

A short lesson in how to write a script (and how to do logistic regression in SPSS).

[Http://kochanski.org/gpk/teaching/0910statistics](http://kochanski.org/gpk/teaching/0910statistics) and [../data8](#) .

The goal is to take some car-counting data and to fit it first with a linear regression where the total observations is the dependent variable and the number of red cars is the independent variable. (That we do in SPSS.) And, then fit it with a logistic regression.

Now, in SPSS, logistic regressions require you to have all the cases, not just the totals of how many of each type there are. So, what we will do is write a little script to synthesize a set of cases that are consistent with the summary data. (Just don't forget that these synthesized cases aren't your real data or you'll end up in trouble.)

So, here's the summary data:

```
$ cat cars_summarized.txt
red,blue,total
10,22,30
10,40,50
10,90,100
14,46,60
11,59,70
45,255,300
$
```

And here's a script to synthesize the cases:

```
$ cat make_cars_raw.py
#! python
"""This program takes summary data on car counts and synthesizes
cases so that we can run a logistic regression in SPSS.
"""

import sys

INPUT = "cars_summarized.txt"
OUTPUT = "cars_temp.txt"
open("LOG.txt", "w").writelines("Running %s %s -> %s\n"
                                % (sys.argv[0], INPUT, OUTPUT)
                                )

input = open(INPUT, "r")
out = open(OUTPUT, "w")

for (i,l) in enumerate(input):
    if i == 0:
        out.writelines("is_red,is_blue,counter\n" )
        out.writelines("# These are synthesized cases, consistent with\n" )
        out.writelines("# the totals in %s. TEMPORARY FILE.\n" % INPUT )
        out.writelines("# Intended for SPSS logistic regression.\n" )
        continue
    r,b,t = l.strip().split(',')
    assert int(r)+int(b) == int(t), "whoops: i=%d" % i
    for i in range(int(r)):
        out.writelines( "1,0,%d" % i )
    else:
        out.writelines( "0,1,%d" % i )

$
```

But when you run it, you get an error:

```
$ python make_cars_raw.py
is_red, is_blue, counter
Traceback (most recent call last):
  File "make_cars_raw.py", line 9, in <module>
    assert int(r)+int(b) == int(t), "whoops: i=%d" % i
AssertionError: whoops: i=1
$
```

Why? Because I mistyped the first line. $10+22$ is not 30. Good thing that the script was doing some basic sanity checks, because I probably wouldn't have noticed.

This script leaves a log file (called LOG.txt) that says what happened:

```
$ cat LOG.txt
Running make_cars_raw.py cars_summarized.txt -> cars_temp.txt
$
```

So, we look back at the raw data (which we kept, of course!) and fix the first line, run the script again, then we can use SPSS. (Well, actually, there's a bug in that script. I forgot to add a newline while writing the synthesized observations, so it all ended up on one line. Fix that, run once again, then look at the synthetic observations and realize that I used the variable "i" twice, for two different purposes. Fix, **that**, by changing the inner loop to "j", and run again.

Now, move it into SPSS.

We will run it as a Logistic regression, and look to see if the "counter" – i.e. who was counting the cars – makes a significant difference. It says "No", as it should, since we generated all the data from the same probability distribution: each counter had a 15% chance of seeing a red car.

In Step0, when there is just the constant in the logistic regression (i.e. we ignore the effect of the "counter" independent variable), we get "Constant" = 2.793 with a standard error of 0.421, and highly significant. But what does that mean?

A logistic regression is written in terms of the log of the "Likelihood ratio", the ratio of the probability of seeing a not-red car to the probability of seeing a red car. So, SPSS is saying that the best estimate for $\log(P(\text{red})/P(\text{not_red})) = 2.793$.

Which means $P(\text{red})/P(\text{not_red}) = 16.33$, which SPSS helpfully gives you in the right-most column.

Now, $P(\text{not-red}) = 1 - P(\text{red})$, so we can do a little algebra to get $P(\text{red}) = 16.33/(1 + 16.33) = 0.9422$.

Hmm..... That's not what I expected..... it says there are 94% red cars, but that's not *my*

impression of Oxford's streets. The other odd thing is that there are only 104 cases reported by SPSS, and the bottom line of the summary file has 300 cases. That's a bug.

Let's look at the code again. The last few lines really ought to look like this:

```
for j in range(int(r)):  
    out.writelines( "1,0,%d\n" % i )  
for j in range(int(b)):  
    out.writelines( "0,1,%d\n" % i )
```

Re-run and repeat SPSS. Now we get "constant"=-1.653, $\exp(\text{constant})=.191$, and we deduce $P(\text{red})=0.160$. We also get the correct number of cases, so this is probably the right answer.

Now, the standard error of the constant is 0.110, so we can (roughly speaking) compute the standard error of $P(\text{red})$ by incrementing "constant" by 0.110 and -0.110 and seeing how much it changes $P(\text{red})$. We find that changing "constant" by one standard deviation either way moves $P(\text{red})$ from 0.146 to 0.176, so the standard error in $P(\text{red})$ is approximately 0.0148.

So, $P(\text{red}) = 0.160 \pm 0.015$, which covers the true value we used to compute the data (i.e. 0.15).

The logistic regression, now done right, indicates that none of the counters are significantly different from the average.

MISC NOTES and LINKS:

Nonparametric Methods

<http://sst-web.tees.ac.uk/external/U0000504/Notes/DataAnalysis/DistFree/NonPara.html>

Logistic Regression

<http://faculty.chass.ncsu.edu/garson/PA765/logistic.htm>

Logistic Regression using R:

<http://www.stat.washington.edu/quinn/classes/536/S/logitexample2.html>

Logit and Probit regression:

Generally similar. Both are a good way to estimate success/failure or counting experiments. Logit and Probit have slightly different mathematical forms with the same qualitative behaviour. Probit is connected to Gaussian distributions: it says that the z-statistic (or the z-transform of the data) is linearly related to all the factors.

Logit says the same for an exponential distribution. It also says that the odds ratio of the two outcomes is a product of something relating to each factor.

Odds ratio: if you're counting blue and red cars, the odds ratio is the chance of getting red divided by the chance of getting blue.

More notes on logistic regression, with detailed descriptions of the SPSS dialogue boxes:

<http://www.jeremymiles.co.uk/regressionbook/extras/appendix2/spss9.0/index.html>