Dealing with Dummy Variables

So far we've been working with ordinal or ratio predictor variables, but what if we want to use a nominal predictor variable?

For example consider the data below showing some employees' salaries in thousands of dollars, gender, and years of experience. What if we wanted to make a model using gender as one of the predictor variables?

```
  salary gender years
1     85   male     8
2     95 female    10
3    100   male    11
```

We cannot use it as it appears since "male"/"female" is not something we can plug into a linear model. But what if we decided to code gender by letting a 1 represent female, and a 0 represent male?

```
  salary gender years
1     85      0     8
2     95      1    10
3    100      0    11
```

Now we can create a linear model using these variables. This is how we can work with nominal variables when creating regression models.

A Clock Example

Imagine an antique clock dealer has collected data on recent auctions of grandfather clocks. The variables are `Price`: final selling price, `Bidders`: the number of bidders, `Age`: the age of the clock, `Temp`: the outside temperature the day of the auction, and `Condition`: the condition of the clock. (We used data similar to this in a previous tutorial. For this tutorial we've added the nominal variable `Condition`.)

```r
clocks <- read_csv("auctionExtra.csv");
```

```
Parsed with column specification:
cols(
  Age = col_double(),
  Bidders = col_double(),
  Price = col_double(),
  Temp = col_double(),
  Condition = col_character()
)
```

```r
head(clocks)
```

```
# A tibble: 6 x 5
    Age Bidders Price  Temp Condition
  <dbl>   <dbl> <dbl> <dbl> <chr>
1   127      13  1235    62 Good
2   115      12  1080    39 Good
3   127       7   845    53 Fair
4   150       9  1522    68 Excellent
5   156       6  1047    64 Good
6   182      11  1979    67 Excellent
```

Note when `R` imports this data set that `Condition` is a column containing characters rather than numbers. We can see that since `R` tells us that column is `col_character()`. A character column like this has some restrictions. For instance, R will return an error if we try to compute the mean of that column.

Let's see what entries are in that column. The following command lists all unique entries in the column.

```r
unique(clocks$Condition)
```

```
[1] "Good"      "Fair"      "Excellent"
```

There are three options in that column. This means we will need two dummy variables to indicate all possible options. Do you recall above that we needed one variable to represent the two options male/female? In general, when we have n options we will need $n - 1$ dummy variables.

Below is a table with dummy variables for `Condition`.

```
# A tibble: 6 x 7
    Age Bidders Price  Temp Condition Cond_Fair Cond_Good
  <dbl>   <dbl> <dbl> <dbl> <chr>         <dbl>     <dbl>
1   127      13  1235    62 Good              0         1
2   115      12  1080    39 Good              0         1
3   127       7   845    53 Fair              1         0
4   150       9  1522    68 Excellent         0         0
5   156       6  1047    64 Good              0         1
6   182      11  1979    67 Excellent         0         0
```

Notice the extra two columns? There is a column to indicate "Fair", and a column to indicate "Good". So if there is a 0 in both columns then that must mean the condition was "Excellent".

Now in fact we don't really need to generate these dummy variables by hand. The process for generating a model involving a nominal variable in R is very straightforward. Simply run the same `lm` command and R will detect the nominal predictor variable and create dummy variables for us automatically. We do not need to manually create a table like the one above.

Let's try to a model to predict price using all the other variables present. I will not use the dummy variables we made, instead I will just tell R to use the variable `Condition` and see what happens.

```r
attach(clocks)
summary(lm(Price ~ Age + Bidders + Temp + Condition))
```

```

Call:
lm(formula = Price ~ Age + Bidders + Temp + Condition)

Residuals:
    Min      1Q  Median      3Q     Max 
-186.51  -69.21  -17.86   70.93  225.48 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)   -399.05576  412.10010  -0.968  0.34179    
Age              8.95540    1.73896   5.150 2.26e-05 ***
Bidders         63.73439   13.29564   4.794 5.79e-05 ***
Temp            -0.09101    1.39354  -0.065  0.94843    
ConditionFair -294.26372  139.96903  -2.102  0.04536 *  
ConditionGood -252.05478   79.66895  -3.164  0.00394 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 116.5 on 26 degrees of freedom
Multiple R-squared:  0.9263,    Adjusted R-squared:  0.9122 
```

```
F-statistic: 65.39 on 5 and 26 DF,  p-value: 6.845e-14
```

Notice there are two variables for "Condition". `R` created dummy variables in the same way we did!

Using backwards elimination we would remove the `Temp` variable and have the resulting model.

```r
model <- lm(Price ~ Age + Bidders + Condition)
summary(model)
```

```
Call:
lm(formula = Price ~ Age + Bidders + Condition)

Residuals:
    Min      1Q  Median      3Q     Max 
-186.87  -69.19  -17.17   70.14  227.44 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)    -401.09     403.27  -0.995  0.32877    
Age               8.94       1.69   5.290 1.40e-05 ***
Bidders          63.66      13.00   4.898 4.01e-05 ***
ConditionFair  -295.63     135.83  -2.176  0.03844 *  
ConditionGood  -253.31      75.87  -3.339  0.00247 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 114.3 on 27 degrees of freedom
Multiple R-squared:  0.9263,    Adjusted R-squared:  0.9154 
F-statistic: 84.86 on 4 and 27 DF,  p-value: 6.914e-15
```

So the equation from this model is

$$Price = -401.09 + 8.94 * Age + 63.66 * Bidders - 295.63 * ConditionFair - 253.31 * ConditionGood$$

In effect this gives us three equations. If the clock is in fair condition then we will plug in 1 for ConditionFair, and 0 for ConditionGood. This gives the equation:

$$Price = -401.09 + 8.94 * Age + 63.66 * Bidders - 295.63 * 1$$

$$Price = -696.72 + 8.94 * Age + 63.66 * Bidders$$

Similarly, a clock that is in good condition will have a 1 for ConditionGood and 0 for ConditionFair giving

$$Price = -654.40 + 8.94 * Age + 63.66 * Bidders$$

And finally, a clock in excellent condition has a 0 for both dummy variables giving the equation

$$Price = -401.09 + 8.94 * Age + 63.66 * Bidders$$

Did you notice that the only thing really changing is the constant term? The coefficents on the other variables stay the same. So the rate at which `Age` affects the price will be the same for clocks in any condition.

EPA Data

The file "EPA gasoline rating 2019.csv"[1] contains data about 2019 model year vehicles collected by the Enivronmental Protection Agency along with the EPA miles per gallon fuel efficiency ration. This data set includes gasoline powered vehicles only. The accompanying text file includes descriptions of each variable.

In particular, I might like to know, which variables affect MPG rating? For the purposes of this example, let's focus on just a couple that seem most obvious. We'll use `Displ` since engine size should affect fuel economy, and we'll use `Veh Class` since that's a nominal variable, and it seems likely that will affect mpg as well.

```
epa <- read_csv("EPA gasoline rating 2019.csv")
```

```
Parsed with column specification:
cols(
  Model = col_character(),
  Displ = col_double(),
  Cyl = col_double(),
  Trans = col_character(),
  Drive = col_character(),
  `Cert Region` = col_character(),
  Stnd = col_character(),
  `Stnd Description` = col_character(),
  `Underhood ID` = col_character(),
  `Veh Class` = col_character(),
  `Air Pollution Score` = col_double(),
  `City MPG` = col_double(),
  `Hwy MPG` = col_double(),
  `Cmb MPG` = col_double(),
  `Greenhouse Gas Score` = col_double(),
  SmartWay = col_character(),
  `Comb CO2` = col_double()
)
```

```
attach(epa)
```

`Displ` is a numerical variable, so we can include that in our model as usual.

However, `Veh Class` is nominal, so we know `R` will need to create dummy variables for us. Let's see how many different classes there are.

```
unique(epa$`Veh Class`)
```

```
 [1] "small car"       "small SUV"       "midsize car"
 [4] "station wagon"   "large car"       "standard SUV"
 [7] "special purpose" "pickup"          "minivan"
[10] "van"
```

Wow, for `Veh Class` we have 10 choices so we'll need 9 dummy variables. (Do you see why?) Luckily `R` will take care of this for us.

We generate the linear model below.

```
summary(lm(`Cmb MPG` ~ Displ + `Veh Class`))
```

```
Call:
lm(formula = `Cmb MPG` ~ Displ + `Veh Class`)
```

[1] EPA (2019). Fuel Economy Data Set [Data File]. Accessed at https://www.fueleconomy.gov/feg/download.shtml

```
Residuals:
    Min      1Q  Median      3Q     Max
-8.1004 -2.3791 -0.4322  1.2878 29.5314

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)                33.44658    0.35288  94.781  < 2e-16 ***
Displ                      -3.11126    0.07015 -44.354  < 2e-16 ***
`Veh Class`midsize car      2.20816    0.33655   6.561 6.52e-11 ***
`Veh Class`minivan         -1.06978    1.04282  -1.026 0.305066
`Veh Class`pickup          -2.17479    0.39483  -5.508 4.01e-08 ***
`Veh Class`small car       -0.45678    0.30189  -1.513 0.130387
`Veh Class`small SUV       -1.84497    0.33102  -5.574 2.77e-08 ***
`Veh Class`special purpose -3.87295    0.59900  -6.466 1.22e-10 ***
`Veh Class`standard SUV    -1.91775    0.37379  -5.131 3.12e-07 ***
`Veh Class`station wagon    0.82363    0.48700   1.691 0.090923 .
`Veh Class`van             -6.74606    1.97023  -3.424 0.000627 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.904 on 2404 degrees of freedom
Multiple R-squared:  0.5748,    Adjusted R-squared:  0.5731
F-statistic:   325 on 10 and 2404 DF,  p-value: < 2.2e-16
```

As expected there are 9 dummy variables for `Veh Class`.

Notice that three of the dummy variables for `Veh Class` are not significant. However, we cannot remove just one of the dummy variables. We need to keep all or none of them. In other words, either `Veh Class` is significant or it is not. In this case, since most of the dummy variables have very small p-values we will keep them.

Consider what happens if we add `Cert Region`, the certification region. I would expect this not to be related to mpg, but let's check.

```r
summary(lm(`Cmb MPG` ~ Displ + `Veh Class` + `Cert Region`))
```

```
Call:
lm(formula = `Cmb MPG` ~ Displ + `Veh Class` + `Cert Region`)

Residuals:
    Min      1Q  Median      3Q     Max
-8.1151 -2.3643 -0.4467  1.3026 29.5464

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)                33.43167    0.36223  92.293  < 2e-16 ***
Displ                      -3.11127    0.07016 -44.346  < 2e-16 ***
`Veh Class`midsize car      2.20845    0.33662   6.561 6.54e-11 ***
`Veh Class`minivan         -1.07034    1.04304  -1.026 0.304913
`Veh Class`pickup          -2.17453    0.39492  -5.506 4.05e-08 ***
`Veh Class`small car       -0.45619    0.30196  -1.511 0.130982
`Veh Class`small SUV       -1.84481    0.33108  -5.572 2.80e-08 ***
`Veh Class`special purpose -3.87282    0.59912  -6.464 1.23e-10 ***
`Veh Class`standard SUV    -1.91752    0.37387  -5.129 3.15e-07 ***
```

```
`Veh Class`station wagon    0.82432    0.48711   1.692 0.090725 .
`Veh Class`van             -6.74565    1.97063  -3.423 0.000629 ***
`Cert Region`FA             0.02910    0.15893   0.183 0.854731
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.905 on 2403 degrees of freedom
Multiple R-squared:  0.5748,    Adjusted R-squared:  0.5729
F-statistic: 295.4 on 11 and 2403 DF,  p-value: < 2.2e-16
```

Here we see the dummy variable for `Cert Region` is not significant, so we should remove it from our model.

Practice

Now use the EPA data to try to create a better model. We've decided that `displ` and `Veh Class` make sense to include while `Cert Region` does not. Consider the other variables in this data set and create a model using all the variables that seem most appropriate. Use backward elimination to refine your model.

Note: You should likely not include `Hwy MPG` or `City MPG` because `Cmb MPG` is just a combination of the two of them. Instead, use characteristics of the vehicle to try and predict `Cmb MPG`.