Exploration librairie swirl : Multivar Examples 2

> swirl()

Welcome to swirl! Please sign in. If you've been here before, use the same name as you did then. If you are new, call yourself something unique.

What shall I call you? ***jlbellier***

Would you like to continue with one of these lessons?

1: Regression Models Least Squares Estimation
2: No. Let me start something new.

Selection: ***2***

Please choose a course, or type 0 to exit swirl.

1: Regression Models
2: Statistical Inference
3: Take me to the swirl course repository!

Selection: ***1***

Please choose a lesson, or type 0 to return to course menu.

1: Introduction
2: Residuals
3: Least Squares Estimation
4: Residual Variation
5: Introduction to Multivariable Regression
6: MultiVar Examples
7: MultiVar Examples2
8: MultiVar Examples3
9: Residuals Diagnostics and Variation
10: Variance Inflation Factors
11: Overfitting and Underfitting
12: Binary Outcomes
13: Count Outcomes

Selection: ***7***

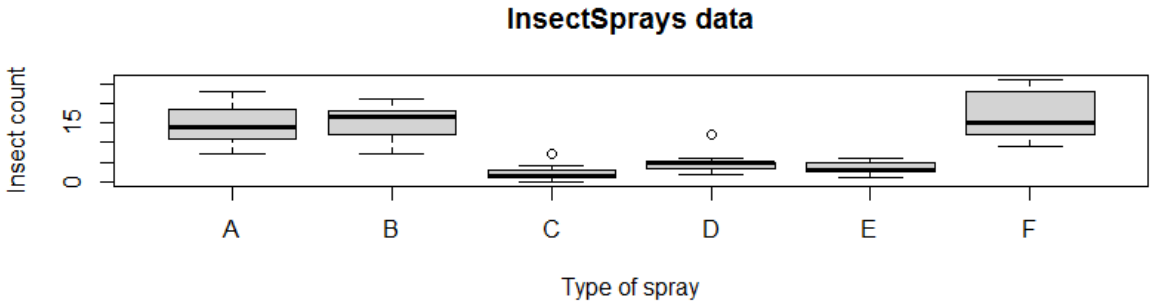
0%

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.)

=== ***3%***

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.



====== ***6%***

How many Insect Sprays are in this dataset?

*> **6***
[1] 6

You are really on a roll!

========= ***9%***

From the boxplot, which spray has the largest median?

*ANSWER: **B***

Excellent job!

============ ***12%***

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.

*> **dim(InsectSprays)***
[1] 72 2

Your dedication is inspiring!

=============== ***15%***

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.

*> **head(InsectSprays,15)***

	count	spray
1	10	A
2	7	A
3	20	A
4	14	A
5	14	A
6	12	A
7	10	A

8 23 A
9 17 A
10 20 A
11 14 A
12 13 A
13 11 B
14 17 B
15 21 B

You are doing so well!

========== **18%**

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.

> **sA**
[1] 10 7 20 14 14 12 10 23 17 20 14 13

You are really on a roll!

========== **21%**

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.)

> **summary(InsectSprays[,2])**
A B C D E F
12 12 12 12 12 12

Nice work!

========== **24%**

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.

========== **27%**

Use the R function sapply to find out the classes of the columns of the data.

> **sapply(InsectSprays,class)**
count spray
"numeric" "factor"

Keep working like that and you'll get there!

========== **30%**

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).

========== **33%**

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.

> ***fit <- lm(count~spray,InsectSprays)***

Excellent job!

========== **36%**

Using R's summary function, look at the coefficients of the model. Recall that these can be accessed with the R construct x$coef.

> ***summary(fit)$coef***

```
            Estimate Std. Error    t value     Pr(>t)
(Intercept)  14.5000000   1.132156 12.8074279 1.470512e-19
sprayB        0.8333333   1.601110  0.5204724 6.044761e-01
sprayC      -12.4166667   1.601110 -7.7550382 7.266893e-11
sprayD       -9.5833333   1.601110 -5.9854322 9.816910e-08
sprayE      -11.0000000   1.601110 -6.8702352 2.753922e-09
sprayF        2.1666667   1.601110  1.3532281 1.805998e-01
```

That's the answer I was looking for.

========== **39%**

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].

> ***est <- summary(fit)$coef[,1]***

All that hard work is paying off!

========== **42%**

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).

========== **45%**

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.

> ***mean(sA)***

```
[1] 14.5
```

Excellent work!

========== 48%

What do you think the mean of sprayB is?

1: 0.83333
2: I haven't a clue
3: -12.41667
4: 15.3333
Selection: 4

Keep working like that and you'll get there!

========== 52%

Verify this now by using R's mean function to compute the mean of sprayB.

> mean(sB)
[1] 15.33333

You got it right!

========== 55%

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.

> nfit <- lm(count~spray-1,InsectSprays)

That's a job well done!

========== 58%

Now, as before, look at the coefficient portion of the summary of nfit.

```
> summary(nfit)$coef
       Estimate Std. Error   t value     Pr(>t)
sprayA 14.500000   1.132156 12.807428 1.470512e-19
sprayB 15.333333   1.132156 13.543487 1.001994e-20
sprayC  2.083333   1.132156  1.840148 7.024334e-02
sprayD  4.916667   1.132156  4.342749 4.953047e-05
sprayE  3.500000   1.132156  3.091448 2.916794e-03
sprayF 16.666667   1.132156 14.721181 1.573471e-22
```

You are amazing!

========== 61%

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.

========== ***64%***

What values does the Estimate column now show?

1: The means of all 6 levels
2: The variances of all 6 levels
3: I have no idea

Selection: ***1***

Nice work!

========== ***67%***

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?

1: sprayF
2: sprayA
3: sprayC
4: sprayB

Selection: ***4***

You're close...I can feel it! Try it again.

Which spray has the highest probability?

1: sprayF
2: sprayB
3: sprayC
4: sprayA

Selection: ***3***

You are amazing!

========== ***70%***

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.

========== ***73%***

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.

Type "spray2 <- relevel(InsectSprays$spray,\"C\")" at the R prompt.

> ***spray2 <- relevel(InsectSprays$spray,"C")***

You are really on a roll!

========== 76%

Now generate a new linear model and put the result in the variable fit2.

> **fit2 <- lm(count~spray2,InsectSprays)**

All that practice is paying off!

========== 79%

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).

> **summary(fit2)$coef**

```
             Estimate Std. Error  t value     Pr(>t)
(Intercept)  2.083333   1.132156 1.840148 7.024334e-02
spray2A     12.416667   1.601110 7.755038 7.266893e-11
spray2B     13.250000   1.601110 8.275511 8.509776e-12
spray2D      2.833333   1.601110 1.769606 8.141205e-02
spray2E      1.416667   1.601110 0.884803 3.794750e-01
spray2F     14.583333   1.601110 9.108266 2.794343e-13
```

Excellent work!

========== 82%

According to this new model what is the mean of spray2C?

1: 14.583333
2: 12.416667
3: The model doesn't tell me.
4: 2.083333

Selection: **4**

Excellent work!

========== 85%

Verify your answer with R's mean function using the array sC as the argument.

> **mean(sC)**
[1] 2.083333

You got it right!

========== 88%

According to this new model what is the mean of spray2A?

1: 14.583333
2: 14.50000
3: I don't have a clue
4: 12.416667

*Selection: **2***

That's the answer I was looking for.

========== 91%

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.

========== 94%

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.

*> **(fit$coef[2]-fit$coef[3])/1.6011***

sprayB

8.275561

That's correct!

========== 97%

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.

========== 100%