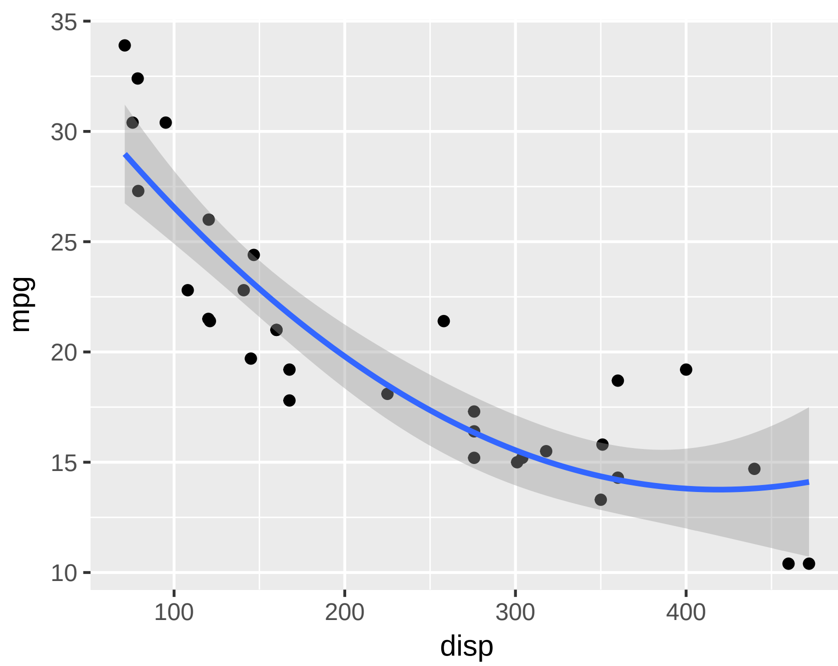
In your next reading assignment, you'll learn about *polynomial regression*. The "linear" in linear regression does not actually refer to the relationship between Y and the predictors X; it refers to the relationship between Y and the *coefficients* $\beta_0, \beta_1, ..., \beta_p$. In the expression $Y = \beta_0 + \beta_1 X + \epsilon$, Y is a linear function of β_0 and β_1 and it is *also* a linear function of X. In the expression $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$, Y is *still* a linear function of the coefficients, but a *quadratic* function of X. This is a simple example of polynomial regression, which allows us to model more complicated relationships between X and Y. Notice, for example, that the relationship between `mpg` and `disp` looks like it might be curved. To accommodate such a relationship, let's try a polynomial regression that includes includes `disp` and `disp^2`. To do this we use the syntax `I([some transformation of a predictor)`

```
reg3 <- lm(mpg ~ disp + I(disp^2), mtcars)
summary(reg3)
```

```
Call:
lm(formula = mpg ~ disp + I(disp^2), data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.9112 -1.5269 -0.3124  1.3489  5.3946 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  3.583e+01  2.209e+00  16.221 4.39e-16 ***
disp        -1.053e-01  2.028e-02  -5.192 1.49e-05 ***
I(disp^2)    1.255e-04  3.891e-05   3.226   0.0031 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.837 on 29 degrees of freedom
Multiple R-squared:  0.7927,	Adjusted R-squared:  0.7784 
F-statistic: 55.46 on 2 and 29 DF,  p-value: 1.229e-10
```

Notice that the coefficient on the quadratic term is *highly* statistically significant, which is strong evidence of curvature in the relationship between `mpg` and `disp`. We can plot the polynomial regression as follows:

```
ggplot(mtcars, aes(x = disp, y = mpg)) +
  geom_point() +
  geom_smooth(method = 'lm', formula = y ~ x + I(x^2))
```



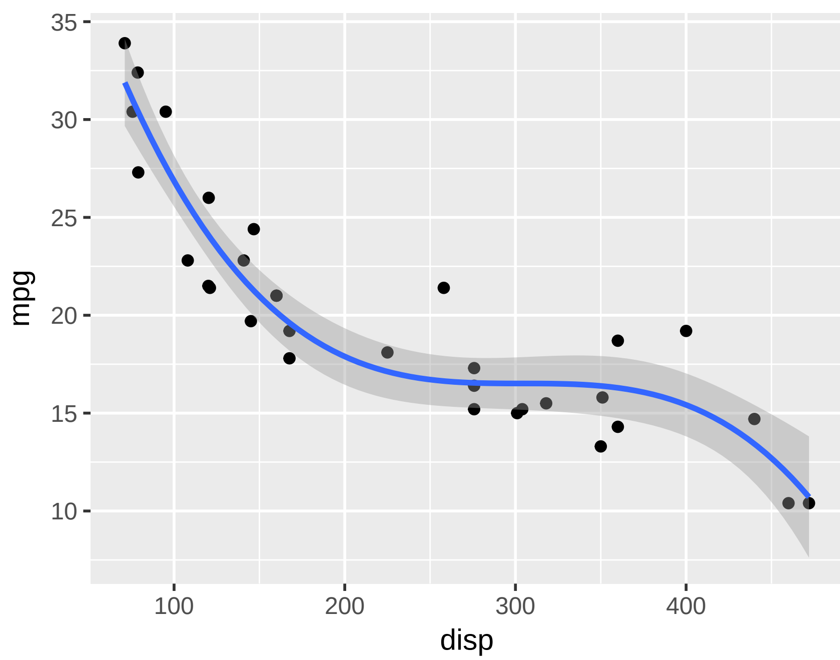
Notice that this requires us to specify the `formula` argument so that `ggplot` knows that we want to plot a quadratic relationship.

Exercise #7

In my code above, I considered a quadratic relationship between `mpg` and `disp`. Add a cubic term to the regression, plot the points and regression function, and display the results using `summary`. Comment on the results.

Solution to Exercise #7

```
reg4 <- lm(mpg ~ disp + I(disp^2) + I(disp^3), mtcars)
ggplot(mtcars, aes(x = disp, y = mpg)) +
  geom_point() +
  geom_smooth(method = 'lm', formula = y ~ x + I(x^2) + I(x^3))
```




```
summary(reg4)
```

```
Call:
lm(formula = mpg ~ disp + I(disp^2) + I(disp^3), data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-3.0896 -1.5653 -0.3619  1.4368  4.7617

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.070e+01  3.809e+00  13.310 1.25e-13 ***
disp        -3.372e-01  5.526e-02  -6.102 1.39e-06 ***
I(disp^2)    1.109e-03  2.265e-04   4.897 3.68e-05 ***
```

```
I(disp^3)   -1.217e-06  2.776e-07  -4.382  0.00015 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.224 on 28 degrees of freedom
Multiple R-squared:  0.8771,    Adjusted R-squared:  0.8639
F-statistic: 66.58 on 3 and 28 DF,  p-value: 7.347e-13
```

Interaction Effects

An idea closely related to polynomial regression that will also be discussed in your next reading assignment is that of an *interaction*. In the model $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_3 + \epsilon$, Y is a linear function of $\beta_0, \beta_1, \beta_2$, and β_3 but a *nonlinear* function of X_1 and X_2. The term $X_1 \times X_2$ is called an *interaction*. To run a regression with an interaction, we use the syntax `[One Predictor]:[Another Predictor]` for example

```r
lm(mpg ~ disp + hp + disp:hp, mtcars)
```

```
Call:
lm(formula = mpg ~ disp + hp + disp:hp, data = mtcars)

Coefficients:
(Intercept)         disp           hp      disp:hp
   39.67426     -0.07337     -0.09789      0.00029
```

Exercise #8

Fit a regression using `disp`, `disp^2`, `wt`, `wt^2` and the interaction between `wt` and `disp` to predict `mpg` and display the coefficient estimates.

Solution to Exercise #8

```r
reg5 <- lm(mpg ~ disp + wt + I(disp^2) + I(wt^2) + disp:wt, mtcars)
coef(reg5)
```

```
  (Intercept)          disp            wt     I(disp^2)       I(wt^2)
 4.692786e+01 -3.172401e-02 -1.062827e+01  2.019044e-04  2.079131e+00
      disp:wt
-2.660633e-02
```

Predicting New Observations

To predict new observations based on a fitted linear regression model in R, we use the `predict` function. For example, consider three hypothetical cars with the following values of displacement and horsepower

```r
mydat <- data.frame(disp = c(100, 200, 300),
                    hp = c(150, 100, 200))
mydat
```

```
  disp  hp
1  100 150
2  200 100
3  300 200
```

Based on the results of `reg1`, we would predict that these cars have the following values of `mpg`

```r
predict(reg1, mydat)
```

```
       1        2        3 
23.97526 22.18264 16.66401 
```

Note the syntax: the first argument of `predict` is a set of regression results while the second is a dataframe (or tibble) with column names that *match* the variables in the regression we carried out.

Exercise #9

1. Check the predictions of `predict` in the preceding chunk "by hand" using `mydat`, `coef`, and `reg1`.
2. Consider three cars with `disp` equal to 125, 175, and 225, respectively. Predict `mpg` for each of these based on the regression from Exercise #7.

Solution to Exercise #9

1. Many possibilities. Here's one:

```r
b <- coef(reg1)
b0 <- b[1]
b1 <- b[2]
b2 <- b[3]
b0 + b1 * mydat$disp + b2 * mydat$hp
```

```
[1] 23.97526 22.18264 16.66401
```

2. Use the following:

```r
mydat <- data.frame(disp = c(125, 175, 225))
predict(reg4, mydat)
```

```
       1        2        3 
23.50258 19.13630 17.12228 
```