```
Course: Regression_Models
Lesson: MultiVar_Examples2


- Class: text
  Output: "MultiVar_Examples2. (Slides for this and other Data Science courses may be found at github https://github.com/DataScienceSpecialization/courses. If you care to use them, they must be downloaded as a zip file and viewed locally. This lesson corresponds to Regression_Models/02_02_multivariateExamples.)"

- Class: text
  Output: This is the second lesson in which we'll look at some regression models with more than one independent variable. We'll begin with the InsectSprays data which we've taken the liberty to load for you. This data is part of R's datasets package. It shows the effectiveness of different insect sprays. We've used the code from the slides to show you a boxplot of the data.

- Class: exact_question
  Output: How many Insect Sprays are in this dataset?
  CorrectAnswer: 6
  AnswerTests: omnitest(correctVal=6)
  Hint: How many letters are  in the sequence of letters  A, B, C, D, E, F ?


- Class: text_question
  Output: From the boxplot, which spray has the largest median?
  CorrectAnswer: B
  AnswerTests: omnitest(correctVal='B')
  Hint: The median of each spray is indicated by the thick horizontal line in each box. Which spray has its median furthest from 0?


- Class: cmd_question
  Output: Let's first try to get a better understanding of the dataset InsectSprays. Use the R function dim to find the dimensions of the data.
  CorrectAnswer: dim(InsectSprays)
  AnswerTests: omnitest(correctExpr='dim(InsectSprays)')
  Hint: Type "dim(InsectSprays)" at the R prompt.

- Class: cmd_question
  Output: The R function dim says that InsectSprays is a 72 by 2 array. Use the R function head to look at the first 15 elements of InsectSprays.
  CorrectAnswer: head(InsectSprays,15)
  AnswerTests: omnitest(correctExpr='head(InsectSprays,15)')
  Hint: Type "head(InsectSprays,15)" at the R prompt.

- Class: cmd_question
  Output: So this dataset contains 72 counts, each associated with a particular different spray. The counts are in the first column and a letter identifying the spray in the second. To save you some typing we've created 6 arrays with just the count data for each spray. The arrays have the names sx, where x is A,B,C,D,E or F. Type one of the names (your choice) of these arrays to see what we're talking about.
  CorrectAnswer: sA
  AnswerTests: ANY_of_exprs('sA','sB','sC','sD','sE','sF')
  Hint: Type sB at the R prompt.

- Class: cmd_question
  Output: As a check, run the R command summary on the second column of the dataset to see how many entries we have for each spray.  (Recall that the expression M[ ,2] yields the second column of the array M.)
  CorrectAnswer: summary(InsectSprays[,2])
  AnswerTests: omnitest(correctExpr='summary(InsectSprays[,2])')
  Hint: Type "summary(InsectSprays[,2])" at the R prompt.

- Class: text
  Output: It's not surprising that with 72 counts we'd have 12 count for each of the 6 sprays. In this lesson we'll consider multilevel factor levels and how we interpret linear models of data with more than 2 factors.
```

```
 - Class: cmd_question
   Output: Use the R function sapply to find out the classes of the columns of the data.
   CorrectAnswer: sapply(InsectSprays,class)
   AnswerTests: omnitest(correctExpr='sapply(InsectSprays,class)')
   Hint: Type "sapply(InsectSprays,class)" at the R prompt.

 - Class: text
   Output: The class of the second "spray" column is factor. Recall from the slides that the equation representing the relationship between a particular outcome and several factors contains  binary variables, one for each factor. This data has 6 factors so we need 6 dummy variables. Each will indicate if a particular outcome (a count) is associated with a specific factor or category (insect spray).

 - Class: cmd_question
   Output: Using R's lm function, generate the linear model in which count is the dependent variable and spray is the independent. Recall that in R formula has the form y ~ x, where y depends on the predictor x. The data set is InsectSprays. Store the model in the variable fit.
   CorrectAnswer: fit <- lm(count ~ spray, InsectSprays)
   AnswerTests: creates_lm_model('fit <- lm(count ~ spray, InsectSprays)')
   Hint: Type "fit <- lm(count ~ spray, InsectSprays)" at the R prompt.

 - Class: cmd_question
   Output: Using R's summary function, look at the coefficients of the model. Recall that these can be accessed with the R construct x$coef.
   CorrectAnswer: summary(fit)$coef
   AnswerTests: omnitest(correctExpr='summary(fit)$coef')
   Hint: Type "summary(fit)$coef" at the R prompt.

 - Class: cmd_question
   Output: Notice that R returns a 6 by 4 array. For convenience, store off the first column of this array, the Estimate column, in a variable called est. Remember the R construct for accessing the first column is x[,1].
   CorrectAnswer: est <- summary(fit)$coef[,1]
   AnswerTests: omnitest(correctExpr='est <- summary(fit)$coef[,1]')
   Hint: Type "est <- summary(fit)$coef[,1]" at the R prompt.


 - Class: text
   Output: Notice that sprayA does not appear explicitly in the list of Estimates. It is there, however, as the first entry in the Estimate column. It is labeled as "(Intercept)". That is because sprayA is the first in the alphabetical list of the levels of the factor, and R by default uses the first level as the reference against which the other levels or groups are compared when doing its t-tests (shown in the third column).

 - Class: cmd_question
   Output: What do the Estimates of this model represent? Of course they are the coefficients of the binary or dummy variables associated with sprays. More importantly, the Intercept is the mean of the reference group, in this case sprayA, and the other Estimates are the distances of the other groups' means from the reference mean. Let's verify these claims now. First compute the mean of the sprayA counts. Remember the counts are all stored in the vectors named sx. Now we're interested in finding the mean of sA.
   CorrectAnswer: mean(sA)
   AnswerTests: omnitest(correctExpr='mean(sA)')
   Hint: Type "mean(sA)" at the R prompt.

 - Class: mult_question
   Output: What do you think the mean of sprayB is?
   AnswerChoices: 15.3333; 0.83333; -12.41667; I haven't a clue
   CorrectAnswer: 15.3333
   AnswerTests: omnitest(correctVal='15.3333')
   Hint: Adding the value of the Intercept to the Estimate for sprayB yields the empirical mean of sprayB.

 - Class: cmd_question
   Output: Verify this now by using R's mean function to compute the mean of sprayB.
   CorrectAnswer: mean(sB)
```

```
  AnswerTests: omnitest(correctExpr='mean(sB)')
  Hint: Type "mean(sB)" at the R prompt.

- Class: cmd_question
  Output: Let's generate another model of this data, this time omitting the intercept. We can easily use R's lm function to do this by appending  " - 1" to the formula, e.g., count ~ spray - 1. This tells R to omit the first level. Do this now and store the new model in the variable nfit.
  CorrectAnswer: nfit <- lm(count ~ spray - 1, InsectSprays)
  AnswerTests: creates_lm_model('nfit <- lm(count ~ spray - 1, InsectSprays)')
  Hint: Type "nfit <- lm(count ~ spray - 1, InsectSprays)" at the R prompt.

- Class: cmd_question
  Output: Now, as before, look at the coefficient portion of the summary of nfit.
  CorrectAnswer: summary(nfit)$coef
  AnswerTests: omnitest(correctExpr='summary(nfit)$coef')
  Hint: Type "summary(nfit)$coef" at the R prompt.

- Class: text
  Output: Notice that sprayA now appears explicitly in the list of Estimates. Also notice how the values of the columns have changed. The means of all the groups are now explicitly shown in the Estimate column. Remember that previously, with an intercept, sprayA was excluded, its mean was the intercept, and the values for the other sprays (estimates, standard errors, and t-tests) were all computed relative to sprayA, the reference group. Omitting the intercept clearly affected the model.

- Class: mult_question
  Output: What values does the Estimate column now show?
  AnswerChoices: The means of all 6 levels; The variances of all 6 levels; I have no idea
  CorrectAnswer: The means of all 6 levels
  AnswerTests: omnitest(correctVal='The means of all 6 levels')
  Hint: The numbers should look familiar, especially for sprayA and sprayB. What values have you computed for these two sprays?

- Class: mult_question
  Output: Without an intercept (reference group) the tests are whether the expected counts (the groups means) are different from zero. Which spray has the least significant result?
  AnswerChoices: sprayC; sprayF; sprayB; sprayA
  CorrectAnswer: sprayC
  AnswerTests: omnitest(correctVal='sprayC')
  Hint: Which spray has the highest probability?

- Class: text
  Output: Clearly, which level is first is important to the model. If you wanted a different reference group, for instance, to compare sprayB to sprayC, you could refit the model with a different reference group.

- Class: cmd_question
  Output: The R function relevel does precisely this. It re-orders the levels of a factor. We'll do this now. We'll call relevel with two arguments. The first is the factor, in this case InsectSprays$spray,  and the second is the level that we want to be first,  in this case "C". Store the result in a new variable spray2.
  CorrectAnswer: spray2 <- relevel(InsectSprays$spray,"C")
  AnswerTests: omnitest(correctExpr='spray2 <- relevel(InsectSprays$spray,"C")')
  Hint: Type "spray2 <- relevel(InsectSprays$spray,\"C\")" at the R prompt.

- Class: cmd_question
  Output: Now generate a new linear model and put the result in the variable fit2.
  CorrectAnswer: fit2 <- lm(count ~ spray2, InsectSprays)
  AnswerTests: creates_lm_model('fit2 <- lm(count ~ spray2, InsectSprays)')
  Hint: Type "fit2 <- lm(count ~ spray2, InsectSprays)" at the R prompt.


- Class: cmd_question
  Output: As before, look at the coef portion of the summary of this new model fit2. See how sprayC is now the intercept (since it doesn't appear explicitly in the list).
  CorrectAnswer: summary(fit2)$coef
  AnswerTests: omnitest(correctExpr='summary(fit2)$coef')
```

```
  Hint: Type "summary(fit2)$coef" at the R prompt.

- Class: mult_question
  Output: According to this new model what is the mean of spray2C?
  AnswerChoices: 2.083333; 12.416667; 14.583333; The model doesn't tell me.
  CorrectAnswer: 2.083333
  AnswerTests: omnitest(correctVal='2.083333')
  Hint: Recall that the intercept is the mean of the reference group, in this case sprayC, so look at the value in the (Intercept) row of the Estimate column.

- Class: cmd_question
  Output: Verify your answer with R's mean function using the array sC as the argument.
  CorrectAnswer: mean(sC)
  AnswerTests: omnitest(correctExpr='mean(sC)')
  Hint: Type "mean(sC)" at the R prompt.


- Class: mult_question
  Output: According to this new model what is the mean of spray2A?
  AnswerChoices: 14.50000; 12.416667; 14.583333; I don't have a clue
  CorrectAnswer: 14.50000
  AnswerTests: omnitest(correctVal='14.50000')
  Hint: Recall that when there is an intercept, the mean of a level that's not the reference, is the intercept + the coefficient (or estimate) of that level, in this case spray2a, so you'll have to add together two numbers. Alternatively, just look back and see what the mean was for the original model.

- Class: text
  Output: Remember that with this model sprayC is the reference group, so the t-test statistics (shown in column 3 of the summary coefficients) compare the other sprays to sprayC. These can be computed by hand using the Estimates and standard error from the original model (fit) which used sprayA as the references.

- Class: cmd_question
  Output: The slides show the details of this but here we'll demonstrate by calculating the spray2B t value.  Subtract fit's sprayC coefficient (fit$coef[3]) from sprayB's (fit$coef[2]) and divide by the standard error which we saw was 1.6011. The result is spray2B's t value. Do this now.
  CorrectAnswer: (fit$coef[2]-fit$coef[3])/1.6011
  AnswerTests: omnitest(correctExpr='(fit$coef[2]-fit$coef[3])/1.6011')
  Hint: Type "(fit$coef[2]-fit$coef[3])/1.6011" at the R prompt.

- Class: text
  Output: We glossed over some details in this lesson. For instance, counts can never be 0 so the assumption of normality is violated. We'll explore this issue more when we discuss Poisson GLMs. For now be glad that you've concluded this second lesson on multivariable linear models.

- Class: mult_question
  Output: "Would you like to receive credit for completing this course on
    Coursera.org?"
  CorrectAnswer: NULL
  AnswerChoices: Yes;No
  AnswerTests: coursera_on_demand()
  Hint: ""
```