Exploration librairie swirl : Multivariable regression

> 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: ***5***

0%

Introduction to Multivariable Regression. (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_01_multivariate. Galton data is from John Verzani's Using R website, http://wiener.math.csi.cuny.edu/UsingR/)

===== ***4%***

In this lesson we'll illustrate that regression in many variables amounts to a series of regressions in one. Using regression in one variable, we'll show how to eliminate any chosen regressor, thus reducing a regression in N variables, to a regression in N-1. Hence, if we know how to do a regression in 1 variable, we can do a regression in 2. Once we know how to do a regression in 2 variables, we can do a regression in 3, and so on. We begin with the galton data and a review of eliminating the intercept by subtracting the means.

========== ***8%***

When we perform a regression in one variable, such as lm(child ~ parent, galton), we get two coefficients, a slope and an intercept. The intercept is really the coefficient of a special regressor which has the same value, 1, at every sample. The function, lm, includes this regressor by default.

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

We'll demonstrate by substituting an all-ones regressor of our own. This regressor must have the same number of samples as galton (928.) Create such an object and name it ones, using ones <- rep(1, nrow(galton)), or some equivalent expression.

*> **ones <- rep(1,nrow(galton))***

You got it!

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

The galton data has already been loaded. The default intercept can be excluded by using -1 in the formula. Perform a regression which substitutes our regressor, ones, for the default using lm(child ~ ones + parent -1, galton). Since we want the result to print, don't assign it to a variable.

*> **lm(child ~ ones + parent -1,galton)***

```
Call:
lm(formula = child ~ ones + parent - 1, data = galton)

Coefficients:
  ones   parent
23.9415   0.6463
```

That's a job well done!

========== ***19%***

The coefficient of ones is 23.9415. Now use the default, lm(child ~ parent, galton), to show the intercept has the same value. This time, DO NOT suppress the intercept with -1.

*> **lm(child ~parent,galton)***

```
Call:
lm(formula = child ~ parent, data = galton)

Coefficients:
(Intercept)      parent
   23.9415      0.6463
```

Keep working like that and you'll get there!

========== ***23%***

The regression in one variable given by lm(child ~ parent, galton) really involves two regressors, the variable, parent, and a regressor of all ones.

1: False
2: True

*Selection: **1***

Keep trying!

Since it produces two coefficients, it must involve two regressors. One is a variable named parent, the other is the constant, 1.

1: False
2: True

*Selection: **2***

You're the best!

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

In earlier lessons we demonstrated that the regression line given by lm(child ~ parent, galton) goes through the point x=mean(parent), y=mean(child). We also showed that if we subtract the mean from each variable, the regression line goes through the origin, x=0, y=0, hence its intercept is zero. Thus, by subtracting the means, we eliminate one of the two regressors, the constant, leaving just one, parent. The coefficient of the remaining regressor is the slope.

========== ***31%***

Subtracting the means to eliminate the intercept is a special case of a general technique which is sometimes called Gaussian Elimination. As it applies here, the general technique is to pick one regressor and to replace all other variables by the residuals of their regressions against that one.

========== ***35%***

Suppose, as claimed, that subtracting a variable's mean is a special case of replacing the variable with a residual. In this special case, it would be the residual of a regression against what?

1: The constant, 1
2: The variable itself
3: The outcome

*Selection: **2***

Almost! Try again.

A residual is the difference between a variable and its predicted value. If, for example, child-mean(child) is a residual, then mean(child) must be its predicted value. But mean(child) is a constant, so the regressor would be a constant.

1: The constant, 1
2: The variable itself
3: The outcome

Selection: ***1***

That's correct!

========== ***38%***

The mean of a variable is the coefficient of its regression against the constant, 1. Thus, subtracting the mean is equivalent to replacing a variable by the residual of its regression against 1. In an R formula, the constant regressor can be represented by a 1 on the right hand side. Thus, the expression, lm(child ~ 1, galton), regresses child against the constant, 1. Recall that in the galton data, the mean height of a child was 68.09 inches. Use lm(child ~ 1, galton) to compare the resulting coefficient (the intercept) and the mean height of 68.09. Since we want the result to print, don't assign it a name.

> ***lm(child ~ 1, galton)***

Call:
lm(formula = child ~ 1, data = galton)

Coefficients:
(Intercept)
68.09

That's the answer I was looking for.

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

The mean of a variable is equal to its regression against the constant, 1.

1: False
2: True

Selection: ***2***

You are really on a roll!

========== ***46%***

To illustrate the general case we'll use the trees data from the datasets package. The idea is to predict the Volume of timber which a tree might produce from measurements of its Height and Girth. To avoid treating the intercept as a special case, we have added a column of 1's to the data which we shall use in its place. Please take a moment to inspect the data using either View(trees) or head(trees).

> View(trees)

All that practice is paying off!

========== 50%

A file of relevant code has been copied to your working directory and sourced. The file, elimination.R, should have appeared in your editor. If not, please open it manually.

========== 54%

The general technique is to pick one predictor and to replace all other variables by the residuals of their regressions against that one. The function, regressOneOnOne, in eliminate.R performs the first step of this process. Given the name of a predictor and one other variable, other, it returns the residual of other when regressed against predictor. In its first line, labeled Point A, it creates a formula. Suppose that predictor were 'Girth' and other were 'Volume'. What formula would it create?

1: Girth ~ Volume - 1
2: Volume ~ Girth - 1
3: Volume ~ Girth

Selection: ***2***

You are really on a roll!

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

The remaining function, eliminate, applies regressOneOnOne to all variables except a given predictor and collects the residuals in a data frame. We'll first show that when we eliminate one regressor from the data, a regression on the remaining will produce their correct coefficients. (Of course, the coefficient of the eliminated regressor will be missing, but more about that later.)

==========62%

For reference, create a model named fit, based on all three regressors, Girth, Height, and Constant, and assign the result to a variable named fit. Use an expression such as fit <- lm(Volume ~ Girth + Height + Constant -1, trees). Don't forget the -1, and be sure to name the model fit for later use.

> ***fit <- lm(Volume~Girth + Height + Constant -1,trees)***

Keep up the great work!

==========65%

Now let's eliminate Girth from the data set. Call the reduced data set trees2 to indicate it has only 2 regressors. Use the expression trees2 <- eliminate("Girth", trees).

> ***trees2 <- eliminate("Girth",trees)***

Keep working like that and you'll get there!

==========69%

Use head(trees2) or View(trees2) to inspect the reduced data set.

> ***View(trees2)***

That's a job well done!

==========73%

Why, in trees2, is the Constant column not constant?

1: Computational precision was insufficient.
2: There must be some mistake
3: The constant, 1, has been replaced by its residual when regressed against Girth.

Selection: ***3***

You got it right!

==========77%

Now create a model, called fit2, using the reduced data set. Use an expression such as fit2 <- lm(Volume ~ Height + Constant -1, trees2). Don't forget to use -1 in the formula.

> ***fit2 <- lm(Volume ~Constant + Height-1,trees2)***

You are doing so well!

==========81%

Use the expression lapply(list(fit, fit2), coef) to print coefficients of fit and fit2 for comparison.

> ***lapply(list(fit, fit2), coef)***

```
[[1]]
    Girth    Height  Constant
4.7081605 0.3392512 -57.9876589

[[2]]
  Constant    Height
-57.9876589 0.3392512
```

That's a job well done!

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

The coefficient of the eliminated variable is missing, of course. One way to get it would be to go back to the original data, trees, eliminate a different regressor, such as Height, and do another 2 variable regession, as above. There are much more efficient ways, but efficiency is not the point of this demonstration. We have shown how to reduce a regression in 3 variables to a regression in 2. We can go further and eliminate another variable, reducing a regression in 2 variables to a regression in 1.

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

Here is the final step. We have used eliminate("Height", trees2) to reduce the data to the outcome, Volume, and the Constant regressor. We have regressed Volume on Constant, and printed the coefficient as shown in the command above the answer. As you can see, the coefficient of Constant agrees with previous values.

```
Call:
lm(formula = Volume ~ Constant - 1, data = eliminate("Height",
    trees2))
```

Coefficients:
Constant
-57.99
==========92%
Suppose we were given a multivariable regression problem involving an outcome and N regressors, where N > 1. Using only single-variable regression, how can the problem be reduced to a problem with only N-1 regressors?

1: Subtract the mean from the outcome and each regressor.
2: Pick any regressor and replace the outcome and all other regressors by their residuals against the chosen one.

Selection: ***2***

You got it!
==========96%
We have illustrated that regression in many variables amounts to a series of regressions in one. The actual algorithms used by functions such as lm are more efficient, but are computationally equivalent to what we have done. That is, the algorithms use equivalent steps but combine them more efficiently and abstractly. This completes the lesson.
==========100%