Stat-340 – Assignment 4 – 2015 Spring Term

Part 1 - Breakfast cereals - Easy

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

- do simple bootstrapping to estimate the SE and confidence interval of a statistic
- use a simple macro variable to control number of iterations
- interpret the bootstrap sampling distribution

We will start again with the basic cereal data. Read the information about the dataset and breakfast cereals from
`http://www.stat.sfu.ca/~cschwarz/Stat-340/Assignments/Cereal/Cereal-description.pdf`

1. Download the cereal dataset from
   `http://www.stat.sfu.ca/~cschwarz/Stat-340/Assignments/Cereal/cereal.csv`
   and save it to your computer in an appropriate directory.
2. Input the data into *SAS* as in Assignment 1.
3. Deal with the coding for missing values as in Assignment 1.
4. Use *Proc Univariate* to estimate the mean, standard deviation, and the standard deviation using the Gini robust procedure (see below) of the number of calories per serving over all cereals.

   Use *SAS* to compute the standard error for the sample mean and 95% confidence interval for the population mean. Hint: refer to previous assignments on how to do this. What is the difference between the standard deviation and the standard error of the sample mean? Interpret the 95% confidence interval for the population mean.

While the formula for the standard error of the sample mean when data is collected using an SRS is well known, the formula for the standard error of the sample standard deviation and 95% confidence interval for the population standard deviation are less well known but could be computed (`http://web.eecs.umich.edu/~fessler/papers/files/tr/stderr.pdf`). Yes, standard errors of sample standard deviations do exist – think carefully about what this represents! *Proc Univariate* computes a 95% confidence interval for the population standard deviation, but does not give you a standard error for the sample standard deviation. What is the difference between the sample standard deviation and the standard error of the sample standard deviation? Interpret the 95% confidence interval for the population standard deviation.

Proc Univariate also gives you estimates of other statistics, but does NOT provide estimates of the standard error of these other statistics nor any confidence intervals for the corresponding population parameter.

For example, the usual estimate of the standard deviation is EXTREMELY sensitive to outliers. Various robust methods of estimating the standard deviation have been proposed, among them the Gini estimate of the standard deviation – see `http://support.sas.com/documentation/cdl/en/procstat/63104/HTML/default/viewer.htm#procstat_univariate_sect031.htm` for details on its computation.

You can get the Gini estimate of the standard deviation by specifying the *robustscale* option on the *Proc Univariate* procedure invocation.

5. There are many other statistics that you could compute where it is unclear on how to compute a standard error or a confidence interval. Bootstrapping is one method that can be used to estimate standard errors and confidence intervals in a wide range of situations.

   CAUTION: **This assignment illustrates how to do a simple bootstrap in the case of a SRS. If the sample or experimental design is more complex, the the bootstrap method must be modified to account for the sampling design.** You will learn how to do bootstrapping for most complex experimental designs or surveys in more advanced courses in statistics.

   For a VERY brief overview of the bootstrapping procedure, read `http://en.wikipedia.org/wiki/Bootstrapping_(statistics)`. Statistics Canada has a nice paper on how to bootstrap in complex surveys at `http://www5.statcan.gc.ca/bsolc/olc-cel/olc-cel?lang=eng&catno=11-522-X200600110416`.

   As outlined in class, the idea behind a bootstrap when data are collected using a SRS design, is to resample, **with replacement**, from the sample, compute the statistic on this resampled set, repeat these two steps many times, and look at the distribution of the statistic over the bootstrap samples. The distribution of the statistic over the bootstrap samples should mimic the distribution of the statistic over repeated samples from the population (why?) and hence the standard deviation of the statistics over the bootstrap samples should be an estimate of the standard error of the statistic (why?). **This last point is THE CRUCIAL aspect of bootstrapping and, inter alia, is also the FUNDAMENTAL CONCEPT OF STANDARD ERRORS in general**.

6. We will start by generating the bootstrapped samples using *Proc SurveySelect*. Your code will look something like:

   ```
   %let nboot=5; /* number of bootstrap reps */

   proc surveyselect data=cereal out=bootsample
           method=urs outhits rep=&nboot samprate=1 seed=2342332;
   run;
   ```

   All computer generated random numbers rely on a complex mathematical calculation that takes the current value of a random number and then generates the next value in sequence. In fact, the computer generated random numbers are actually NOT random at all because the same sequence of values will always be generated. Refer to `http://en.wikipedia.org/wiki/Random_number_generation` for a brief explanation of how computers generate pseudo-random numbers. This sequence of random numbers must be initialized with a number that is traditionally called the *seed*. If you rerun the program with the same seed value, you will get the same sequence of random numbers which makes it easier to debug your program. But then for your final run, you should replace the seed by a suitable value (check *SAS* documentation.)

   What does *method=urs* and *outhits* do? Why do we want *samprate=1*?

   What does the *%let nboot=5;* do? Why do we debug our program with a small value for the number of bootstrap samples? Notice how we refer to the macro variable in subsequent statements through the use of *&nboot*.

   In general, this is good programming practice to create macro variables for things like the number of simulations rather than hard coding them in your code. This way, all you have to do is change one statement in your *SAS* code and the new value is then automatically used in the rest of your script.

7. Look at the output from *Proc SurveySelect*. How many observations were read in and how many observations were created? Does this make sense?

   Print out the first 20 records. Notice what happens if the same cereal is selected multiple times.

   Also notice that the *Replicate* variable has been added to the dataset to "group" the records belonging to the same bootstrap sample together.

8. Tabulate the number of observations in each replicate using *Proc Tabulate*. Hint: your table statement will look something like

   ```
   proc tabulate **** missing;
      class ****;
      table replicate, n*f=5.0;
   ```

   Does the output make sense?

9. Now we need to compute our statistics for each bootstrap replicate and output the results to a dataset. By now, you should be able to "guess" the general structure of the code:

   ```
   proc univariate data=bootsamp noprint;
      by *****;
      var *****;
      output out=bootstat
             mean=mean_calories
             stddev=std_calories
             std_gini=gini_std_calories;
   run;
   ```

   What is the purpose of the *NOprint* option? The usefulness of this should become clearer when we increase the number of bootstrap samples to 1000.

10. Print out the first 10 records of the *bootstat* dataset. Do you understand what this dataset represents?

11. We now find the standard deviation, 2.5^{th} and 97.5^{th} percentiles of the SAMPLE MEANS from the replicated bootstrap samples. Why? Your code will look something like:

    ```
    proc univariate data=bootstat noprint;
       var *****;
       output out=SE_boot_mean
                  stddev=SE_boot_mean
                  pctlpts=2.5 97.5  pctlpre=cl_mean;
    run;
    ```

    What does the *pctlpts=* option do? What does the *pctlpre* option do?

    The above method of finding the 95% confidence interval for the POPULATION mean based on the bootstrap sampling distribution is a simple (but naive) way to do this. There are better methods that you will learn about in more advanced classes in statistics.

12. Print out the *SE_ boot_ mean* dataset. Compare the standard error of the sample mean from the bootstrap estimates and the confidence limit for the population mean from the bootstrap estimates to the standard error of the sample mean and 95% confidence interval for the population mean computed from the values computed by *Proc Univariate* at the start of this assignment. Of course, with only 5 bootstrap samples, the *se* and 95% confidence interval from bootstrapping are not very reliable.

    Increase the number of replicates to 1000 and see how these bootstrap values now compare to the original estimates from *Proc Univariate* which are based on formulae.

13. We also want to display our bootstrap sampling distribution of the sample mean. Use *Proc SGplot* to display a histogram and a kernel density smoother of the individual bootstrap values of the sample mean. Hint: you did something like this in earlier assignments.

    What is the difference between a sampling distribution (that you just created above) and the histogram of the calories per serving? THIS IS A CRUCIAL CONCEPT TO UNDERSTAND IF YOU WANT TO BE A STATISTICIAN!

    What do you notice about the shape of the sampling distribution above? Why does it have this shape? Hint: what does the Central Limit Theorem say about sampling distributions? What is the asymptotic distribution of Maximum Likelihood Estimates as the sample size increases?

14. We would like to annotate our histogram with the confidence limits from the bootstrap sample. We could take the values and manually add reference lines though the use of the *REFLINE* statement in *Proc Sgplot* but then we would have to manually change this every time we reran the program. We would like to annotate the graph automatically.

    Full details are available at `http://support.sas.com/resources/papers/proceedings11/277-2011.pdf`. You will get a crash course here. Note, the process of annotating plots is NOT part of the course and you won't be responsible for it on an exam, but I want to show you the power of *SAS*.

    Place the following code segment BEFORE your *SGplot*.

```
data sgannods;
   set SE_boot_mean;
   function= 'line';
   y1space='datapercent'; x1space='datavalue';
   y2space='datapercent'; x2space='datavalue';
   x1=cl_mean2_5;  y1=0; x2=cl_mean2_5;  y2=50; output;
   x1=cl_mean97_5; y1=0; x2=cl_mean97_5; y2=50; output;
run;
proc print data=sgannods;
```

```
   title2 'annotation instructions';
run;
```

This creates instructions (annotation instructions) for *Proc SGplot* to add things to the graph. Here I want vertical lines stretching from the top to 1/2 up the graph showing the 2.5^{th} and 97.5^{th} percentile (which correspond to the 95% confidence interval - why?). The *function* statement says that I want to draw a line; the *xxSPACE* statements give the type of co-ordinate system to use to draw the line, i.e. either the actual data values for the X axis or percentage of the data area in the Y (vertical) direction. Finally I give the (x_1, y_1) and (x_2, y_2) co-ordinates for the start and end of the line.

To apply these annotations to the plot, modify the first statement to read

```
proc sgplot .... sganno=sgannods;
```

where *sgannods* is the name of the annotation dataset.

15. Repeat this for the sample standard deviation and the robust (Gini) standard deviation. Hint: You don't have to replicate the entire program, just repeat the code after you compute the *SE_ boot_ mean* dataset - why?

    If you were doing this in a real world situation, you would write a macro to do this replicated code. While this is not part of the course, have a look at the final solutions (next week) to see how this could be done.

16. Create a table (by hand, but good for you if you can do this in *SAS* through various merges etc.) showing

    - The three statistics (mean, standard deviation, Gini standard deviation) computed from the original dataset along with the reported SE (only available for the mean), and reported 95% confidence interval (only available for the mean and simple standard deviation).
    - The bootstrap estimates of the standard error and 95% confidence limits for the three statistics.

    This table will form part of your report (below).

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 (very) brief description of the dataset.
  - Your table of results. In your write up, compare the SE of the mean computed analytically and via bootstrapping. Compare the 95% CI for the mean and standard deviation computed analytically and via bootstrapping. Discuss the shape of the sampling distribution.
  - If you have space, include your graphs of the sampling distributions.

  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 - American Time of Use study - Intermediate

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

- recode data and check your recoding;
- summarize data in different subcategories;
- do an ANCOVA to see if changes over time are the same across groups;
- use the *group* option in *Proc SGplot* to get different lines for different groups.

We return to the American Time of Use study from Assignment 1.

1. Download the time of use *.csv file into your directory.
2. Read in the ATUS summay data file as you did in previous assignments.
3. Compute the derived variables for the total time watching TV and an indicator variable representing if the person watched TV during the selected day.
4. The data file has a variable recording the sex of the respondent. This is a numerical code (check the code book on my web site in the *atusint-codebk0313.pdf* file. Recode this numerical code to a new variable with the values *male* or *female*.
5. The data file has a variable recording the age of the respondent - check the code book. Recode the continuous age variable into another variable representing the following age categories: 25 year or less; 26 to 35 years of age; 36 to 50 years of age; 51 to 65 years of age; 65+ years of age.
6. Use *Proc Tabulate* or *Proc SGplot* to check your recoding.
7. Compute the mean time spent watching TV for each year-sex combination using a weighted mean as you did in Assignment 1. Print the first few records. We CANNOT compute a standard error these means using the simple formulae available in *Proc Means* because the design is NOT a CRD/SRS. It would be possible to compute a standard error using a variant of a bootstrap approach, but this is beyond the scope of this course. Read Section 7.5 in the User Guide for details.

8. Plot the mean time watching TV over time for each sex on the same graph. Use the *Group=* option in the appropriate *Proc SGplot* statements. Join the points for each sex (hint: use a *series* statement), and draw a regression line for each sex (hint: use the *reg* statement in *Proc SGplot*). Based on this plot, what would you conclude?

   Note that the default action of *SAS* is to differential the groups using color, shape, and line type. In general, using color is not recommended because people who are color blind cannot distinguish the different colors, and color does not photocopy well. It is almost always preferable to label the individual lines (rather than using a legend) and use more distinct symbols and line types. This can be done in *SAS* but is beyond the scope of this course.

9. *Proc GLM* can be used to compare slopes between groups. For example, is there evidence that the change in mean TV viewing over time is different between males and females? Use the weighted means and the following code fragment:

```
proc glm data=.....;
   title2 'test if the slope is the same for males and females';
   class sex;
   model meantv = sex tuyear sex*tuyear;
run;
```

   Notice that the model statement now has a classification variable (*sex*), a continuous variable (*tuyear*), and an interaction term between the two variables (*sex*tuyear*). This model is called Analysis of Covariance (ANCOVA) to distinguish it from ANOVA (comparing means among groups), regression (computing trends over time).

   Despite each year's data being collected in a complex sampling design, it is appropriate to use the CRD/SRS methods on the weighted means to look for trends over time. This is an example where the experimental unit for the trend analysis is the mean for that year-sex combination, but the observations were taken on individual people within the year-sex combination, i.e. pseudo-replication. A standard method of dealing with pseudo-replicates is to compute an appropriate summary statistic prior to the analysis at the year level; in this case, the weighted mean.

   Look at the Type III test section of the output and look at the p-value for the interaction term. If the p-value associated with this term is small, it indicates that there is evidence the slopes are different among the groups. What do you conclude? What do the other lines in the Type III test table represent and what do you conclude?

   You will learn more about ANCOVA in more advanced courses in your career. Note there is a potential problem with this analysis – if you look at the scatter of the plots over time, there is a high correlation between the residuals between the sexes. For example, if one sex has a point above the trend line in a year, then the other sex also tends to have a positive residual, and vice-versa. There also appears to be long-term cycles as the points move above and below the line in a systematic pattern. This is evidence of non-independence which violates one of the assumptions of ANCOVA. The p-values would need to be adjusted for this violation of assumptions, but this is beyond this course. [It turns out that such an adjustment doesn't change the conclusions of this simple analysis.]

10. Repeat the above, except this time use age class as the categorical variable rather than sex. What do you conclude?

    This analysis has similar problems with non-independence as noted earlier and a more complex analysis could be done in more advance courses.

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 (very) brief description of the dataset.
  - An explanation of your findings.
  - A comment on the potential problems in the analysis.
  - The two plots of the trends over time by sex and by age class.

  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 03: Review and preparing for term test

In this part of this assignment, you will work on a few short exercises designed to review some of the material from the first three assignments and introduce some new things about the *Data* step.

Put all of the code from all of the sub-parts in one single *SAS* file. There is NO writeup for this part of the assignment.

1. **Problems with input data**
   Outdoor temperature is measured in degrees Celcius (`http://en.wikipedia.org/wiki/Celsius`) in Canada and degrees Fahrenheit (`http://en.wikipedia.org/wiki/Fahrenheit`) in the US. Look at `http://www.stat.sfu.ca/~cschwarz/Stat-340/Assignments/Assign04/assign04-part03-ds01.txt` which has the temperatures in degrees Celcius or degrees Fahrenheit in a few cities in early January.

   Read in the data and convert all temperatures to degrees Celcius. Note that
   $$C = (F - 32) * \frac{5}{9}$$
   Print out the final dataset that contains the city and temperature (1 decimal place) but not the observation number. **Make sure that the label for the temperature indicates it is in degrees Celcius**.

2. **Column input**
   So far you have used the *list* input style of *SAS*. In this style of input, variables are separated by at least one delimiter (typically a blank) and there is no requirement that data values be aligned in the input file.

   In some cases (particularly in dealing with data that was collected many years ago), space on the input record was at a premium and data was often "crunched" together without spaces between values. For example, an old style of input medium was the punch card (`http://en.wikipedia.org/wiki/Punched_card`) in which you had at most 80 columns of data.[1]

   Look at `http://www.stat.sfu.ca/~cschwarz/Stat-340/Assignments/Assign04/assign04-part03-ds02.txt`. The first two records are the variable names and a character "counter" so you can see where the various columns in the data are. The data variables are:

   - Make of car in columns 1-5.

[1] As a historical note, my first *SAS* programs were coded using punch cards using a IBM System 360.

- Model of car in columns 6-12.
- Miles per gallon (mpg[2]), Imperial measurement of fuel economy, in columns 13-14.
- Weight of the car (lbs[3]) in columns 15-18.
- Price of the car ($) in columns 19-22.

In *column input*, you specify the columns that contain the variable. For example, to read in the make and model of the car, your code would look something like:

```
data *****;
   infile *****;
   length make ***** model ****;
   input make $ 1-5  model $ 6-12;
```

Write *SAS* code to read in all of the variables from the car data (using the URL method) and print out the final dataset.

3. *Split-Apply-Combine (SAC) paradigm*
The *Split-Apply-Combine (SAC)* paradigm is a common task, implemented in *SAS* using the *By* statement, various procedures, and the *ODS OUTPUT* or OUTPUT OUT= commands within the procedure.

For example, suppose you wanted to compare the average amount of sugar by shelf in the cereal data. Your code would look something like:

```
data cereal;  /* define the grouping variable */
    /* read in cereal data */
    /* make sure shelf and sugar are defined */
 run;

proc sort data=cereal by shelf; run; /* sort by grouping variable */
proc univariate data=cereal  ....  cibasic ;
   by shelf;  /* separate analysis for each shelf */
   var sugars;
   ods output .....;
run;

proc sgplot data=....;  /* plot the estimate and 95% ci */
   ....
run;
```

You've also analyzed the proportion of survival by passenger class in the Titanic, the number of accidents per day across the months, etc.

[2]There are 1.6 km in a mile, and 4.54 L in 1 Imperial gallon. The US gallon is smaller than the Imperial gallon. A conversion calculator is available at `http://mpg.webix.co.uk`

[3]There are 2.2 lbs (pounds) in a kilogram

Go back to the accident dataset, and make a side-by-side confidence interval plot of the proportion of fatal accidents by MONTH. You will have to create a month variable from the date, create a fatality indicator, use *Proc Freq* or *Proc Genmod* to estimate the proportion of fatalities in each month, and finally *Proc SGplot* to plot the final estimates and confidence intervals.

Are you surprised by the results? There is NO write up for this part.

Commments from the marker.

Here are the comments from the marker from previous years assignments.

Part 01 - Cereal

- Many students did Not give a brief description of the analysis. Most papers began with a quick blurb about the dataset itself, followed by a detailed discussion of the mean, sd, gini, etc – without even a mention of the fact that it was calories/serving that was the variable of interest. Most people who lost marks on this question lost it because they didn't mention the variable they were analyzing.

- Many students did not interpret the results (brief comparison of analytic and bootstrap estimates)
- Other mistakes included not using enough replicates in bootstrapping. A number of people used only 5 replicates, and deemed it sufficient. Some even presented histograms from those bootstrap runs, and claimed that things looked "normal".

  Use 5-10 replicates to TEST your program, but don't forget to increase the number of replicate bootstrap samples to around 1000.

Part 02 - Time of Use

This is new this year, and so there are no comments available.

Part 03 - Snappers

- When converting to celsius, keeping a 'c' appended to the values so that they were still character variables.
- Not properly parsing the car variables