Stat-340 – Assignment 5 – 2014 Spring Term

Part 1 - Breakfast cereals - Intermediate

In this part of the assignment, you will learn how to:

- simulate data that satisfy the assumptions of a model.
- simulate data that does not satisfy the assumptions of a model.
- use a simple simulation to study the effect of a violation of an assumption on the performance of an estimator.

Simulation studies are often used to investigate the performance of an estimation method when some of the usual assumptions about the problem are violated. For example, any regression analysis makes several assumptions – Section 15.4.4 of my CourseNotes and `http://en.wikipedia.org/wiki/Regression_analysis#Underlying_assumptions`.

A common assumption in regression models is that the random noise around the regression line (the noise or residuals) follow a Normal distribution with mean 0 and constant variance σ^2. But what happens if this assumption is violated? For example, the *Log-Normal* distribution assumes that the logarithm of the residuals follows a normal distribution, but then the residuals have a long right tail (see `http://en.wikipedia.org/wiki/Log-normal_distribution`). This long right tail will tend to give the occasional outlier to the model.

Any simulation study generally has several stages:

- Decide upon the model (or design) space, In multiple regression, you need to decide which values of the X variables will be used, and the sample size to be simulated.

- Choose realistic values for the population parameters. In regression situations, we need to choose the intercept and slopes that are realistic, along with an value for the standard deviation of points around the regression line.
- Decide upon the aspect of the model to be examined. In this example, we will change the distribution of the the random error (the residuals) from a normal distribution to a log-normal distribution.
- Generate simulated data under BOTH the correct model and the incorrect model. Often about 1000 simulated sets of data will be generated. Just like in the previous assignment, you will "debug" your program using a smaller number of simulations.
- Analyze each of the simulated datasets. This is similar to what you did in Assignment 4 for bootstrapping.
- Extract the estimates from each simulated dataset.
- Summarize the **estimates** from the simulated datasets. For example, recall that an estimator is unbiased if the mean of the estimates over (infinite) repeated samples from the population matches the population parameter. So we will examine the mean of the estimates from our 1000 simulated samples to see if the mean of the estimates are close to the POPULATION values used to generate the data.

  Similarly, the standard deviation of the estimates over (infinite) repeated samples from the population is, by definition, the standard error of the estimate (by now, you should definitely understand this point!). So we will compute the standard deviation of our estimates from the 1000 simulated samples to find the standard error under each model.

  Finally, we can also examine the performance of the hypothesis tests. If all assumptions are satisfied, then the hypothesis test against the POPULATION parameter value used to generate the samples should NOT be "rejected" (why?). However, again by definition, when ever you compare the p-value to the "magic" $\alpha = 0.5$, you are willing to accept a 5% chance of a false positive (Type I error) – refer to `http://en.wikipedia.org/wiki/Type_I_and_type_II_errors`. So if you do 1000 hypothesis tests against the population parameter values, you should find about $0.05 \times 1000 = 50$ false positive results.

We will use a multiple regression of calories/serving against the amount of fat, protein, complex carbohydrates and sugars in the cereal in our simulation study. We will base our simulation on the cereal dataset used in previous assignments.

1. Read the cereal data into *SAS*.

2. Deal with the coding for missing values as done in previous assignments..

3. **Create the model (design) space.** There are several ways to create the *model* (also known as the *design*) space. You could generate individual X values (see Part II of this assignment) or use existing data to represent the distribution of X variable likely to be encountered in practice. For example, it would be difficult to "make up" sets of X variables (the fat, protein, complex carbohydrates, and sugars) that matched real life cereals without a great deal of investigation in how manufacturers make these decisions.

   For this reason will use the actual values of protein, fat, complex carbohydrates, and sugar found in actual cereals as the value of X for our simulated data. So we need keep only these variables and discard the rest.

   Your code will look something like:

```
data ModelSpace;
   set cereal;
   keep protein ***** ***** *****;
run;
```

   Print out the first 10 records of the *ModelSpace* dataset. What was the impact of the the *keep* statement. How could you accomplish the same result by using a *drop* statement?

   Notice that the *model* space does NOT have a response value (number of calories/serving). We don't want to analyze an existing dataset, but want to create simulated datasets using ONLY the value of X from this cereal dataset.

4. **Decide on the known parameter values.** Reread the section on nutrition labels at `http://en.wikipedia.org/wiki/Food_energy`. We will use the standard values for calories per gram of fat, protein, and carbohydrates as the KNOWN parameters to simulate our responses. Note that sugar is carbohydrate and so has the same calories per gram. Read `http://www.briancalkins.com/simplevscomplexcarb.htm` for an explanation of the difference between a simple carbohydrate (such as sugar) and a complex carbohydrate (such as from grain).

   Recall that a regression model has the form:

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{3i} + \ldots + \epsilon_i$$

   For this problem, we will have 4 predictor variables plus the intercept, and will assume that the intercept is 0 because virtually ALL of the calories in cereals comes from these four X variables. So there are a total of 5 (why)

β parameters that must be set.

We will set the parameter values using macro variables using a section of code similar to:

```
%let beta_0         = 0;
%let beta_fat       = 9;
%let beta_protein   = 4;
....
```

You will see why we want to macro variables rather than 'hard coding' the slopes later on in the assignment.

5. **Determine the value of the noise parameter.** What is the distribution of the noise (the ϵ) in the model? It turns out that a suitable value for the standard deviation of the noise is around 5[1] calories/serving. Create macro variable for the standard deviation in a similar way as above.

6. **Generate some simulated data.** We will start by simulating some data WHEN ALL ASSUMPTIONS ARE TRUE. In particular, we will generate some data with a Normal distribution for the random part (the residuals, the ϵ_i's) of the model.

   We now want to use the *model* space and the values of the KNOWN parameters to generate some simulated data that, we hope looks like what we could have obtained through a proper sampling scheme.

   Your code will look something like:

```
%let nsim = 5;
data simdata_no_violations;
   call streaminit(314159);
   do sim=1 to &nsim;
        do mycereal=1 to ncereals;
            set modelspace point=mycereal  nobs=ncereals; /* get the X values */
            mu_calories = &beta_0 +
                          &beta_fat                    *fat +
                          &beta_protein             *protein +
                          &beta_complex_carbs*complex_carbs  +
                          &beta_sugars             *sugars;
            epsilon = rand('normal', 0, &sigma);
            calories = mu_calories + epsilon;
            output;
       end;
  end;
```

[1] I determined this using a multiple regression analysis on the cereal dataset using calories/serving vs. the 4 predictor variables that we are using.

```
  stop;  /* DONT FORGET THIS STATEMENT! */
run;
```

Notice that we once again use a macro variable to control the number of simulated datasets that are generated. It should be set to a small value to debug our program but then changed to the value of 1000 for the final run. So the above code will generate 5 simulated datasets (each having 77 records). The *sim* variable will identify the simulated data from each simulation.

You did add a parameter earlier for the intercept didn't you?

Notice that I generate the simulated amount of calories in two steps. I first generated the mean number of calories expected under various combinations of the fat, protein, complex carbs, and sugars. Where did these values come from?

Then I added some random noise. What does the *STREAMINT* function do? Why do you want to set the seed for a random number generator? What does the *RAND* function do, and what are the arguments to this function.

What does the *STOP* statement do? Notice that if you remove this statement, be prepared to halt your program as it continues indefinitely until the end of the universe!

Note the use of the *set* statement in *SAS* and the *point=* option which "selects" rows from the model space dataset. The *nobs=* clause informs *SAS* that the variable *ncereals* holds the number of cereals in the model space dataset and (rather opaquely) you can use this value even before you read a data value from the dataset. (*SAS* "looks ahead" to see where *ncereals* is defined and "knows" the value in advance of the *set* statement.

7. Print out the first 10 records of the *simdata_no_violations* dataset. Do you understand all of the variables?

8. Now we are back to familiar territory. Use *Proc Reg* to analyze each set of simulated data using a *By* statement to repeat the analysis for each simulated data set (which variable defines the data from each simulation?). Use the *OUTEST=MyEst_no_violations* option on the *Proc Reg* statement to capture the estimates for each slope. Also don't forget the NOPRINT option so that you don't get all of the intermediate output.

   IMPORTANT If your program is generating a LARGE pdf file (hundreds of pages), you likely forgot to turn OFF output from procedures when you analyze the simulated datasets. Don't forget the NOPRINT option on the *Proc Reg* statement[2].

[2] Alternatively, you can bracket code where you want output to be halted using `ods select off;` ... code for which output is to be ignored... `ods select on;` to bracket the code.

9. Print out the first few records of *MyEst_ no_ violations* dataset created by *Proc Reg*. Do you understand what each variable contains?

   Take a deep breath, and go back and review all of the steps. DO YOU UNDERSTAND WHAT WAS DONE AT EACH STEP AND WHY?

10. Lets now see if the least-square estimators for the regression coefficients are unbiased. What is the definition of a unbiased estimator? Hint: `http://en.wikipedia.org/wiki/Bias_of_an_estimator`.

    Of course, we CANNOT find the actual expectation of an estimator from simulated data (why?), but we can see if the average value of the estimator over the simulated sets of data matches the true (and in this case) known population value. Why? THIS IS A CRUCIAL STEP IN UNDERSTANDING sampling distributions.

11. Find the average of the estimates over the simulations and compare them to the population value. Your code will look something like:

```
proc univariate data=MyEst_no_violation  /* NO semicolon here */
   location = &beta_0 &beta_fat &beta_protein
                  &beta_complex_carbs &beta_sugars;
   var intercept fat protein complex_carbs sugars;
   histogram / normal;
run;
```

    Notice that the order of the population values in the *location=* option must match the order of the variables in the *VAR* statement (why?).

    You will get a separate section of output for each of the variables on the *var* statement. Be sure you understand what is is being computed here! Go back to the definition of a sampling distribution.

    Part of the output is labelled as *Test for Location ....* What is being done here?

    What is the function of the *HISTOGRAM* statement?

    Do you understand the output? Think carefully of what is being shown and how it illustrates a sampling distribution, the distribution of an estimator, a variation of the central limit theorem, bias, etc.

    What would you see in the histograms if the least-square estimator is BIASED?

    What does the std deviation of the estimator over the simulated datasets measure? Why?

Now you can see why we use macro variables for the parameter values - we only need to change them at the start of the program and the program is automatically updated.

12. It is a real pain trying to get the results from the histogram in *Proc Univariate* into an rtf file. It is much easier to recreate the histogram (and the fitted normal curve) using *Proc SGplot* and the *histogram* and *density* statements within *Proc Sgplot*. Create the histogram for the sampling distribution of the estimated slope associated with grams of fat. (Only do this for one of the slopes – the slope associated with grams of fat).

    You will use this histogram in your report (see below).

    You should add a vertical reference line for the population slope using the macro variable defined earlier. Hint: use something like:

```
refline &beta_fat / axis=x lineattrs=(thickness=10)
        label='Population slope';
```

    in your *Proc SGplot*. What do the various options on the *refine* statement do? Again, notice the use of the macro variable.

13. Of course, with only 5 simulations, you can't learn very much. Go back and increase the number of simulations to 1000 and rerun the above code. What does this tell you?

14. Now let us see what happens when the assumption of normality of the residuals is violated. Duplicate your code from the above and change all the dataset names that contain *no_ violations* to now include *violations* to keep the two studies separate.

    The log-normal distribution is a skewed distribution whose LOGARITHM follows a normal distribution. Read about its properties at: `http://en.wikipedia.org/wiki/Log-normal_distribution`.

    The *SAS* function *RAND('lognormal')* generate a log-normal random variable whose mean is exp(.5) and whose **VARIANCE** is $(\exp(1)-1)\exp(1)$. Because the mean is NOT zero, and because the standard deviation of this function cannot be specified, we need to standardize the log-normal random variable to have the correct mean and standard deviation to match the normal distribution used in the NO violation study. Think carefully why we need to standardize to a mean of 0 and adjust the standard deviation to match the previous simulation?

    So if a random variable W has a mean of m and a standard deviation of s, what is the mean and standard deviation of the random variable

$$Z = \frac{W - m}{s} \times \sigma$$

The residuals are suppose to have a mean of 0 and a standard deviation of σ so we need some intermediate steps. Replace the section of code where you added a normally distributed residual with the following. **CAUTION: I gave you the VARIANCE of the log-normal distribution above**

```
mu_calories = ....;
W = rand('lognormal');
epsilon = (W - xxxx)/ xxxx * xxxxx;
calories = mu_calories + epsilon; /* add the noise to the mean */
```

Do you understand the intermediate steps? Why do we need to "transform" the log-normal distribution to have the same mean and variance as when the assumptions are all true.

15. Run this simulation now with 1000 replicates.

16. Compare your results from the simulation with NO violation to the simulation with the violation of the assumption of normality. So are your results consistent with the following statement taken from Wikipedia `http://en.wikipedia.org/wiki/Least_squares`:

> "In a linear model in which the errors have expectation zero conditional on the independent variables, are uncorrelated and have equal variances, the ... unbiased estimator... is its least-squares estimator. ... The assumption of equal variance is valid when the errors all belong to the same distribution.
>
> However, if the errors are not normally distributed, a central limit theorem often nonetheless implies that the parameter estimates will be approximately normally distributed so long as the sample is reasonably large. For this reason, given the important property that the error mean is independent of the independent variables, the distribution of the error term is not an important issue in regression analysis. Specifically, it is not typically important whether the error term follows a normal distribution."

In particular be VERY clear that the sample size is NOT the number of simulated datasets. What is the sample size in this simulation?

Hand in the following using the electronic assignment submission system:

- Your *SAS* code that did the above analysis.
- A PDF file containing all of your *SAS* output.

- A one page (maximum) double spaced PDF file containing a short write up on this analysis explaining the results of this analysis suitable for a manager who has had one course in statistics. You should include the following:
  - A short, 3 or 4 sentence description of the simulation process.
  - The plot of the sampling distribution of the slope associated with grams of fat when the assumption of normality is satisfied and when the assumption of normality is not satisfied. **BOTH PLOTS SHOULD HAVE THE SAME SCALE ON THE X AXIS. YOU MAY HAVE TO RERUN YOUR CODE WITH THE AXIS LIMITS SPECIFIED.** By having a common axis, it is MUCH easier to compare the sampling distributions under the two simulations.
  - A table showing the mean of the slope associated with grams of fat from both simulations along with the standard error of the estimate (NOT OF THE MEAN) (from the simulation study) and the p-value from the relevant test of hypothesis. Each you will get slightly different results because each of you used a different seed (didn't you?).

  You will likely find it easiest to do the write up in a word processor and then print the result to a PDF file for submission. Pay careful attention to things like number of decimal places reported and don't just dump computer output into the report without thinking about what you want.

Part 2 - Road Accidents with Injury - Intermediate

We return back to the road accident database. What is the influence of external factors (e.g. speed and urban/rural) on the odds of a fatality?

In this part of the assignment, you will learn how to:

- how to recode variables
- how to run and interpret *Proc Genmod*

Again consider the accident database. We are now interested in the impact of speed and area (urban or rural) on the probability of fatality. We don't actually have the speed of the vehicles involved in the collisions, but we do have the speed limit which will serve as a surrogate for the actual speed (why?). Generally speaking the use of surrogate (or proxy) variables leads to interesting stastistical problems in more complex analyzes.

Similarly, urban vs. rural is ill defined. For example, in the lower Mainland of British Columbia, do you define Gaglardi Drive coming up to SFU – is the urban or rural? How do you define parts of Surrey as urban or rural? We don't have information on how the urban/rural designation was created in the database, so we will use it as it. Again, one of the hallmarks of a statistician[3] is to question any data values in data bases asking exactly how it was collected and defined.

1. Download the accident *.csv file into your directory.

   From the accident dataset, you will need the AccidentID, the accident severity, the speed limit, and the urban/rural variables.

   Look at the code book available on the web as noted in other assignments and use the codes for the above variables to define several derived variables.

   Define a *fatal* variable taking the values yes/no as you did in a previous assignment.
   Define a *area* variable taking the values *urban* or *rural*. Any accidents with other than urban/rural should be discarded.
   Define a *speed* variable taking the values *00+-30*, *30+-50*, and 60+ depending on the speed limit, where, for example, the recoded value *30+-50* represents speed limits higher than 30 km/hr and less than or equal to 50 km/hr.

[3] As opposed to a trained monkey

Remember to use good coding practices and NEVER assume that your input data is 100% correct!

2. Print out the first 10 records of the dataset to ensure that you've read the data properly and did the coding properly.

3. *Proc Tabulate* is your friend! Use *Proc Tabulate* to check your coding by creating tables that take the original variable and cross classify it by the coded variable. For example, part of the *Tabulate* procedure will look something like:

```
table accident_severity, fatal*n*f=comma7.0;
```

   Don't forget to specify the *missing* option on the procedure statement (why? - a good exam questions)

   How do you use these table to check your derived variables?

4. Delete any record with missing values in any of the *fatality*, *area*, *speed*, variables.

5. Return to *Proc Genmod* and run a model that look at the log(odds) of fatality as a function of the *speed* and *area* variables. Your model will look like:

```
proc genmod data=****;
   class .....;
   model fatal = area speed area*speed / dist=*****   type3;
   lsmeans area*speed / ***** adjust=tukey ;
   ods output .....;
run;
```

   Refer to a previous assignment for details on *Proc Genmod*.

   What option on the *lsmeans* statement will covert the logodds back to the probability scale?

   What are your factors? What are their levels? Is each factor continuous or categorical? If a factor is categorical, what statement do you use to indicate this? The terms in the model correspond to the main effects of each variable and their interaction. An interaction occurs if the effect of a factor (the difference in fatality rates between levels of the factor) are not consistent across the levels of the second factor. Draw a prototype plot by hand (i.e. no date is necessary) of how the probability of fatality might vary as a function of speed with a separate relationship for urban and rural areas when there is and when there is not an interaction between the two factors. (Hint: refer to the section on profile plots in Section 10.1.1 in my CourseNotes).

Don't forget to model the log(odds) of a FATALITY, i.e. of the *yes* category. You may have to use the *Descending* option on the *Proc Genmod* statement.

Extract the odds for each area and speed combination using the appropriate *ODS OUTPUT tablename=mydsname* statement.

6. Look at the results of the Type 3 tests. What do you conclude. Are you surprised at any of the results? If the p-value associated with the interaction term is small, it indicates that there is evidence the relationship between the proportion of fatality and speed is different among the two areas. What do you conclude? What do the other lines in the Type III test table represent and what do you conclude?

7. Look at the estiamated "lsmeans". (Recall that because the response variable is a categorical response, the values report are NOT means.) Do you understand the output?

8. Make a "nice" table with suitable headers, decimal places, etc. for use in your final report.

9. Create a plot (using *SGplot*) of the log-odds of fatality vs. speed with a separate series for each area (hint: use the *group* option on many of the statements within *Proc SGplot*. Join up the points for each area. DO NOT put a regression line on the plot.

   Make a similar plot of the probability of survival rather than the odds.

   Here is one example of why the analysis on the logods scale is preferred over the analysis on the probability scale. Look at the plot on the logodds scale. The relationship between the logodds and speed limit is almost parallel (how can you tell? - remember the function of the the confidence intervals on the plots). This would indicate a consistent difference in the logodds of a fatality between urban and rural areas and would lead to a simple conclusion. which is???

   However, the plot on the probability scale does not look parallel at all – a more difficult interpretation.

   Why do you think that the probability of a fatality is different in rural areas vs. urban areas even after adjusting for speed. How is this seen on the plots?

Hand in the following using the online submission system:

- Your *SAS* code.
- A PDF file containing the the output from your *SAS* program.

- A TWO page (maximum) double spaced PDF file containing a short write up on this analysis suitable for a manager of traffic operations who has had one course in statistics. You should include:
  - A (very) brief description of the dataset.
  - A graph showing the log(odds) of fatality vs speed for urban and rural areas.
  - A table of same.
  - A brief explanation of how to interpret the graph and any interesting findings

Part 3 - Why you should check residuals - Challenging

Read http://scienceblogs.com/notrocketscience/2009/04/03/monkey-see-monkey-calculate-statist
By now I hope you aren't one of the t*a**ed m**k**s as part of this study.

I've mentioned several times the importance of model assessment in Statistics. One important part of model assessment is looking at residual plots.

In this part of the assignment, you will learn how to:

- create interesting residual plots.

Let us begin.

1. Download the dataset *interesting1.csv* from the website at http://www.stat.sfu.ca/~cschwarz/Stat-340/Assignments/Assign05/interesting1.csv.
2. Read in the dataset using the (very) uninteresting variables Y, $X1$, etc.
3. Use *Proc SGscatter* to create a scatterplot matrix of all variables. What do you conclude based this plot? (Note this is not the *SGplot* procedure.)
4. Use *Proc Reg* to do a multiple regression of Y vs. all of the X variables. Use the $OUTPUT\ OUT=$ statement to output the predicted values along with the residuals.

   What do you conclude based on the output from *Proc Reg*? Are you surprised given the results of the scatterplot matrix?
5. Plot the residual against the predicted value. Use -2 to +2 for the range of the X axis. What do you conclude?
6. An interesting article on how to do this is found at:

   http://www4.stat.ncsu.edu/~stefanski/NSF_Supported/Hidden_Images/Residual_Surrealism_TAS_2007.pdf and http://www4.stat.ncsu.edu/~stefanski/NSF_Supported/Hidden_Images/stat_res_plots.html.

   Who says that Statisticians don't have fun!
7. You may also want to repeat the above "analysis" on the *interesting2.csv* and *interesting3.csv* files in the same directory.

Hand in the following using the online submission system:

- Your *SAS* code.
- A PDF files containing the output from your *SAS* program.
- The residual plot and your interpretation of the plot.

Comments from the marker

Part 01 Cereal

- General problems with simulation procedure.
- Not stating the assumption that is violated and the distribution of the error term under the violation
- Giving results of wrong hypothesis test. Lots of students reported the p-value for the test of $H_0 : \beta_{fat} = 0$ instead of $H_0 : \beta_{fat} = 9$.

  Notice that you are interested in testing if the estimator for the slope remains unbiased in the presences on an assumption violation. Hence you want to test if the mean of the estimator equal the actual slope (9) used to generate the data.
- No legends for plots
- The writeups were either really good, or completely incomprehensible, no consistent middle ground here. In marking this I looked for a clear explanation of what questions the simulation was trying to answer. Most students mentioned a thing or two about bootstrapping, but this is NOT a bootstrapping experiment.
- A number of students clearly didn't use enough replicates in the simulation to get good results.
- Another mistake I saw a few times was not comparing the experiment results with the *standard* values, but rather with the values obtained from the regression model from the previous question.
- Finally, I penalized papers that didn't state what the conclusion of the experiment was. I wasn't too demanding with this – as long as they mentioned that the resulting estimate was unbiased and that CLT has "kicked in" in some way, they got full marks.

Part 02 Accidents

Here the marks were quite a bit more spread out. Common things I penalized for:

- The main problem here was the interpretation of the results if they were counter-intuitive.
- Some confidence intervals were inverted

- Not interpreting any of the results
- Not explaining what they were trying to measure, or at least mentioning that new variables were created.
- Posting relative frequency plots and claiming that they were, indeed, odds ratio plots.
- Posting plots without reasonable explanation of how it could be interpreted.
- Drawing incorrect conclusions from the plots. ALWAYS check your output carefully to understand what is being produced!

Part 03 Interesting

This one was more or less all or nothing – if the plot showed the quote, people got full marks. I gave partial marks to one or two people whose plot had reversed axes, and the quote didn't appear properly due to the way *SAS* plots stuff. Residuals are ALWAYS plotted along the Y-axis!