Lab 12 (In Class)

Maria Barouti

3/18/2021

Based on `Applied Statistics with R (appliedstats)` by David Dalpiaz (https://github.com/daviddalpiaz/appliedstats)

Indicator or Dummy Variables

We will briefly use the built in dataset `mtcars`.

```
mtcars
```

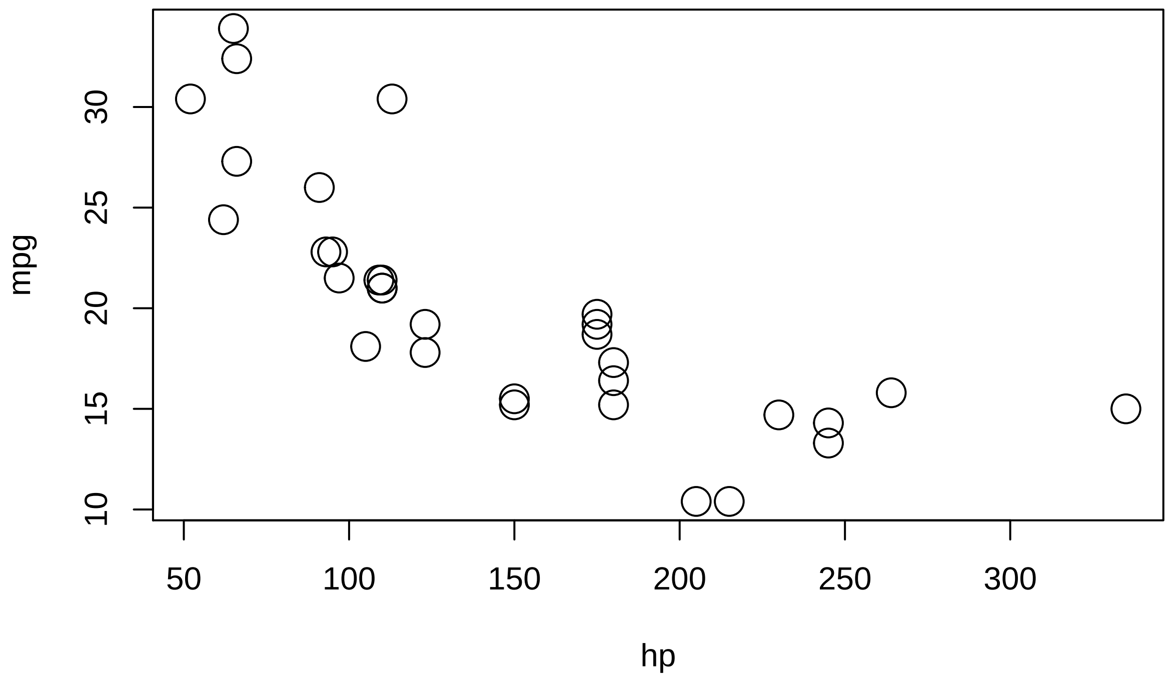
```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

We will be interested in three of the variables: `mpg`, `hp`, and `am`.

- `mpg`: fuel efficiency, in miles per gallon.
- `hp`: horsepower, in foot-pounds per second.
- `am`: transmission. Automatic or manual.

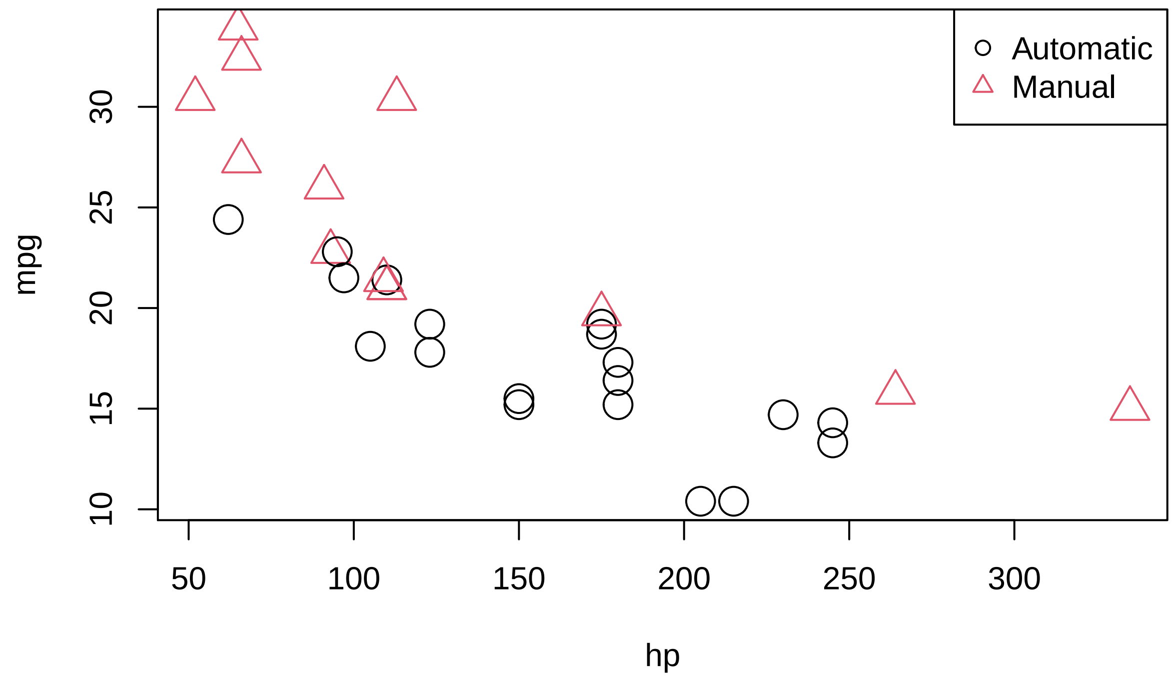
As we often do, we will start by plotting the data. We are interested in `mpg` as the response variable, and `hp` as a predictor.

```
plot(mpg ~ hp, data = mtcars, cex = 2)
```



Since we are also interested in the transmission type, we could also label the points accordingly.

```
plot(mpg ~ hp, data = mtcars, col = am + 1, pch = am + 1, cex = 2)
legend("topright", c("Automatic", "Manual"), col = c(1, 2), pch = c(1, 2))
```



We used a common `R` "trick" when plotting this data. The `am` variable takes two possible values; `0` for automatic transmission, and `1` for manual transmissions. `R` can use numbers to represent colors, however the color for `0` is white. So we take the `am` vector and add `1` to it. Then observations with automatic transmissions are now represented by `1`, which is black in `R`, and manual transmission are represented by `2`, which is red in `R`. (Note, we are only adding `1` inside the call to `plot()`, we are not actually modifying the values stored in `am`.)

We now fit the SLR model

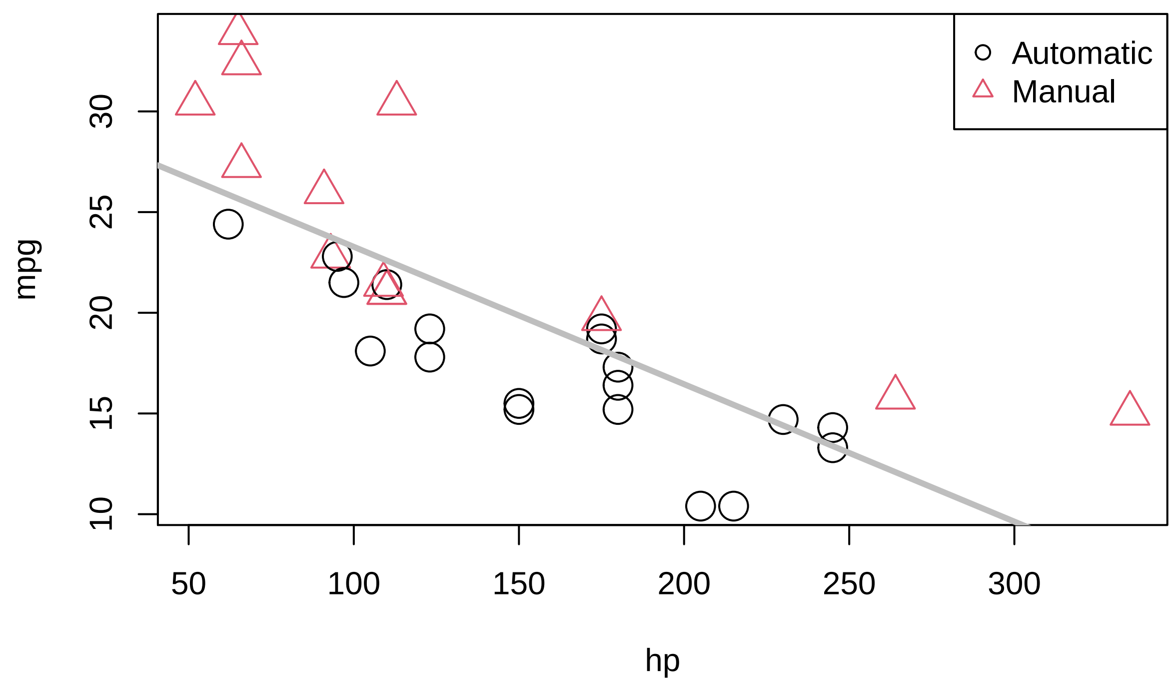
$$Y = \beta_0 + \beta_1 X_1 + \epsilon,$$

where Y is `mpg` and X_1 is `hp`. For notational brevity, we drop the index i for observations.

```
mpg_hp_slr = lm(mpg ~ hp, data = mtcars)
```

We then re-plot the data and add the fitted line to the plot.

```
plot(mpg ~ hp, data = mtcars, col = am + 1, pch = am + 1, cex = 2)
abline(mpg_hp_slr, lwd = 3, col = "grey")
legend("topright", c("Automatic", "Manual"), col = c(1, 2), pch = c(1, 2))
```



We should notice a pattern here. The red, manual observations largely fall above the line, while the black, automatic observations are mostly below the line. This means our model underestimates the fuel efficiency of manual transmissions, and overestimates the fuel efficiency of automatic transmissions. To correct for this, we will add a predictor to our model, namely, `am` as X_2.

Our new model is

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon,$$

where X_1 and Y remain the same, but now

$$X_2 = \begin{cases} 1 & \text{manual transmission} \\ 0 & \text{automatic transmission} \end{cases}.$$

First, note that `am` is already a dummy variable, since it uses the values `0` and `1` to represent automatic and manual transmissions. Often, a variable like `am` would store the character values `auto` and `man` and we would either have to convert these to `0` and `1`, or, as we will see later, `R` will take care of creating dummy variables for us.

So, to fit the above model, we do so like any other multiple regression model we have seen before.

```
mpg_hp_add = lm(mpg ~ hp + am, data = mtcars)
```

Briefly checking the output, we see that `R` has estimated the three β parameters.

```
mpg_hp_add
```

```
##
## Call:
## lm(formula = mpg ~ hp + am, data = mtcars)
##
## Coefficients:
## (Intercept)           hp           am
##    26.58491     -0.05889      5.27709
```

Since X_2 can only take values `0` and `1`, we can effectively write two different models, one for manual and one for automatic transmissions.

For automatic transmissions, that is $X_2 = 0$, we have,

$$Y = \beta_0 + \beta_1 X_1 + \epsilon.$$

Then for manual transmissions, that is $X_2 = 1$, we have,

$$Y = (\beta_0 + \beta_2) + \beta_1 X_1 + \epsilon.$$

Notice that these models share the same slope, β_1, but have different intercepts, differing by β_2. So the change in `mpg` is the same for both models, but on average `mpg` differs by β_2 between the two transmission types.

We'll now calculate the estimated slope and intercept of these two models so that we can add them to a plot. Note that:

- $\hat{\beta}_0$ = `coef(mpg_hp_add)[1]` = 26.5849137
- $\hat{\beta}_1$ = `coef(mpg_hp_add)[2]` = -0.0588878
- $\hat{\beta}_2$ = `coef(mpg_hp_add)[3]` = 5.2770853

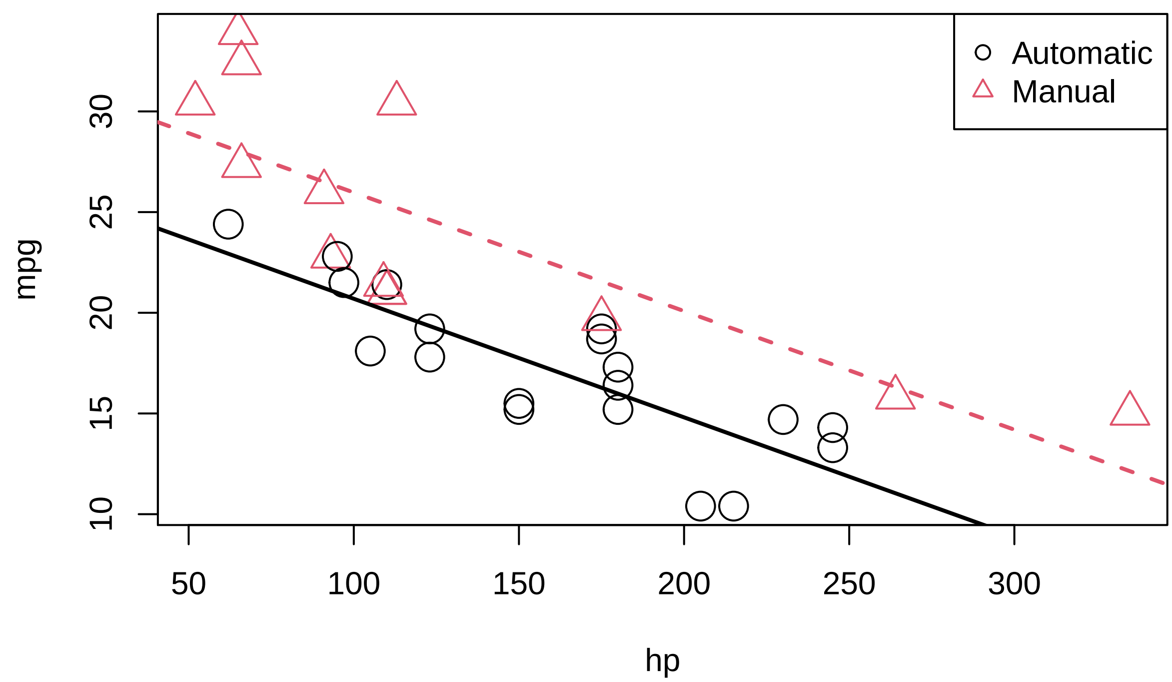
We can then combine these to calculate the estimated slope and intercepts.

```r
int_auto = coef(mpg_hp_add)[1]
int_manu = coef(mpg_hp_add)[1] + coef(mpg_hp_add)[3]

slope_auto = coef(mpg_hp_add)[2]
slope_manu = coef(mpg_hp_add)[2]
```

Re-plotting the data, we use these slopes and intercepts to add the "two" fitted models to the plot.

```r
plot(mpg ~ hp, data = mtcars, col = am + 1, pch = am + 1, cex = 2)
abline(int_auto, slope_auto, col = 1, lty = 1, lwd = 2) # add line for auto
abline(int_manu, slope_manu, col = 2, lty = 2, lwd = 2) # add line for manual
legend("topright", c("Automatic", "Manual"), col = c(1, 2), pch = c(1, 2))
```



We notice right away that the points are no longer systematically incorrect. The red, manual observations vary about the red line in no particular pattern without underestimating the observations as before. The black, automatic points vary about the black line, also without an obvious pattern.

They say a picture is worth a thousand words, but as a statistician, sometimes a picture is worth an entire analysis. The above picture makes it plainly obvious that β_2 is significant, but let's verify mathematically. Essentially we would like to test:

$$H_0 : \beta_2 = 0 \quad \text{vs} \quad H_1 : \beta_2 \neq 0.$$

This is nothing new. Again, the math is the same as the multiple regression analyses we have seen before. We could perform either a t or F test here. The only difference is a slight change in interpretation. We could think of this as testing a model with a single line (H_0) against a model that allows two lines (H_1).

To obtain the test statistic and p-value for the t-test, we would use

```r
summary(mpg_hp_add)$coefficients["am",]
```

```
##     Estimate   Std. Error      t value     Pr(>|t|)
## 5.277085e+00 1.079541e+00 4.888270e+00 3.460318e-05
```

To do the same for the F test, we would use

```r
anova(mpg_hp_slr, mpg_hp_add)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ hp
## Model 2: mpg ~ hp + am
```

```
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1     30 447.67
## 2     29 245.44  1    202.24 23.895 3.46e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Notice that these are indeed testing the same thing, as the p-values are exactly equal. (And the F test statistic is the t test statistic squared.)

Recapping some interpretations:

- $\hat{\beta}_0 = 26.5849137$ is the estimated average `mpg` for a car with an automatic transmission and **0** `hp`.
- $\hat{\beta}_0 + \hat{\beta}_2 = 31.8619991$ is the estimated average `mpg` for a car with a manual transmission and **0** `hp`.
- $\hat{\beta}_2 = 5.2770853$ is the estimated **difference** in average `mpg` for cars with manual transmissions as compared to those with automatic transmission, for **any** `hp`.
- $\hat{\beta}_1 = -0.0588878$ is the estimated change in average `mpg` for an increase in one `hp`, for **either** transmission types.

Interactions

To remove the "same slope" restriction, we will now discuss **interaction**. To illustrate this concept, we will return to the `autompg` dataset we created in the last chapter, with a few more modifications.

```
# read data frame from the web
autompg = read.table(
  "http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data",
  quote = "\"",
  comment.char = "",
  stringsAsFactors = FALSE)
# give the dataframe headers
colnames(autompg) = c("mpg", "cyl", "disp", "hp", "wt", "acc", "year", "origin", "name")
# remove missing data, which is stored as "?"
autompg = subset(autompg, autompg$hp != "?")
# remove the plymouth reliant, as it causes some issues
autompg = subset(autompg, autompg$name != "plymouth reliant")
# give the dataset row names, based on the engine, year and name
rownames(autompg) = paste(autompg$cyl, "cylinder", autompg$year, autompg$name)
# remove the variable for name
autompg = subset(autompg, select = c("mpg", "cyl", "disp", "hp", "wt", "acc", "year", "origin"))
# change horsepower from character to numeric
autompg$hp = as.numeric(autompg$hp)
# create a dummary variable for foreign vs domestic cars. domestic = 1.
autompg$domestic = as.numeric(autompg$origin == 1)
# remove 3 and 5 cylinder cars (which are very rare.)
autompg = autompg[autompg$cyl != 5,]
autompg = autompg[autompg$cyl != 3,]
# the following line would verify the remaining cylinder possibilities are 4, 6, 8
#unique(autompg$cyl)
# change cyl to a factor variable
autompg$cyl = as.factor(autompg$cyl)
```

```
str(autompg)
```

```
## 'data.frame':    383 obs. of  9 variables:
```

```
## $ mpg     : num  18 15 18 16 17 15 14 14 14 15 ...
## $ cyl     : Factor w/ 3 levels "4","6","8": 3 3 3 3 3 3 3 3 3 3 ...
## $ disp    : num  307 350 318 304 302 429 454 440 455 390 ...
## $ hp      : num  130 165 150 150 140 198 220 215 225 190 ...
## $ wt      : num  3504 3693 3436 3433 3449 ...
## $ acc     : num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
## $ year    : int  70 70 70 70 70 70 70 70 70 70 ...
## $ origin  : int  1 1 1 1 1 1 1 1 1 1 ...
## $ domestic: num  1 1 1 1 1 1 1 1 1 1 ...
```

We've removed cars with `3` and `5` cylinders , as well as created a new variable `domestic` which indicates whether or not a car was built in the United States. Removing the `3` and `5` cylinders is simply for ease of demonstration later. The new variable `domestic` takes the value `1` if the car was built in the United States, and `0` otherwise, which we will refer to as "foreign." (We are arbitrarily using the United States as the reference point here.) We have also made `cyl` and `origin` into factor variables, which we will discuss later.

We'll now be concerned with three variables: `mpg`, `disp`, and `domestic`. We will use `mpg` as the response. We can fit a model,

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon,$$

where

- Y is `mpg`, the fuel efficiency in miles per gallon,
- X_1 is `disp`, the displacement in cubic inches,
- X_2 is `domestic` as described above, which is a dummy variable.

$$X_2 = \begin{cases} 1 & \text{Domestic} \\ 0 & \text{Foreign} \end{cases}$$

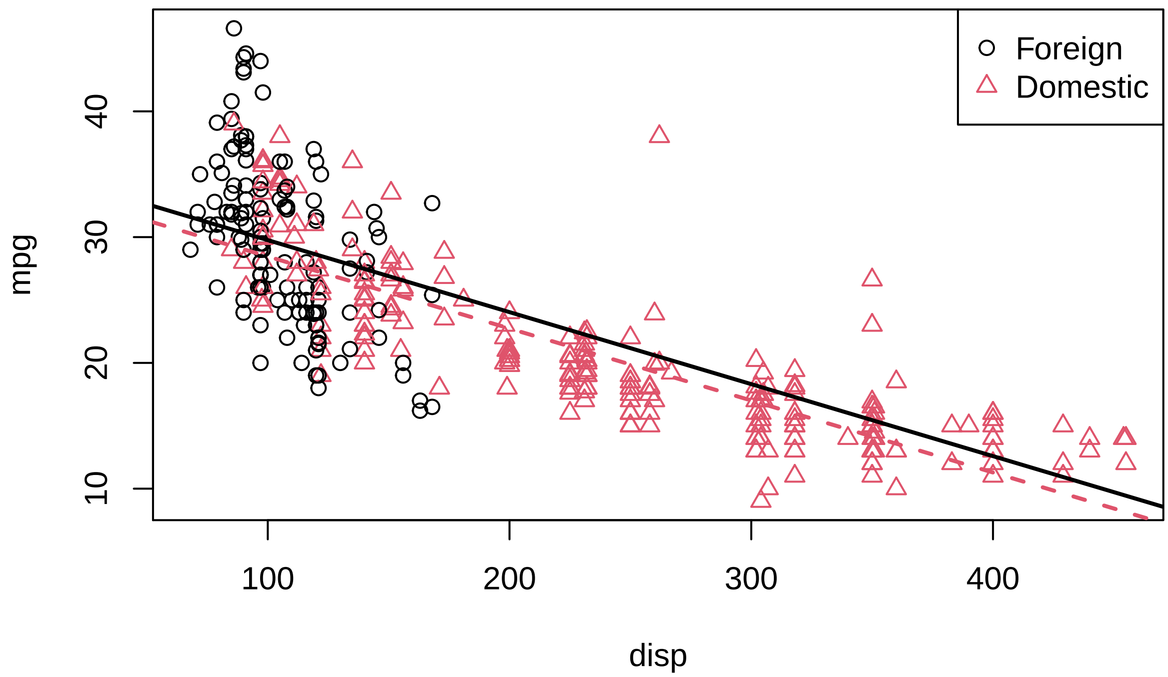
We will fit this model, extract the slope and intercept for the "two lines," plot the data and add the lines.

```r
mpg_disp_add = lm(mpg ~ disp + domestic, data = autompg)

int_for = coef(mpg_disp_add)[1]
int_dom = coef(mpg_disp_add)[1] + coef(mpg_disp_add)[3]

slope_for = coef(mpg_disp_add)[2]
slope_dom = coef(mpg_disp_add)[2]

plot(mpg ~ disp, data = autompg, col = domestic + 1, pch = domestic + 1)
abline(int_for, slope_for, col = 1, lty = 1, lwd = 2) # add line for foreign cars
abline(int_dom, slope_dom, col = 2, lty = 2, lwd = 2) # add line for domestic cars
legend("topright", c("Foreign", "Domestic"), pch = c(1, 2), col = c(1, 2))
```



This is a model that allows for two *parallel* lines, meaning the `mpg` can be different on average between foreign and domestic cars of the same engine displacement, but the change in average `mpg` for an increase in displacement is the same for both. We can see this model isn't doing very well here. The red line fits the red points fairly well, but the black line isn't doing very well for the black points, it should clearly have a more negative slope. Essentially, we would like a model that allows for two different slopes.

Consider the following model,

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon,$$

where X_1, X_2, and Y are the same as before, but we have added a new **interaction** term $X_1 X_2$ which multiplies X_1 and X_2, so we also have an additional β parameter β_3.

This model essentially creates two slopes and two intercepts, β_2 being the difference in intercepts and β_3 being the difference in slopes. To see this, we will break down the model into the two "sub-models" for foreign and domestic cars.

For foreign cars, that is $X_2 = 0$, we have

$$Y = \beta_0 + \beta_1 X_1 + \epsilon.$$

For domestic cars, that is $X_2 = 1$, we have

$$Y = (\beta_0 + \beta_2) + (\beta_1 + \beta_3) X_1 + \epsilon.$$

These two models have both different slopes and intercepts.

- β_0 is the average `mpg` for a foreign car with **0** `disp`.
- β_1 is the change in average `mpg` for an increase of one `disp`, for **foreign** cars.
- $\beta_0 + \beta_2$ is the average `mpg` for a domestic car with **0** `disp`.
- $\beta_1 + \beta_3$ is the change in average `mpg` for an increase of one `disp`, for **domestic** cars.

How do we fit this model in R? There are a number of ways.

One method would be to simply create a new variable, then fit a model like any other.

```
autompg$x3 = autompg$disp * autompg$domestic # THIS CODE NOT RUN!
do_not_do_this = lm(mpg ~ disp + domestic + x3, data = autompg) # THIS CODE NOT RUN!
```

You should only do this as a last resort. We greatly prefer not to have to modify our data simply to fit a model. Instead, we can tell R we would like to use the existing data with an interaction term, which it will create automatically when we use the `:` operator.

```
mpg_disp_int = lm(mpg ~ disp + domestic + disp:domestic, data = autompg)
```

An alternative method, which will fit the exact same model as above would be to use the `*` operator. This method automatically creates the interaction term, as well as any "lower order terms," which in this case are the first order terms for `disp` and `domestic`

```
mpg_disp_int2 = lm(mpg ~ disp * domestic, data = autompg)
```

We can quickly verify that these are doing the same thing.

```
coef(mpg_disp_int)
```

```
##   (Intercept)          disp      domestic disp:domestic
##    46.0548423    -0.1569239   -12.5754714     0.1025184
```

```
coef(mpg_disp_int2)
```

```
##   (Intercept)          disp      domestic disp:domestic
##    46.0548423    -0.1569239   -12.5754714     0.1025184
```

We see that both the variables, and their coefficient estimates are indeed the same for both models.

```
summary(mpg_disp_int)
```

```
##
## Call:
## lm(formula = mpg ~ disp + domestic + disp:domestic, data = autompg)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.8332  -2.8956  -0.8332   2.2828  18.7749
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    46.05484    1.80582  25.504  < 2e-16 ***
## disp           -0.15692    0.01668  -9.407  < 2e-16 ***
## domestic      -12.57547    1.95644  -6.428 3.90e-10 ***
## disp:domestic   0.10252    0.01692   6.060 3.29e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.308 on 379 degrees of freedom
## Multiple R-squared:  0.7011, Adjusted R-squared:  0.6987
## F-statistic: 296.3 on 3 and 379 DF,  p-value: < 2.2e-16
```

We see that using `summary()` gives the usual output for a multiple regression model. We pay close attention to the row for `disp:domestic` which tests,

$$H_0 : \beta_3 = 0.$$

In this case, testing for $\beta_3 = 0$ is testing for two lines with parallel slopes versus two lines with possibly different slopes. The `disp:domestic` line in the `summary()` output uses a t-test to perform the test.

We could also use an ANOVA F-test. The additive model, without interaction is our null model, and the interaction model is the alternative.

```
anova(mpg_disp_add, mpg_disp_int)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ disp + domestic
## Model 2: mpg ~ disp + domestic + disp:domestic
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    380 7714.0
## 2    379 7032.6  1    681.36 36.719 3.294e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

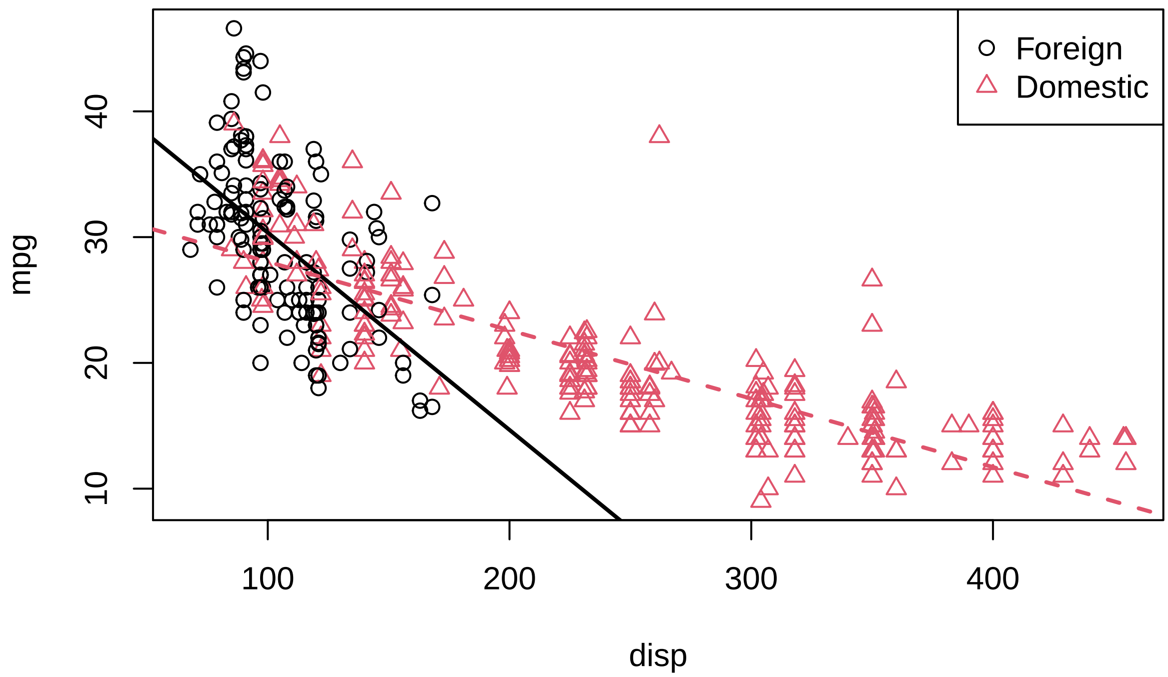
Again we see this test has the same p-value as the t-test. Also the p-value is extremely low, so between the two, we choose the interaction model.

```
int_for = coef(mpg_disp_int)[1]
int_dom = coef(mpg_disp_int)[1] + coef(mpg_disp_int)[3]

slope_for = coef(mpg_disp_int)[2]
slope_dom = coef(mpg_disp_int)[2] + coef(mpg_disp_int)[4]
```

Here we again calculate the slope and intercepts for the two lines for use in plotting.

```
plot(mpg ~ disp, data = autompg, col = domestic + 1, pch = domestic + 1)
abline(int_for, slope_for, col = 1, lty = 1, lwd = 2) # line for foreign cars
abline(int_dom, slope_dom, col = 2, lty = 2, lwd = 2) # line for domestic cars
legend("topright", c("Foreign", "Domestic"), pch = c(1, 2), col = c(1, 2))
```



We see that these lines fit the data much better, which matches the result of our tests.

So far we have only seen interaction between a categorical variable (`domestic`) and a numerical variable (`disp`). While this is easy to visualize, since it allows for different slopes for two lines, it is not the only type of interaction we can use in a model. We can also consider interactions between two numerical variables.