Chapter 11

Categorical Predictors and Interactions

> "The greatest value of a picture is when it forces us to notice what we never expected to see."
>
> — **John Tukey**

After reading this chapter you will be able to:

- Include and interpret categorical variables in a linear regression model by way of dummy variables.
- Understand the implications of using a model with a categorical variable in two ways: levels serving as unique predictors versus levels serving as a comparison to a baseline.
- Construct and interpret linear regression models with interaction terms.
- Identify categorical variables in a data set and convert them into factor variables, if necessary, using R.

So far in each of our analyses, we have only used numeric variables as predictors. We have also only used *additive models*, meaning the effect any predictor had on the response was not dependent on the other predictors. In this chapter, we will remove both of these restrictions. We will fit models with categorical predictors, and use models that allow predictors to *interact*. The mathematics of multiple regression will remain largely unchanging, however, we will pay close attention to interpretation, as well as some difference in `R` usage.

11.1 Dummy Variables

For this chapter, we will briefly use the built in dataset `mtcars` before returning to our `autompg` dataset that we created in the last chapter. The `mtcars` dataset is somewhat smaller, so we'll quickly take a look at the entire dataset.

```
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
```

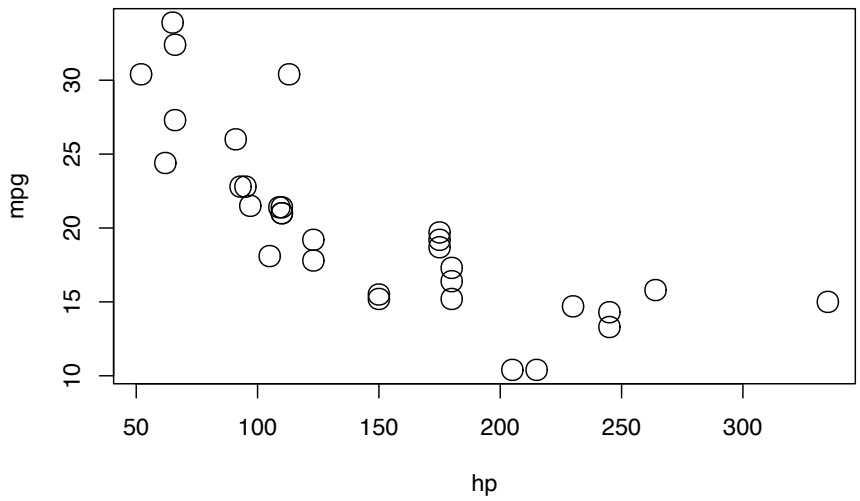
We will be interested in three of the variables: `mpg`, `hp`, and `am`.

- `mpg`: fuel efficiency, in miles per gallon.

- `hp`: horsepower, in foot-pounds per second.
- `am`: transmission. Automatic or manual.

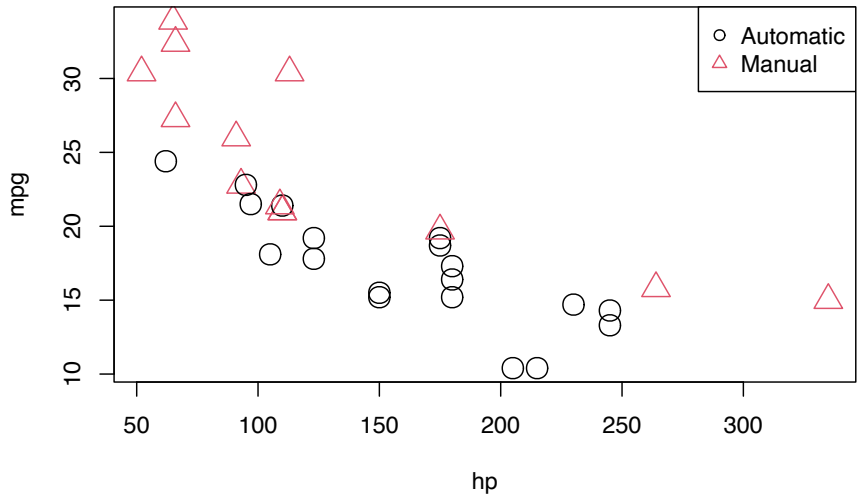
As we often do, we will start by plotting the data. We are interested in `mpg` as the response variable, and `hp` as a predictor.

```
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

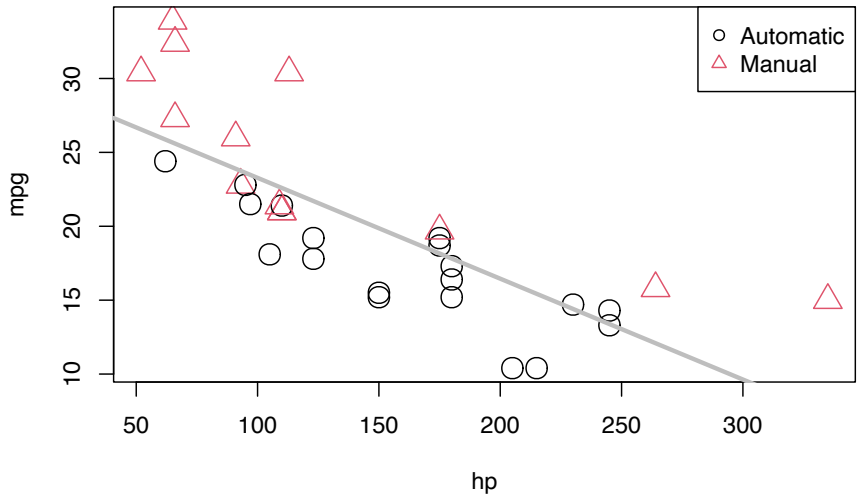
$$Y = \beta_0 + \beta_1 x_1 + \epsilon,$$

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

$$x_2 = \begin{cases} 1 & \text{manual transmission} \\ 0 & \text{automatic transmission} \end{cases}.$$

In this case, we call x_2 a **dummy variable**. A dummy variable is somewhat unfortunately named, as it is in no way "dumb". In fact, it is actually somewhat clever. A dummy variable is a numerical variable that is used in a regression analysis to "code" for a binary categorical variable. Let's see how this works.

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

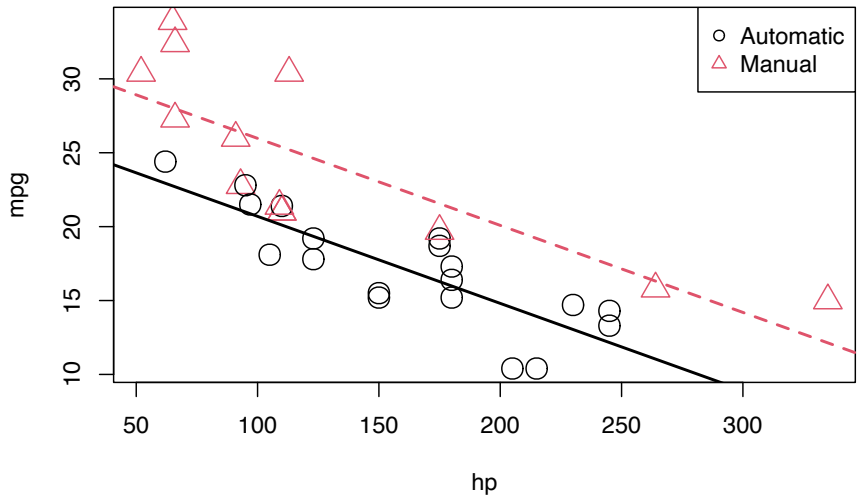
We can then combine these to calculate the estimated slope and intercepts.

```
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.

```
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

```
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

```
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.

We should take special notice of those last two. In the model,

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon,$$

we see β_1 is the average change in Y for an increase in x_1, *no matter* the value of x_2. Also, β_2 is always the difference in the average of Y for *any* value of x_1. These are two restrictions we won't always want, so we need a way to specify a more flexible model.

Here we restricted ourselves to a single numerical predictor x_1 and one dummy variable x_2. However, the concept of a dummy variable can be used with larger multiple regression models. We only use a single numerical predictor here for ease of visualization since we can think of the "two lines" interpretation. But in general, we can think of a dummy variable as creating "two models," one for each category of a binary categorical variable.

11.2 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$hp)
# create a dummy variable for foreign vs domestic cars. domestic = 1.
autompg$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 in the chapter and would not be done in practice. 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.

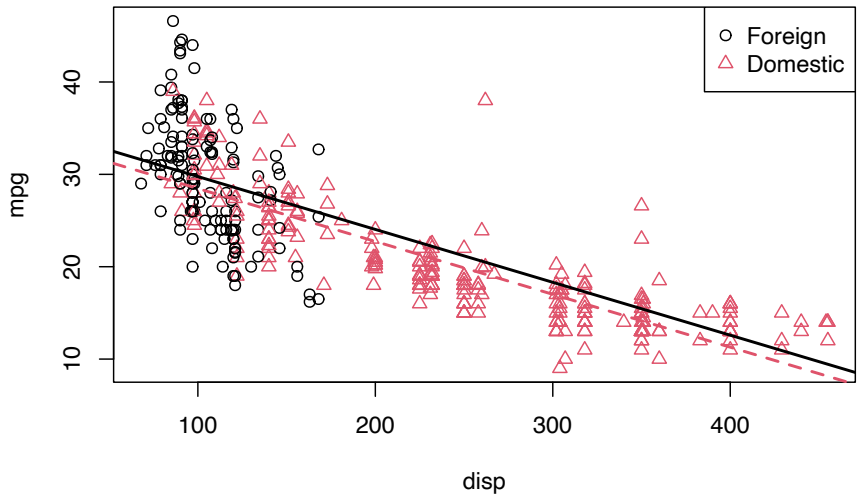
```
mpg_disp_add = lm(mpg ~ disp + domestic, data = autompg)

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
```

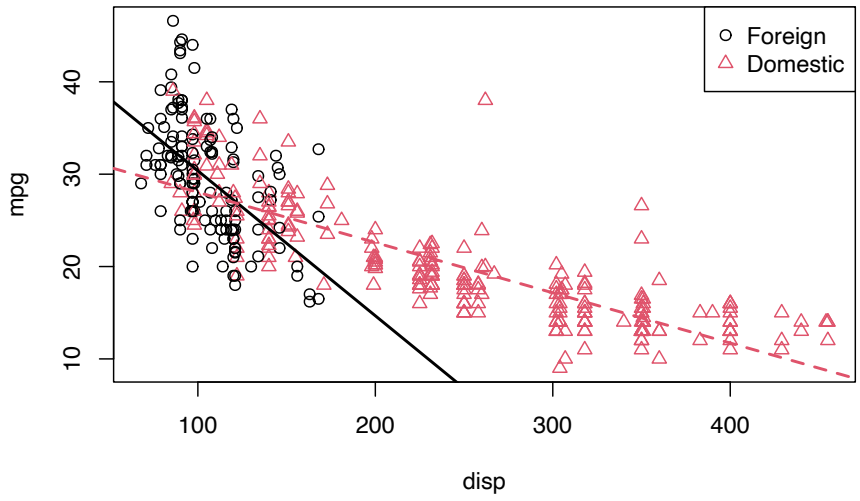
Again we see this test has the same p-value as the t-test. Also the p-value is extremely low, so between the two, we choose the interaction model.

```
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.

Consider the model,

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \epsilon,$$

where

- Y is `mpg`, the fuel efficiency in miles per gallon,
- x_1 is `disp`, the displacement in cubic inches,
- x_2 is `hp`, the horsepower, in foot-pounds per second.

How does `mpg` change based on `disp` in this model? We can rearrange some terms to see how.

$$Y = \beta_0 + (\beta_1 + \beta_3 x_2) x_1 + \beta_2 x_2 + \epsilon$$

So, for a one unit increase in x_1 (`disp`), the mean of Y (`mpg`) increases $\beta_1 + \beta_3 x_2$, which is a different value depending on the value of x_2 (`hp`)!

Since we're now working in three dimensions, this model can't be easily justified via visualizations like the previous example. Instead, we will have to rely on a test.

```
mpg_disp_add_hp = lm(mpg ~ disp + hp, data = autompg)
mpg_disp_int_hp = lm(mpg ~ disp * hp, data = autompg)
summary(mpg_disp_int_hp)
```

```
##
## Call:
## lm(formula = mpg ~ disp * hp, data = autompg)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.7849  -2.3104  -0.5699   2.1453  17.9211
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.241e+01  1.523e+00   34.42   <2e-16 ***
## disp        -1.002e-01  6.638e-03  -15.09   <2e-16 ***
## hp          -2.198e-01  1.987e-02  -11.06   <2e-16 ***
## disp:hp      5.658e-04  5.165e-05   10.96   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.896 on 379 degrees of freedom
## Multiple R-squared:  0.7554, Adjusted R-squared:  0.7535
## F-statistic: 390.2 on 3 and 379 DF,  p-value: < 2.2e-16
```

Using `summary()` we focus on the row for `disp:hp` which tests,

$$H_0 : \beta_3 = 0.$$

Again, we see a very low p-value so we reject the null (additive model) in favor of the interaction model. Again, there is an equivalent F-test.

```
anova(mpg_disp_add_hp, mpg_disp_int_hp)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ disp + hp
## Model 2: mpg ~ disp * hp
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    380 7576.6
```

```
## 2    379 5754.2  1    1822.3 120.03 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can take a closer look at the coefficients of our fitted interaction model.

```
coef(mpg_disp_int_hp)
```

```
##   (Intercept)          disp            hp       disp:hp
## 52.4081997848 -0.1001737655 -0.2198199720  0.0005658269
```

- $\hat{\beta}_0 = 52.4081998$ is the estimated average `mpg` for a car with 0 `disp` and 0 `hp`.
- $\hat{\beta}_1 = -0.1001738$ is the estimated change in average `mpg` for an increase in 1 `disp`, **for a car with 0 `hp`**.
- $\hat{\beta}_2 = -0.21982$ is the estimated change in average `mpg` for an increase in 1 `hp`, **for a car with 0 `disp`**.
- $\hat{\beta}_3 = 5.658269 \times 10^{-4}$ is an estimate of the modification to the change in average `mpg` for an increase in `disp`, for a car of a certain `hp` (or vice versa).

That last coefficient needs further explanation. Recall the rearrangement we made earlier

$$Y = \beta_0 + (\beta_1 + \beta_3 x_2)x_1 + \beta_2 x_2 + \epsilon.$$

So, our estimate for $\beta_1 + \beta_3 x_2$, is $\hat{\beta}_1 + \hat{\beta}_3 x_2$, which in this case is

$$-0.1001738 + 5.658269 \times 10^{-4} x_2.$$

This says that, for an increase of one `disp` we see an estimated change in average `mpg` of $-0.1001738 + 5.658269 \times 10^{-4} x_2$. So how `disp` and `mpg` are related, depends on the `hp` of the car.

So for a car with 50 `hp`, the estimated change in average `mpg` for an increase of one `disp` is

$$-0.1001738 + 5.658269 \times 10^{-4} \cdot 50 = -0.0718824$$

And for a car with 350 `hp`, the estimated change in average `mpg` for an increase of one `disp` is

$$-0.1001738 + 5.658269 \times 10^{-4} \cdot 350 = 0.0978657$$

Notice the sign changed!

11.3 Factor Variables

So far in this chapter, we have limited our use of categorical variables to binary categorical variables. Specifically, we have limited ourselves to dummy variables which take a value of `0` or `1` and represent a categorical variable numerically.

We will now discuss **factor** variables, which is a special way that `R` deals with categorical variables. With factor variables, a human user can simply think about the categories of a variable, and `R` will take care of the necessary dummy variables without any 0/1 assignment being done by the user.

```
is.factor(autompg$domestic)
```

```
## [1] FALSE
```

Earlier when we used the `domestic` variable, it was **not** a factor variable. It was simply a numerical variable that only took two possible values, `1` for domestic, and `0` for foreign. Let's create a new variable `origin` that stores the same information, but in a different way.

```
autompg$origin[autompg$domestic == 1] = "domestic"
autompg$origin[autompg$domestic == 0] = "foreign"
head(autompg$origin)
```

```
## [1] "domestic" "domestic" "domestic" "domestic" "domestic" "domestic"
```

Now the `origin` variable stores `"domestic"` for domestic cars and `"foreign"` for foreign cars.

```
is.factor(autompg$origin)
```

```
## [1] FALSE
```

However, this is simply a vector of character values. A vector of car models is a character variable in `R`. A vector of Vehicle Identification Numbers (VINs) is a character variable as well. But those don't represent a short list of levels that might influence a response variable. We will want to **coerce** this origin variable to be something more: a factor variable.

```
autompg$origin = as.factor(autompg$origin)
```

Now when we check the structure of the `autompg` dataset, we see that `origin` is a factor variable.

```
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  : Factor w/ 2 levels "domestic","foreign": 1 1 1 1 1 1 1 1 1 1 ...
##  $ domestic: num  1 1 1 1 1 1 1 1 1 1 ...
```

Factor variables have **levels** which are the possible values (categories) that the variable may take, in this case foreign or domestic.

```
levels(autompg$origin)
```

```
## [1] "domestic" "foreign"
```

Recall that previously we have fit the model

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \epsilon,$$

where

- Y is `mpg`, the fuel efficiency in miles per gallon,
- x_1 is `disp`, the displacement in cubic inches,
- x_2 is `domestic` a dummy variable where `1` indicates a domestic car.

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

```
##
## Call:
## lm(formula = mpg ~ disp * domestic, data = autompg)
##
## Coefficients:
##   (Intercept)           disp       domestic  disp:domestic
##       46.0548        -0.1569       -12.5755         0.1025
```

So here we see that

$$\hat{\beta}_0 + \hat{\beta}_2 = 46.0548423 + -12.5754714 = 33.4793709$$

is the estimated average `mpg` for a **domestic** car with 0 `disp`.

Now let's try to do the same, but using our new factor variable.

```
(mod_factor = lm(mpg ~ disp * origin, data = autompg))
```

```
##
## Call:
## lm(formula = mpg ~ disp * origin, data = autompg)
##
## Coefficients:
##        (Intercept)                disp       originforeign  disp:originforeign
##           33.47937            -0.05441            12.57547            -0.10252
```

It seems that it doesn't produce the same results. Right away we notice that the intercept is different, as is the the coefficient in front of `disp`. We also notice that the remaining two coefficients are of the same magnitude as their respective counterparts using the domestic variable, but with a different sign. Why is this happening?

It turns out, that by using a factor variable, `R` is automatically creating a dummy variable for us. However, it is not the dummy variable that we had originally used ourselves.

`R` is fitting the model

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \epsilon,$$

where

- Y is `mpg`, the fuel efficiency in miles per gallon,
- x_1 is `disp`, the displacement in cubic inches,
- x_2 **is a dummy variable created by `R`.** It uses `1` to represent a **foreign car**.

So now,

$$\hat{\beta}_0 = 33.4793709$$

is the estimated average `mpg` for a **domestic** car with 0 `disp`, which is indeed the same as before.

When R created x_2, the dummy variable, it used domestic cars as the **reference** level, that is the default value of the factor variable. So when the dummy variable is 0, the model represents this reference level, which is domestic. (R makes this choice because domestic comes before foreign alphabetically.)

So the two models have different estimated coefficients, but due to the different model representations, they are actually the same model.

11.3.1 Factors with More Than Two Levels

Let's now consider a factor variable with more than two levels. In this dataset, cyl is an example.

```
is.factor(autompg$cyl)
```

```
## [1] TRUE
```

```
levels(autompg$cyl)
```

```
## [1] "4" "6" "8"
```

Here the cyl variable has three possible levels: 4, 6, and 8. You may wonder, why not simply use cyl as a numerical variable? You certainly could.

However, that would force the difference in average mpg between 4 and 6 cylinders to be the same as the difference in average mpg between 6 and 8 cylinders. That usually make senses for a continuous variable, but not for a discrete variable with so few possible values. In the case of this variable, there is no such thing as a 7-cylinder engine or a 6.23-cylinder engine in personal vehicles. For these reasons, we will simply consider cyl to be categorical. This is a decision that will commonly need to be made with ordinal variables. Often, with a large number of categories, the decision to treat them as numerical variables is appropriate because, otherwise, a large number of dummy variables are then needed to represent these variables.

Let's define three dummy variables related to the cyl factor variable.

$$v_1 = \begin{cases} 1 & \text{4 cylinder} \\ 0 & \text{not 4 cylinder} \end{cases}$$

$$v_2 = \begin{cases} 1 & \text{6 cylinder} \\ 0 & \text{not 6 cylinder} \end{cases}$$

$$v_3 = \begin{cases} 1 & 8 \text{ cylinder} \\ 0 & \text{not } 8 \text{ cylinder} \end{cases}$$

Now, let's fit an additive model in `R`, using `mpg` as the response, and `disp` and `cyl` as predictors. This should be a model that uses "three regression lines" to model `mpg`, one for each of the possible `cyl` levels. They will all have the same slope (since it is an additive model), but each will have its own intercept.

```
(mpg_disp_add_cyl = lm(mpg ~ disp + cyl, data = autompg))
```

```
##
## Call:
## lm(formula = mpg ~ disp + cyl, data = autompg)
##
## Coefficients:
## (Intercept)         disp         cyl6         cyl8
##    34.99929     -0.05217     -3.63325     -2.03603
```

The question is, what is the model that `R` has fit here? It has chosen to use the model

$$Y = \beta_0 + \beta_1 x + \beta_2 v_2 + \beta_3 v_3 + \epsilon,$$

where

- Y is `mpg`, the fuel efficiency in miles per gallon,
- x is `disp`, the displacement in cubic inches,
- v_2 and v_3 are the dummy variables define above.

Why doesn't `R` use v_1? Essentially because it doesn't need to. To create three lines, it only needs two dummy variables since it is using a reference level, which in this case is a 4 cylinder car. The three "sub models" are then:

- 4 Cylinder: $Y = \beta_0 + \beta_1 x + \epsilon$
- 6 Cylinder: $Y = (\beta_0 + \beta_2) + \beta_1 x + \epsilon$
- 8 Cylinder: $Y = (\beta_0 + \beta_3) + \beta_1 x + \epsilon$

Notice that they all have the same slope. However, using the two dummy variables, we achieve the three intercepts.

- β_0 is the average `mpg` for a 4 cylinder car with 0 `disp`.
- $\beta_0 + \beta_2$ is the average `mpg` for a 6 cylinder car with 0 `disp`.

- $\beta_0 + \beta_3$ is the average `mpg` for a 8 cylinder car with 0 `disp`.

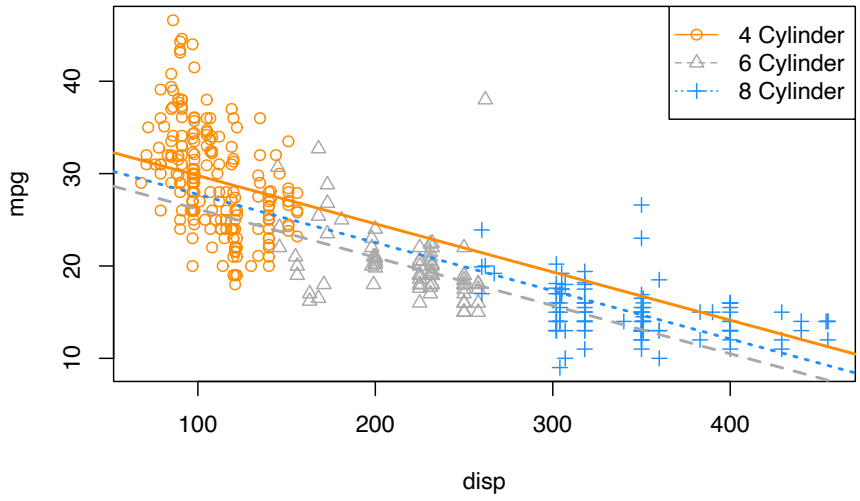
So because 4 cylinder is the reference level, β_0 is specific to 4 cylinders, but β_2 and β_3 are used to represent quantities relative to 4 cylinders.

As we have done before, we can extract these intercepts and slopes for the three lines, and plot them accordingly.

```
int_4cyl = coef(mpg_disp_add_cyl)[1]
int_6cyl = coef(mpg_disp_add_cyl)[1] + coef(mpg_disp_add_cyl)[3]
int_8cyl = coef(mpg_disp_add_cyl)[1] + coef(mpg_disp_add_cyl)[4]

slope_all_cyl = coef(mpg_disp_add_cyl)[2]

plot_colors = c("Darkorange", "Darkgrey", "Dodgerblue")
plot(mpg ~ disp, data = autompg, col = plot_colors[cyl], pch = as.numeric(cyl))
abline(int_4cyl, slope_all_cyl, col = plot_colors[1], lty = 1, lwd = 2)
abline(int_6cyl, slope_all_cyl, col = plot_colors[2], lty = 2, lwd = 2)
abline(int_8cyl, slope_all_cyl, col = plot_colors[3], lty = 3, lwd = 2)
legend("topright", c("4 Cylinder", "6 Cylinder", "8 Cylinder"),
       col = plot_colors, lty = c(1, 2, 3), pch = c(1, 2, 3))
```



On this plot, we have

- 4 Cylinder: orange dots, solid orange line.

- 6 Cylinder: grey dots, dashed grey line.
- 8 Cylinder: blue dots, dotted blue line.

The odd result here is that we're estimating that 8 cylinder cars have better fuel efficiency than 6 cylinder cars at **any** displacement! The dotted blue line is always above the dashed grey line. That doesn't seem right. Maybe for very large displacement engines that could be true, but that seems wrong for medium to low displacement.

To attempt to fix this, we will try using an interaction model, that is, instead of simply three intercepts and one slope, we will allow for three slopes. Again, we'll let `R` take the wheel (no pun intended), then figure out what model it has applied.

```
(mpg_disp_int_cyl = lm(mpg ~ disp * cyl, data = autompg))
```

```
##
## Call:
## lm(formula = mpg ~ disp * cyl, data = autompg)
##
## Coefficients:
## (Intercept)         disp         cyl6         cyl8    disp:cyl6    disp:cyl8
##    43.59052     -0.13069    -13.20026    -20.85706      0.08299      0.10817
```

```
# could also use mpg ~ disp + cyl + disp:cyl
```

`R` has again chosen to use 4 cylinder cars as the reference level, but this also now has an effect on the interaction terms. `R` has fit the model.

$$Y = \beta_0 + \beta_1 x + \beta_2 v_2 + \beta_3 v_3 + \gamma_2 x v_2 + \gamma_3 x v_3 + \epsilon$$

We're using γ like a β parameter for simplicity, so that, for example β_2 and γ_2 are both associated with v_2.

Now, the three "sub models" are:

- 4 Cylinder: $Y = \beta_0 + \beta_1 x + \epsilon$.
- 6 Cylinder: $Y = (\beta_0 + \beta_2) + (\beta_1 + \gamma_2) x + \epsilon$.
- 8 Cylinder: $Y = (\beta_0 + \beta_3) + (\beta_1 + \gamma_3) x + \epsilon$.

Interpreting some parameters and coefficients then:

- $(\beta_0 + \beta_2)$ is the average `mpg` of a 6 cylinder car with 0 `disp`

- $(\hat{\beta}_1 + \hat{\gamma}_3) = -0.1306935 + 0.1081714 = -0.0225221$ is the estimated change in average `mpg` for an increase of one `disp`, for an 8 cylinder car.

So, as we have seen before β_2 and β_3 change the intercepts for 6 and 8 cylinder cars relative to the reference level of β_0 for 4 cylinder cars.

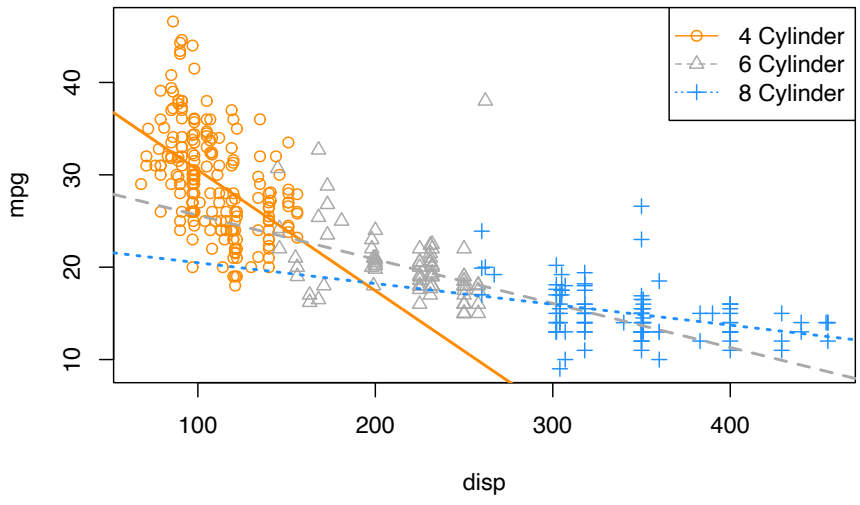
Now, similarly γ_2 and γ_3 change the slopes for 6 and 8 cylinder cars relative to the reference level of β_1 for 4 cylinder cars.

Once again, we extract the coefficients and plot the results.

```
int_4cyl = coef(mpg_disp_int_cyl)[1]
int_6cyl = coef(mpg_disp_int_cyl)[1] + coef(mpg_disp_int_cyl)[3]
int_8cyl = coef(mpg_disp_int_cyl)[1] + coef(mpg_disp_int_cyl)[4]

slope_4cyl = coef(mpg_disp_int_cyl)[2]
slope_6cyl = coef(mpg_disp_int_cyl)[2] + coef(mpg_disp_int_cyl)[5]
slope_8cyl = coef(mpg_disp_int_cyl)[2] + coef(mpg_disp_int_cyl)[6]

plot_colors = c("Darkorange", "Darkgrey", "Dodgerblue")
plot(mpg ~ disp, data = autompg, col = plot_colors[cyl], pch = as.numeric(cyl))
abline(int_4cyl, slope_4cyl, col = plot_colors[1], lty = 1, lwd = 2)
abline(int_6cyl, slope_6cyl, col = plot_colors[2], lty = 2, lwd = 2)
abline(int_8cyl, slope_8cyl, col = plot_colors[3], lty = 3, lwd = 2)
legend("topright", c("4 Cylinder", "6 Cylinder", "8 Cylinder"),
       col = plot_colors, lty = c(1, 2, 3), pch = c(1, 2, 3))
```



This looks much better! We can see that for medium displacement cars, 6 cylinder cars now perform better than 8 cylinder cars, which seems much more reasonable than before.

To completely justify the interaction model (i.e., a unique slope for each `cyl` level) compared to the additive model (single slope), we can perform an F-test. Notice first, that there is no t-test that will be able to do this since the difference between the two models is not a single parameter.

We will test,

$$H_0 : \gamma_2 = \gamma_3 = 0$$

which represents the parallel regression lines we saw before,

$$Y = \beta_0 + \beta_1 x + \beta_2 v_2 + \beta_3 v_3 + \epsilon.$$

Again, this is a difference of two parameters, thus no t-test will be useful.

```
anova(mpg_disp_add_cyl, mpg_disp_int_cyl)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ disp + cyl
## Model 2: mpg ~ disp * cyl
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    379 7299.5
## 2    377 6551.7  2    747.79 21.515 1.419e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As expected, we see a very low p-value, and thus reject the null. We prefer the interaction model over the additive model.

Recapping a bit:

- Null Model: $Y = \beta_0 + \beta_1 x + \beta_2 v_2 + \beta_3 v_3 + \epsilon$
  - Number of parameters: $q = 4$
- Full Model: $Y = \beta_0 + \beta_1 x + \beta_2 v_2 + \beta_3 v_3 + \gamma_2 x v_2 + \gamma_3 x v_3 + \epsilon$
  - Number of parameters: $p = 6$

```
length(coef(mpg_disp_int_cyl)) - length(coef(mpg_disp_add_cyl))
```

```
## [1] 2
```

We see there is a difference of two parameters, which is also displayed in the resulting ANOVA table from R. Notice that the following two values also appear on the ANOVA table.

```
nrow(autompg) - length(coef(mpg_disp_int_cyl))
```

```
## [1] 377
```

```
nrow(autompg) - length(coef(mpg_disp_add_cyl))
```

```
## [1] 379
```

11.4 Parameterization

So far we have been simply letting R decide how to create the dummy variables, and thus R has been deciding the parameterization of the models. To illustrate the ability to use alternative parameterizations, we will recreate the data, but directly creating the dummy variables ourselves.

```
new_param_data = data.frame(
  y = autompg$mpg,
  x = autompg$disp,
  v1 = 1 * as.numeric(autompg$cyl == 4),
  v2 = 1 * as.numeric(autompg$cyl == 6),
  v3 = 1 * as.numeric(autompg$cyl == 8))

head(new_param_data, 20)
```

```
##     y   x v1 v2 v3
## 1  18 307  0  0  1
## 2  15 350  0  0  1
## 3  18 318  0  0  1
## 4  16 304  0  0  1
## 5  17 302  0  0  1
## 6  15 429  0  0  1
## 7  14 454  0  0  1
## 8  14 440  0  0  1
## 9  14 455  0  0  1
## 10 15 390  0  0  1
## 11 15 383  0  0  1
```

```
## 12 14 340  0  0  1
## 13 15 400  0  0  1
## 14 14 455  0  0  1
## 15 24 113  1  0  0
## 16 22 198  0  1  0
## 17 18 199  0  1  0
## 18 21 200  0  1  0
## 19 27  97  1  0  0
## 20 26  97  1  0  0
```

Now,

- `y` is `mpg`
- `x` is `disp`, the displacement in cubic inches,
- `v1`, `v2`, and `v3` are dummy variables as defined above.

First let's try to fit an additive model using `x` as well as the three dummy variables.

```
lm(y ~ x + v1 + v2 + v3, data = new_param_data)
```

```
##
## Call:
## lm(formula = y ~ x + v1 + v2 + v3, data = new_param_data)
##
## Coefficients:
## (Intercept)            x           v1           v2           v3
##    32.96326     -0.05217      2.03603     -1.59722           NA
```

What is happening here? Notice that `R` is essentially ignoring `v3`, but why? Well, because `R` uses an intercept, it cannot also use `v3`. This is because

$$1 = v_1 + v_2 + v_3$$

which means that 1, v_1, v_2, and v_3 are linearly dependent. This would make the $X^\top X$ matrix singular, but we need to be able to invert it to solve the normal equations and obtain $\widehat{\beta}$. With the intercept, `v1`, and `v2`, `R` can make the necessary "three intercepts". So, in this case `v3` is the reference level.

If we remove the intercept, then we can directly obtain all "three intercepts" without a reference level.

```
lm(y ~ 0 + x + v1 + v2 + v3, data = new_param_data)
```

```
##
## Call:
## lm(formula = y ~ 0 + x + v1 + v2 + v3, data = new_param_data)
##
## Coefficients:
##        x        v1        v2        v3
## -0.05217  34.99929  31.36604  32.96326
```

Here, we are fitting the model

$$Y = \mu_1 v_1 + \mu_2 v_2 + \mu_3 v_3 + \beta x + \epsilon.$$

Thus we have:

- 4 Cylinder: $Y = \mu_1 + \beta x + \epsilon$
- 6 Cylinder: $Y = \mu_2 + \beta x + \epsilon$
- 8 Cylinder: $Y = \mu_3 + \beta x + \epsilon$

We could also do something similar with the interaction model, and give each line an intercept and slope, without the need for a reference level.

```
lm(y ~ 0 + v1 + v2 + v3 + x:v1 + x:v2 + x:v3, data = new_param_data)
```

```
##
## Call:
## lm(formula = y ~ 0 + v1 + v2 + v3 + x:v1 + x:v2 + x:v3, data = new_param_data)
##
## Coefficients:
##       v1        v2        v3      v1:x      v2:x      v3:x
## 43.59052  30.39026  22.73346  -0.13069  -0.04770  -0.02252
```

$$Y = \mu_1 v_1 + \mu_2 v_2 + \mu_3 v_3 + \beta_1 x v_1 + \beta_2 x v_2 + \beta_3 x v_3 + \epsilon$$

- 4 Cylinder: $Y = \mu_1 + \beta_1 x + \epsilon$
- 6 Cylinder: $Y = \mu_2 + \beta_2 x + \epsilon$
- 8 Cylinder: $Y = \mu_3 + \beta_3 x + \epsilon$

Using the original data, we have (at least) three equivalent ways to specify the interaction model with `R`.

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

```
##
## Call:
## lm(formula = mpg ~ disp * cyl, data = autompg)
##
## Coefficients:
## (Intercept)         disp         cyl6         cyl8    disp:cyl6    disp:cyl8
##    43.59052     -0.13069    -13.20026    -20.85706      0.08299      0.10817
```

```
lm(mpg ~ 0 + cyl + disp : cyl, data = autompg)
```

```
##
## Call:
## lm(formula = mpg ~ 0 + cyl + disp:cyl, data = autompg)
##
## Coefficients:
##      cyl4       cyl6       cyl8  cyl4:disp  cyl6:disp  cyl8:disp
##  43.59052   30.39026   22.73346   -0.13069   -0.04770   -0.02252
```

```
lm(mpg ~ 0 + disp + cyl + disp : cyl, data = autompg)
```

```
##
## Call:
## lm(formula = mpg ~ 0 + disp + cyl + disp:cyl, data = autompg)
##
## Coefficients:
##      disp       cyl4       cyl6       cyl8  disp:cyl6  disp:cyl8
##  -0.13069   43.59052   30.39026   22.73346    0.08299    0.10817
```

They all fit the same model, importantly each using six parameters, but the coefficients mean slightly different things in each. However, once they are interpreted as slopes and intercepts for the "three lines" they will have the same result.

Use `?all.equal` to learn about the `all.equal()` function, and think about how the following code verifies that the residuals of the two models are the same.

```
all.equal(fitted(lm(mpg ~ disp * cyl, data = autompg)),
          fitted(lm(mpg ~ 0 + cyl + disp : cyl, data = autompg)))
```

```
## [1] TRUE
```

11.5 Building Larger Models

Now that we have seen how to incorporate categorical predictors as well as interaction terms, we can start to build much larger, much more flexible models which can potentially fit data better.

Let's define a "big" model,

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_1 x_2 + \beta_5 x_1 x_3 + \beta_6 x_2 x_3 + \beta_7 x_1 x_2 x_3 + \epsilon.$$

Here,

- Y is `mpg`.
- x_1 is `disp`.
- x_2 is `hp`.
- x_3 is `domestic`, which is a dummy variable we defined, where `1` is a domestic vehicle.

First thing to note here, we have included a new term $x_1 x_2 x_3$ which is a three-way interaction. Interaction terms can be larger and larger, up to the number of predictors in the model.

Since we are using the three-way interaction term, we also use all possible two-way interactions, as well as each of the first order (**main effect**) terms. This is the concept of a **hierarchy**. Any time a "higher-order" term is in a model, the related "lower-order" terms should also be included. Mathematically their inclusion or exclusion is sometimes irrelevant, but from an interpretation standpoint, it is best to follow the hierarchy rules.

Let's do some rearrangement to obtain a "coefficient" in front of x_1.

$$Y = \beta_0 + \beta_2 x_2 + \beta_3 x_3 + \beta_6 x_2 x_3 + (\beta_1 + \beta_4 x_2 + \beta_5 x_3 + \beta_7 x_2 x_3) x_1 + \epsilon.$$

Specifically, the "coefficient" in front of x_1 is

$$(\beta_1 + \beta_4 x_2 + \beta_5 x_3 + \beta_7 x_2 x_3).$$

Let's discuss this "coefficient" to help us understand the idea of the *flexibility* of a model. Recall that,

- β_1 is the coefficient for a first order term,
- β_4 and β_5 are coefficients for two-way interactions,
- β_7 is the coefficient for the three-way interaction.

If the two and three way interactions were not in the model, the whole "coefficient" would simply be

$$\beta_1.$$

Thus, no matter the values of x_2 and x_3, β_1 would determine the relationship between x_1 (`disp`) and Y (`mpg`).

With the addition of the two-way interactions, now the "coefficient" would be

$$(\beta_1 + \beta_4 x_2 + \beta_5 x_3).$$

Now, changing x_1 (`disp`) has a different effect on Y (`mpg`), depending on the values of x_2 and x_3.

Lastly, adding the three-way interaction gives the whole "coefficient"

$$(\beta_1 + \beta_4 x_2 + \beta_5 x_3 + \beta_7 x_2 x_3)$$

which is even more flexible. Now changing x_1 (`disp`) has a different effect on Y (`mpg`), depending on the values of x_2 and x_3, but in a more flexible way which we can see with some more rearrangement. Now the "coefficient" in front of x_3 in this "coefficient" is dependent on x_2.

$$(\beta_1 + \beta_4 x_2 + (\beta_5 + \beta_7 x_2) x_3)$$

It is so flexible, it is becoming hard to interpret!

Let's fit this three-way interaction model in `R`.

```
big_model = lm(mpg ~ disp * hp * domestic, data = autompg)
summary(big_model)
```

```
##
## Call:
## lm(formula = mpg ~ disp * hp * domestic, data = autompg)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.9410  -2.2147  -0.4008   1.9430  18.4094
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       6.065e+01  6.600e+00   9.189  < 2e-16 ***
## disp             -1.416e-01  6.344e-02  -2.232   0.0262 *
```

```
## hp               -3.545e-01  8.123e-02  -4.364 1.65e-05 ***
## domestic         -1.257e+01  7.064e+00  -1.780   0.0759 .
## disp:hp           1.369e-03  6.727e-04   2.035   0.0426 *
## disp:domestic     4.933e-02  6.400e-02   0.771   0.4414
## hp:domestic       1.852e-01  8.709e-02   2.126   0.0342 *
## disp:hp:domestic -9.163e-04  6.768e-04  -1.354   0.1766
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.88 on 375 degrees of freedom
## Multiple R-squared:   0.76,  Adjusted R-squared:  0.7556
## F-statistic: 169.7 on 7 and 375 DF,  p-value: < 2.2e-16
```

Do we actually need this large of a model? Let's first test for the necessity of the three-way interaction term. That is,

$$H_0 : \beta_7 = 0.$$

So,

- Full Model: $Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_1 x_2 + \beta_5 x_1 x_3 + \beta_6 x_2 x_3 + \beta_7 x_1 x_2 x_3 + \epsilon$
- Null Model: $Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_1 x_2 + \beta_5 x_1 x_3 + \beta_6 x_2 x_3 + \epsilon$

We fit the null model in `R` as `two_way_int_mod`, then use `anova()` to perform an F-test as usual.

```
two_way_int_mod = lm(mpg ~ disp * hp + disp * domestic + hp * domestic, data = autompg)
#two_way_int_mod = lm(mpg ~ (disp + hp + domestic) ^ 2, data = autompg)
anova(two_way_int_mod, big_model)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ disp * hp + disp * domestic + hp * domestic
## Model 2: mpg ~ disp * hp * domestic
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    376 5673.2
## 2    375 5645.6  1    27.599 1.8332 0.1766
```

We see the p-value is somewhat large, so we would fail to reject. We prefer the smaller, less flexible, null model, without the three-way interaction.

A quick note here: the full model does still "fit better." Notice that it has a smaller RMSE than the null model, which means the full model makes smaller (squared) errors on average.

```
mean(resid(big_model) ^ 2)
```

```
## [1] 14.74053
```

```
mean(resid(two_way_int_mod) ^ 2)
```

```
## [1] 14.81259
```

However, it is not much smaller. We could even say that, the difference is insignificant. This is an idea we will return to later in greater detail.

Now that we have chosen the model without the three-way interaction, can we go further? Do we need the two-way interactions? Let's test

$$H_0 : \beta_4 = \beta_5 = \beta_6 = 0.$$

Remember we already chose $\beta_7 = 0$, so,

- Full Model: $Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_1 x_2 + \beta_5 x_1 x_3 + \beta_6 x_2 x_3 + \epsilon$
- Null Model: $Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon$

We fit the null model in `R` as `additive_mod`, then use `anova()` to perform an F-test as usual.

```
additive_mod = lm(mpg ~ disp + hp + domestic, data = autompg)
anova(additive_mod, two_way_int_mod)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ disp + hp + domestic
## Model 2: mpg ~ disp * hp + disp * domestic + hp * domestic
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    379 7369.7
## 2    376 5673.2  3    1696.5 37.478 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here the p-value is small, so we reject the null, and we prefer the full (alternative) model. Of the models we have considered, our final preference is for

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_1 x_2 + \beta_5 x_1 x_3 + \beta_6 x_2 x_3 + \epsilon.$$

11.6 R Markdown

The R Markdown file for this chapter can be found here:

- cat-int.Rmd

The file was created using R version 4.1.0.