Stat-340/341 – Assignment 9
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
- interpret the bootstrap sampling distribution

There are a number of web pages that give a quick introduction to doing bootstrapping in *R*:

- `http://www.statmethods.net/advstats/bootstrapping.html`
- `http://www.mayin.org/ajayshah/KB/R/documents/boot.html`
- `http://statistics.ats.ucla.edu/stat/r/library/bootstrap.htm`
- `http://dist.stat.tamu.edu/pub/rvideos/Bootstrapping2/Bootstrapping.html`

We will start again with the cereal data.

1. Read the information about the dataset and breakfast cereals from
   `http://www.stat.sfu.ca/~cschwarz/Stat-340/Assignments/Cereal/Cereal-description.pdf`

2. 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.

3. Input the data into *R* as in previous assignments.

4. Deal with missing values as in previous assignments.

5. Download and install the *boot* package in *R*.

   The main bootstrapping function is *boot()* and has the following format:

   ```
   bootobject <- boot(data= , statistic= , R=, ...)
   ```

   where *data* is the raw data (data frame, vector, or matrix), *statistic* is the FUNCTION that computes the statistic of interest on the resampled data; and *R* is the number of boot strap samples.

   The *boot()* function calls the *statistic* function *R* times. Each time, it generates a set of random indices, with replacement, from the integers *1:nrow(data)*. These indices are used within the statistic function to select a sample. The statistics are calculated on the sample and the results are accumulated in the *boot object* which is returned.

   Once you generate the bootstrap samples, the *print(bootobject)* and *plot(bootobject)* functions can be used to examine the results. If the results look reasonable, you can use the *boot.ci()* method to obtain confidence intervals for the statistic(s).

   Look at some of the fore-mentioned web pages for examples.

6. We want to find the sampling distribution of several statistics for which the standard error is not easily found using analytical methods:

   - the median
   - the sample standard deviation
   - the Gini estimate of the standard deviation. 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.

7. Create a function *my.est(data)* that computes the sample mean, the sample median, the sample standard deviation, and the Gini estimate of the standard deviation from a VECTOR of data and returns a vector or a data frame of the results with separate elements corresponding to these values.

Normally you would place this function in a separate file and source it in, but for this assignment, include the function definition directly in your code.

The value of G used in the Gini standard deviation starts with all possible pairwise differences of the individual data points. This can be easily computed in R using a generalized outer “product” (the *outer()* function). For example look at the results of

```
X <- c(1,2,3)
Y <- c(4,5,6)

outer(X, Y, FUN="-")
     [,1] [,2] [,3]
[1,]   -3   -4   -5
[2,]   -2   -3   -4
[3,]   -1   -2   -3
```

Do you see what the *outer()* function does? Hint: the (i, j) entry in the matrix that is created is what function of the i^{th} entry of the first vector and the j^{th} entry of the second vector? The *outer()* function will apply the function in quotes to every combination of elements in X and Y.

So what does *outer(X,X,FUN=’-’)* do? (Notice that both input vectors to the *outer()* function are now the same). When we do this on the same vector (X):

```
outer(X, X, FUN="-")
     [,1] [,2] [,3]
[1,]    0   -1   -2
[2,]    1    0   -1
[3,]    2    1    0
```

Notice that we now have an almost symmetric matrix of the differences between X and itself.

How does this help you compute the Gini standard deviation? We need the absolute value of the differences – what function in R do you think will give the absolute value of its entry?

Next we need the sum of the absolute values of the differences. How do you think this will be computed?

After finding the sum of the absolution value of the differences over the entire matrix of results, you will have 2× the sum of the absolute values of the differences that the Gini requires. Why? Review the formula for the Gini estimate carefully. What does $i < j$ mean under the summation sign mean and what happens if you find the sum of the entire matrix of absolute value of the differences?

There is a *choose()* function that may help you calculate the binomial coefficient if you are having trouble calculating it yourself. But read the documentation as well, please!

Hint: Don't forget the $\sqrt{\pi}/2$ term. Note that R has the value of π in the automatic variable *pi*[1].

Test your function and you should get the following:

```
> my.est(cereals$calories)
      mean med       sd      gsd
1 105.0649 100 21.62013 20.33536
> my.est(cereals$sugars)
      mean med       sd      gsd
1 7.026316   7 4.378656 4.485241
```

Hint: You did deal with missing values in your function, didn't you? You can name the elements of a vector by using something along the lines of

```
X <- c(first=27, second=343, blah=2343)
X
```

8. Use your function to estimate the mean, median, standard deviation, and the standard deviation using the Gini robust procedure of the number of calories per serving and the amount of sugar/serving (as above).

   Use these value to estimate the standard error and 95% confidence interval for the mean of both variables using the standard formula and the t-distribution. The *qt()* function will be helpful. 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. Yes, standard errors of sample standard deviations do exist – think carefully about what this represents!

   What is the difference between the sample standard deviation and the standard error of the sample standard deviation (if you knew it).

   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 or a CRD. If the sample design or experimental design

[1] Happy Pi day on March 14 of last week. The year 2015 has a special Pi Day that won't ever be repeated – read `http://en.wikipedia.org/wiki/Pi_Day` for details.

is more complex than an SRS or CRD, the the bootstrap method must be modified to account for the sampling/experimental design. You will learn how to do this in more advanced courses in statistics.

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.

9. The *boot()* function in *R* does all of the heaving lifting for us. We need to modify our function *my.est()* slightly, because the *boot()* function wants to send TWO arguments to our function. The first argument will be the original data, and the second argument will be a vector of indices that indicates which data will be resampled on each call.

   Modify your function above as follows:

   ```
   my.est <- function(data, indices){
       data <- data[indices]  # this does the actual selection of the bootstrap sample
       ... rest of code
   }
   ```

   BE SURE YOU UNDERSTAND what the INDICES vector looks like and what it is doing!! You function MUST return a vector and NOT a data frame (groan...).

   So now I generate my bootstrap results using:

   ```
   my.boot.results.calories <- boot( cereals$calories, statistic=my.est, R=1000)
   ```

10. The returned results is a list object with 2 elements. Do a *str()* on the results.
    List the *t0* object – what are these values?
    List the first 10 records of the *t* object – what are these values?

11. The returned object is an object of class *boot* and has several methods defined for it. For example, use the *print()* method – what does this show you?

12. The *plot()* method will require an index to indicate which statistic you created should be plotted. Try the following

```
plot(my.boot.results.calories, index=1)
title(main='Calories Sample Mean', line=3)
```

Read the help file for title to see the purpose of the *line=3* option, and/or try the statement with and without the argument.

BE SURE YOU UNDERSTAND WHAT IS BEING SHOWN! What is the difference between the SAMPLING DISTRIBUTION and the HISTOGRAM of the raw data? 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?

Get plots for the other statistics that you produced.

Why do the results for the median look odd?

Compare the sampling distributions for the sample standard deviation and the Gini standard deviation. What do you conclude?

13. Notice that the Base *R plot()* function automatically "recognized" that the object being plotted was returned from the *boot()* function and so it knew what types of plots to make. How does this happen?

    Run the following command:

    ```
    class(my.boot.results.calories)
    ```

    This returns the value of "boot" which indicates that it was created by the bootstrap functions. When an object has a particular class (e.g. *boot*, or *lm*, or *glm* etc.), there may be a method for that class of object. Usually it is named *plot.boot* or *plot.lm*, or *plot.glm* etc., i.e. the function name with a period (.) and then the class of the object.

    You can see the methods associated with an object using the *methods()* function.

    ```
    > methods(class=class(my.boot.results.calories))
    [1] c.boot*     plot.boot*  print.boot*
    ```

    You see that the *plot()* function has a specific variant for the *boot* class of objects.

    You can then use

    ```
    help(plot.boot)
    ```

    to find the help file for this specific variant of the *plot()* function.

14. Get the confidence intervals for each statistic using the *boot.ci()* function. Like the *plot()* method for bootstrap objects, the *boot.ci()* function requires an index argument. You can also specify the method to compute the confidence levels using something like:

    ```
    # Sample mean percentile ci from bootstrapping
    boot.ci(my.boot.results.calories, index=1, type="perc")
    ```

    The above method of finding the 95% confidence interval 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.

    Compare the estimated standard error and 95% confidence intervals computed using the standard normal theory for the sample mean to those from bootstrapping. How do they compare?

15. Repeat the above for the sugars/serving.

    If you were doing this in a real world situation, you would write a function to do this replicated code.

16. Create a table showing

    - The statistics of interest (mean, median standard deviation, Gini standard deviation)
    - The bootstrap estimates of the standard error and 95% confidence limits for the statistics.

    You will find that it is difficult to extract the information from the bootstrap object[2] as it uses a function to compute the values on the fly. Your best bet is to access the *$t* component in the bootstrap object directly, and then find the *sd()* (why?) of each column (hint, use the *aaply()* function along the second margin).

    Similarly, when you compute the bootstrap confidence intervals, the *$percent* components returns 5 elements being the confidence levels, the two order statistics where the confidence interval endpoints are located, and then the upper and lower confidence limit. You will have to extract the last two components. There doesn't appear to be a simple way to find all of the confidence limits for all the statistics simultaneously, i.e. the *index* argument only accepts a single value.

    You will find it easiest to "hard code" the extraction for the four statistics, but if you are really keen, you can use a member of the APPLY function family[3] – see the solutions for details.

[2] Sigh..., R is free, but not cheap.

[3] Sigh..., R is free, but not cheap.

Hand in the following using the electronic assignment submission system:

- Your *R* code that did the above analysis.
- An HTML file containing all of your *R* 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 for amount of sugar/serving. This should have the sample mean, sample median, sample standard deviation, and the Gini estimate of the standard deviation. For the sample mean, you also have the *se* and confidence intervals computed using a formula, but these are not available for the other estimates. Finally, include the bootstrap estimates of the *se* and the upper and lower bootstrap confidence limits for ALL four statistics/parameters.

    Compare the *se* of the mean computed analytically and via bootstrapping. Discuss the shape of the sampling distributions.
  - If you have space, give 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 - Road Accidents with Injury - Intermediate

We return back to the road accident database. The database is a (near) complete census of accidents in Great Britain in 2010. However, census information is expensive to collect and in many cases a sampling approach is preferable. Sampling is, of course, the raison d'etre of Statistics!

A common exercise for a statistician is to predict the amount of sampling required. What this usually entails, as a first step, is to provide a graph of the precision of the estimate as a function of sample size. Then one can decide the tradeoff in spending money to reduce the standard error, i.e. how much is information worth?

In some cases, this is relatively easy to do because the standard error of an estimate exists in closed form. What happens if there is no formula for the standard error. Again this can often be done using a simulation method.

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

- how to select samples from a population
- compute an index from each sample
- find the *se* of this index from replicate samples generated using the *rdply()* function in the *plyr* package.
- plot the *se* as a function of sample size
- see if the results on your plot above is consistent with generally accepted principles.

Again consider the accident database.

1. Read in the accident data into *R*.

   Create a variable, the AccidentIndex defined as the product of the number of vehicles and the number of casualties divided by the accident severity. Think about what a large value of the AccidentIndex implies vs. lower values of the AccidentIndex.

   Don't forget that you will have to process the imported data to deal with missing values in some of the codes.

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

3. The *sample()* function selects random samples by returning the INDEX to the items being selected. Read the help file on this function. Select a random sample of size 10 from the accident data frame, without replacement using the *sample()* function using:

   ```
   my.sample <- sample(1:nrow(radf), size=10, replace=FALSE)
   my.sample
   ```

   BE SURE YOU UNDERSTAND WHAT IS BEING RETURNED!

   The *sampling* package has many more extensive functions for many other types of sampling (e.g. cluster sampling, probability proportional to size sampling (pps), etc).

4. Write a function that takes the road accident data frame and the sample size as arguments and returns a vector containing the sample size, the mean and the 90^{th} percentile of a sample of the accident index values (without replacement). Your function should start something like:

   ```
   AI.stat <-  function( radf, sample.size){
          my.sample <- sample(1:nrow(radf), size=sample.size, replace=FALSE)
          sampled.acc.index <- radf$AccidentIndex [......]
       ... your code here ...
   }
   ```

   Hint: read the help on the *quantile()* function.

   For example, here is what my function returns at various sample sizes – your mileage will vary.

   ```
   > AI.stat(radf, 10)
   sample.size        mean       per90
         10.00        1.05        2.20
   > AI.stat(radf, 10)
   sample.size        mean       per90
    10.0000000   0.7666667   1.1000000
   > AI.stat(radf,100)
   sample.size        mean       per90
       100.000       1.225       2.000
   > AI.stat(radf,100)
   sample.size        mean       per90
   100.0000000   0.9566667   2.0000000
   ```

5. The *plyr* package has several functions that repeatedly apply a function and combines the results into a data structure. The *rdply()* runs an expression multiple times, and returns a dataframe of the results (assuming that the expression returns a vector or data.frame result).

For example, consider

```
my.result <- rdply(5, AI.stat(radf=radf, sample.size=10, .id="rep"))
my.result
```

What does the '5' do as the first argument of *rdply()*? What does the *.id* argument do?

6. Create a new function that takes the road accident data frame, the sample size, and the number of replicates of that size to draw. For each replicate, this new function should call your *AI.stat()* function to generate the the mean and 90th percentiles. After all the draws are completed, compute the standard deviation over the replicated samples of the mean and of the 90^{th} percentile of the AccidentIndex– this represents the approximate standard error for each statistic (why?). Return the sample size, the *se* of the mean and *se* of the 90th percentile. Hint: There is no need to use a formula for the *se*'s – what statistic can be used to compute thee *se*'s given repeated values of the statistic?

   DO YOU UNDERSTAND WHAT THE FUNCTION IS SUPPOSED TO COMPUTE AND WHY?

   You function should start something like:

```
AI.sim <- function (sample.size, radf , nsamples=1000, ){
# Generate nsamples of size=sample size from radf
# For each sample, compute the mean and per90 of the accident index
# After all samples are generated, compute the sd of the mean and per90 values
#
   my.result <- rdply(nsamples, AI.stat(radf=radf, sample.size=sample.size). .id=....)
   ..... do further computations here
}
```

   Note that the FIRST argument must be the sample size as this will passed to this function from an *plyr* function later on in the assignment.

   Here are some results of my calls. Your mileage will vary[4].

```
 AI.sim( sample.size=20, radf=radf,  nsamples=100)
sample.size     se.mean    se.per90
 20.0000000   0.2481723   0.9555620
> AI.sim( sample.size=20,  radf=radf, nsamples=100)
sample.size     se.mean    se.per90
 20.0000000   0.2427101   0.4300768
> AI.sim( sample.size=200,  radf=radf,  nsamples=100)
 sample.size      se.mean     se.per90
200.00000000   0.09941868   0.29723001
```

[4] `http://en.wiktionary.org/wiki/your_mileage_may_vary`

```
> AI.sim( sample.size=2000,  radf=radf, nsamples=100)
 sample.size      se.mean     se.per90
2.000000e+03 3.083609e-02 9.486833e-02
>
```

DO YOU UNDERSTAND WHAT IS BEING COMPUTED!!! Notice that I've used *nsample=10* to override the default of 1000 samples to speed up my computations for debugging.

There are multiple types of "sample size" being used here – you need to keep them straight!

7. We now want to call our *AI.sim()* function with different sample sizes to investigate the performance of the *se* of the two statistics over a range of sample sizes.

   We will want to call the *AI.sim()* function with sample sizes 100, 200, 400, 1000, 2000 and 4000. Naive users of *R* would like to use a loop, but Rexperts will use a member of the *plyr* family of functions.

   The *ldply()* function take a list, and for each entry of the list, calls the appropriate function. If the input data is not a list, it is converted to a list. For example, see what happens when you use:

   ```
   ldply(c(100,200), AI.sim, radf=radf, nsamples=1000)
   ```

   Read the help files on the *ldply()* function. In this function, and in many other functions, you will an argument denoted by 3 dots $(\cdots)$. This indicates additional arguments that are passed to the function being called by *ldply()* over and above the first argument, which in our case is the the size of the sample being selected from the accident database.

8. Expand the above *ldply()* call for the the sample sizes of interest (100, 200, 400, 1000, 2000 and 4000). Your final result at this point should have the structure:

   ```
   Sample size    se_p90   se_mean
   100             xxx         xxx
   200             xxx         xxx
   400             xxx         xxx
   ....
   ```

9. In many cases, standard errors decline as a function of $\sqrt{n}$. We want to see if that is true about the mean accident index and the 90th percentile of the accident index. Create derived variables for the $\log(sample\ size)$, $\log(se\_p90)$, and $\log(se\_mean)$.

10. Fit a line to the relationship between the $\log(sample\ size)$ and the $\log(se)$ for the two statistics. What do you conclude by looking at the slope (and confidence interval for the slope) about the relationship?

11. We want to plot the relationship between the log(se) of each statistic and log(n).

    There are several ways to plot multiple Y variables against the same set of X values. The most common way is to convert a dataset from the wide- to the long-format, and then use the *group* option in the *ggplot()* function to plot separate lines and points and get a nice legend.

    Read `http://stackoverflow.com/questions/3777174/plotting-two-variables-as-lines-using-g` for an illustration of this. The *reshape2* package has the *melt()* function which converts from wide-format to long-format. You can read about it at `http://seananderson.ca/2013/10/19/reshape.html`. Your code will look something like:

    ```
    my.result2 <-melt(my.result, id="sample.size")
    my.result2

    > my.result2
       sample.size variable      value
    1          100  se.mean 0.11309656
    2          200  se.mean 0.08186335
    ...
    7          100 se.per90 0.34125325
    8          200 se.per90 0.30256027
    ...
    ```

    The *variable* column has the "name" of the curve. Now use *ggplot*:

    ```
    ggplot(data=my.result2, aes(x=log(sample.size), y=log(value),
            group=variable, color=variable, shape=variable, linetype=variable))+
        geom_point()+
        geom_line ()
    ```

    We also want to add the two fitted lines to the plot - use the *geom_ smooth(method="lm", se=FALSE)* layer to the plot above.

    Don't forget to add suitable titles and axes labels. Then save the plot using the *ggsave()* function.

    What is the approximate slope of the line between the two variables? Look at the estimated slope. Is the decline in standard error with sample size consistent with the $\sqrt{n}$ rule? How did you tell?

Hand in the following using the online submission system:

- Your *R* code.
- An HTML file containing the the output from your *R* program.

- A one 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 decline in $\log(se)$ as a function of $\log(n)$ with an accompanying table of the fit and an explanation of the implications of the slope when investigating the improvement in precision as a function of sample size.

Part 3 - Cigarette Butts in Bird Nests - Challenging

Download and read the following paper where the effect of cigarette butts on the parasitic load in bird nests was investigated.

> Suarez-Rodriguez, M., Lopez-Rull, I. and Garcia, C.M. (2013).
> Incorporation of cigarette butts into nests reduces nest ectoparasite load in urban birds: new ingredients for an old recipe?
> Biological Letters 9, 20120931.
> `http://dx.doi.org/10.1098/rsbl.2012.0931`

The authors have provided the raw data as an electronic supplement to this paper. I've downloaded the data and you can access it from `http://www.stat.sfu.ca/~cschwarz/Stat-340/Assignments/BirdButts`. This is an Excel workbook with two worksheets corresponding to the analyses in Figure 1 and Figure 2.

We will try and reproduce some of the results from this paper. You will find, rather surprisingly, that some of the results in the published paper are INCORRECT (!). This is NOT an unusual occurence in scientific papers – often authors use inappropriate statistical methods to analyze their data and this is not noticed in the peer review process.

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

- use *R* to read Excel spreadsheets directly;
- use summary functions to examine standard deviations and balance in experimental designs;
- write a model for a two factor CRD ANOVA; notice that *R* gives you incremental rather than marginal sums of squares
- write a model for a two factor CRD ANCOVA;
- "reverse engineer" an analysis in a paper to try and figure out what was done.
- use the *t.test()* function
- how to specify subsets of observations in analyses using the *subset=* argument

- extract information from modeling functions.

Let us begin.

1. Review the how the experiment was conducted to count the number of parasites per nest. Look at Figure 1, especially the insert in the top right of the figure.

2. Install the *xlsx* package into *R*, start the package using the *library()* command, and read the help file on the *read.xlxs()* function.

   Import the data from the first sheet in the workbook. Use code similar to:

   ```
   library(xlsx)
   ndf <- read.xlsx('nests.xls', sheetName="Correlational",
               header=TRUE, stringsAsFactors=FALSE)
   ```

   Notice that this function uses the *stringsAsFactors* argument rather than the *as.is* argument to prevent converting strings to factors. There is NO consistency among *R* packages![5] Also notice how *R* converts "illegal" variable names (e.g. names with spaces) into legal *R* variable names.

   Print out the first 10 records to ensure that the data has been read properly.

3. There are two factors in this first study – host species and nest content. We wish to examine the effect of two factor, *Species* and *Nest Content* on the MEAN number of parasites in the nests.

   How many levels are in each factor? Hint: If you don't remember the definition of a level, read Section 4.1 of my course notes at `http://www.stat.sfu.ca/~cschwarz/CourseNotes`.

4. Create a simple summary table comparing the number of nests measured by host species and nest content in much the same way as in previous assignments. The code will look something like:

   ```
   xtabs(....)
   ```

   Is the design balanced? What would the counts look like if the design was balanced?

   Unbalanced designs are usually NOT a problem for modern software, especially for simple designs. You can run into problems in more complex designs, so that is why it is a good reason to look at the balance of your design before any analysis.

[5] R is free but not cheap.

5. Next, you should construct what is known as an *Interaction Plot* or a *Profile Plot* for the average number of number of mites/nest and how it depends on the two factors.

   Use the *ddply()* function to compute the sample mean and 95% c.i. for the population mean number of mites for each combination of host species and nest content. Hint: look at some of the functions you created in earlier assignments. Your final result should look something like:

   ```
   > report
     Species Nest.content     mean       lcl      ucl
   1    HOFI       chicks 47.57143 17.300764 77.84209
   2    HOFI         eggs 31.70000 10.832903 52.56710
   3    HOFI        empty 18.58333  9.547109 27.61956
   4    HOSP       chicks 24.71429  3.331111 46.09746
   5    HOSP         eggs 31.33333  8.955982 53.71068
   6    HOSP        empty 30.41667 14.271278 46.56206
   ```

   Now create the interaction plot using *ggplot*. If you don't know what an interaction plot is, read Section 10.1.1 of my course notes.

   ```
   plotInt <- ggplot(data=report, aes(x=Nest.content, y=...,
          group=Species, color=Species, linetype=Species))+
     geom_point()+
     geom_line()+
     geom_errorbar(aes(ymin=..., ymax=...), width=0.2, position="dodge")+
     ggtitle(...)+
     ylab(...)
   ```

   What does the interaction plot show you? What is the purpose of *position="dodge"* argument?

6. Before conducting any analysis, you must ensure that you understand the experimental design. The most common error made by people who are not true statisticians is to simply run a computer package against the data without understanding the design. For example, they would use a computer package to analyze a blocked design without specifying that it was blocked design, and so would get wrong answers!

   There are many ways in which you can run a two factor experiment. Some of the designs that you will learn of in more advanced classes in statistics are a two-factor CRD; a RCB; a fractional factorial CRD; a Split-Plot with CRD at upper level; Split-plot with RCB at upper level; etc. Consequently, as noted before, it is important to understand the design before you do the analysis!

   What are the attributes of a completely randomized design as applied to a two factor design? Hint: Read section 10.1 of my course notes at `http://www.stat.sfu.ca/~cschwarz/CourseNotes`. How do these apply to this experiment?

7. The *lm()* function is the standard procedure to use in relatively simple experimental designs with a single size experimental unit and all units independent of each other. The *lme()* or *lmer()* functions is typically used for more complex experimental designs.

   Create variables for SPECIES and NEST.CONTENTS that are both factors! Did you want to make one of these variable an ordered factor?

   Your model in *lm()* will look something like:

   ```
   Number.of.mites ~ SpeciesF + Nest.contentF + SpeciesF:Nest.contentF
   ```

   Notice that interaction terms in *R* are specified using the colon (:) to separate the factor names and not an asterisk (*) as used by *SAS*.

   Also use the *lm()* function with the same data, but now with the model having the terms in a slightly different order

   ```
   Number.of.mites ~ Nest.contentF +  SpeciesF +  SpeciesF:Nest.contentF
   ```

   And again, fit the *lm()* function with the same data, but now with the model having the terms in a slightly different order

   ```
   Number.of.mites ~ SpeciesF:Nest.contentF + Nest.contentF +  SpeciesF
   ```

8. Use the *summary()* function to display information about the three model fits. Are all of the results the same? They should be the same, because the same final model (i.e. main effects and interaction has been fit).

   Why is there only one term labelled as “Species’; only 2 terms labelled as “Nest.content”; and only 2 terms labelled as being interaction effects. What is the reference class for each set of terms?

   The t-tests seen in the *summary()* output are not very useful as they don’t test hypotheses that are very interesting.

9. Use the *anova()* function to display the F tests for the three models with terms in a different order. Notice that the results of the ANOVA are different depending on the order of the term, which seems odd.

   It turns out that *lm()* fits factors incrementally when finding the ANOVA table – you will cover this in more detail in more advanced classes in Statistics, e.g. Stat-350.

   The discussion of Type I vs III SS in R gets quite heated – for the R-community dogma on this issue, you should read Venables’ Exegeses on linear models at `http://www.stats.ox.ac.uk/pub/MASS3/Exegeses.pdf`

   Fortunately, the *car* package provides an extension to the *anova()* function that provides the necessary Type III SS. **CAUTION: You must change the way *lm()* fits a model BEFORE you use the revised *anova()***

function. This is NOT documented in the package and the only posts that discuss this are buried deep in a blog posting[6]

10. Download and install the *car* package and read the help on the (revised) *anova()* function which it calls *Anova* (notice the capital A). You can get the marginal sum of squares (equivalent to what *SAS* produces using

    ```
    cat("\n\nUse the Type III tests from the Anova() function from the car package")
    cat(  "\nbut you need to set the treatment contrasts to sum rather than treatment")
    cat(  "\nBEFORE fitting the lm() model!")
    cat("  \nSee http://r.789695.n4.nabble.com/Type-I-v-s-Type-III-Sum-Of-Squares-in-ANOVA-
    library(car)
    old.options <- options()
    options(contrasts=c(unordered="contr.sum", ordered="contr.poly"))
    options()$contrasts
    my.lm.model <- lm(Number.of.mites ~ ...}
    Anova(my.lm.model, type=3)
    options(old.options)
    ```

    Compute the marginal F-tests for all three models and notice that they now are the same.

11. Compare the *df* column from the Type III tests with those in the upper right corner of Figure 1 of the bird paper. Notice that the *df* for the main effect of species, the main effect of nest content, and the interaction terms match up, but the *F* values do not.

    Part of the reason that the *F* values do not line up, is that the authors adjusted the tests for the effect of species and nest content by the weight of the butts present in the nests. For example, perhaps one species tended to gather more butts, on average, that the other species. Then the difference in mean number of parasites might be caused by the differing number of butts, on average, in the nests and Figure 1 shows that butt weight does have an effect on the mean number of mites.

12. A model where you adjust the tests for factor effects by a covariates (weight of butts) is called Analysis of Covariance. You will learn about ANCOVA in more advanced courses in statistics. Modify the model statement in the *lm()* function along the lines of (it is necessary only to modify one of the models from this point onwards):

    ```
      Number.of.mites ~ SpeciesF + Nest.contentF+ SpeciesF:Nest.contentF + Butts.weight
    ```

    Notice that the variable *Butts.weight* is NOT declared as factor. An ANCOVA model has a mixture of types of variables – some are categorical (factors) and some are continuous.

[6] R is free, but not cheap.

Re-run the analysis, get the Type III marginal tests, and compare the degrees of freedom and the F statistics. We see that *df* for the factors match, but the *df* for *Error* (the term labelled *Residual*) doesn't match.

13. Hmm....Notice in our *lm()* output, the total of the *df* from error + all of the terms in the model add to 56, which is 1 less than the total number of nests analyzed. But in the table in Figure 1 of the paper, the total (excluding the *df* for the intercept term) is only 41. This indicates that they did NOT use all of the data!!

    Because the total *df* (excluding the intercept) must equal $n-1$, it appears that they only used 42 data points. But which 42 data points?

    Have a look at the spreadsheet. Notice that there appears to be additional data collected after the first 42 nests. Perhaps they only analyzed the first 42 nests in Figure 1 of the original paper?

14. Repeat the previous analysis only using the first 42 nests. You can easily do this using the *subset=* argument of *lm()*. Create a vector that takes the value of TRUE for the first 42 cases, and FALSE elsewhere and use this vector in the call to *lm()*.

    Hurrah!! Now all of the *df*, F-values, and p-values match the table in Figure 1 of the paper.

    Very weird that they only used 42 nests for the analysis in Figure 1 where as in the body of the text they used all the data (e.g. as you saw in a previous assignment, you verified the two-sample results). It is also weird that the graph has an asterisk beside the intercept indicating that the analysis was done on the log-scale, but in fact, the analysis reported in the table is on the anti-log scale.

15. Notice that your formal test of interaction failed to find evidence of an effect, but the interaction.plot created earlier appeared to show evidence of an interaction. Why the difference? How could you improve the earlier plot to avoid making this type of error in the future?

16. Create the diagnostic plots from the last analysis (refer to previous assignments on how to request this). It is pretty clear from the residual plots that the variance is not homogeneous and tends to increase with the mean. (Which diagnostic plots shows this and how do you know?)

17. Because the variance of the residuals is increasing with the mean response, this suggests that a log-transformation may be useful. In more advanced classes you will learn about Taylor's Power Law which tell you the appropriate transformation based on the relationship between the standard deviation and the mean. Repeat the previous analysis (based on ALL nests) but using $\log(number\ of\ mites)$. Do your conclusions change?

    But, examine the diagnostic plots. There is at least and possibly two outliers. Identify the outlier(s). Can you spot the outlier(s) in Figure 1 of the paper?

18. Repeat the previous analysis after the outlier(s) are removed - hint, once again use the *subset=* argument to remove the observation from the fit. Do your conclusions change?
19. Finally, create a corrected Table for Figure 1.
20. As a final note, we used ANOVA which assumes that residuals have a normal distribution. It is quite clear from the analysis that the variance depends on the mean. This, and the fact that the response variable (number of mites) is a smallish count, imply that a Poisson ANOVA should have been done. You will see how to use a Poisson ANOVA in the classes on discrete data analysis later in your studies.

Hand in the following using the online submission system:

- Your *R* code.
- An HTML files containing the output from your *R* program.
- A one page (maximum) double spaced PDF file containing a short write up of the results of the experiment. Which observation(s) were removed because they were an outlier. Tell your manager what was wrong with the results reported in Figure 1 of the paper. What are your final conclusions? Provide the final, corrected, table of effect test results similar in style to that seen in Figure 1 of the paper (but you can't get, and don't want the intercept line). This table should use log($mites$) as the response variable, remove the one or two outliers, and use all of the remaining nests (not just the first 42).

Whew! This is an example of the level of analysis and *R* coding that you should be able to do by the end of your undergraduate degree without extensive hand-holding that you get in these assignment instructions.

Comments from the marker

This assignment looks familiar, doesn't it? That's cause it's virtually unchanged from Assignment 4, except now you're expected to do your coding in R. But the rest is the same. And you have the solutions for Assignment 5, as well as my feedback for your writeups, and the benefit of weeks of being able to figure out what that assignment was all about. Thus, I hope you would excuse me if my marking was a bit harsher this time around.

Ornamental plots and data dumps. Despite my rambling soliloquies of the way statistics should be presented, these are still surprisingly common even now, at the end of the course. So, whenever you get a mark off for "ornamental plot" or a "data dump", what it means is – I have no idea why the plot/table you included is important, what features of it are important, and what conclusions I should be drawing from it.

I would also like to point out that when you're referencing the plot in your text, phrases like "it's clear from the plot that blahblahblah" are kind of dangerous. They come off as being patronizing and cocky, especially if the blahblahblah is not at all made clear to me by the plot. I know textbooks like to use this language, but that's cause they're written by patronizing and cocky people. To be safe, it's better to use the following phrase structure: "Note feature A and feature B of the plot: they are an indication of blahblahblah."

Avoid computer diarrhea when ever possible. It is rarely suitable to just dump computer output in to a report for someone. *SAS* has much nicer facilities for creating nicely formatted output compared to *R*, but you can use *Sweave* in *R* if you really want to directly integrate *R* output into a report. [You need to know LaTeX, a superior typesetting document compared to *MSWord*.]

Copying and "paraphrasing" Carl's assignment solutions. So, I already harped on the subject how copying without proper citation is bad. I would like to further point out that closely paraphrasing is equally bad, and it doesn't fool anybody who has read the original into believing that they're reading your own work.

In your university work, you're likely to come across a multitude of written documents where you think the author said something really smart, that you'd really like to repeat, but for whatever reason you don't want to cite verbatim. You would need to paraphrase, which means to explain in your OWN words. When that happens, this is what is the honest thing to do:

How to paraphrase: Read the original text, and understand it completely. That last bit is important. If you don't understand it completely, it's a good idea to ask. Then close the original text, go and explain what you learned to a friend, preferably a non-expert. I use my mom, cause she's the only non-expert willing to listen to me explain stuff. Then go back and write down your explanation. Cite the original author if necessary, and publish.

How NOT to paraphrase: Take the text verbatim, but break the phrases up into shorter ones. Replace all the commas with "ands", and all the "ands" with commas. If the text contains a list with points labeled as (a) and (b), label them as (1) and (2) instead, and use a different format. Add a few "therefores" to help with the flow. Selectively erase phrases you don't understand, but do keep the ones that sound the smartest. Change the font and publish.

Part 1 - Cereal

Some general comments from the marker:

- Many students lost marks for not comparing the standard errors computed analytically and using the bootstrap. (coded compare *se*'s)
- Many students lost marks for not reporting analytic standard errors and confidence intervals
- Tables that were difficult to read lost marks (esp. poorly formatted pasted R output)
- Some students interpreted this question as comparing the estimates for calories/serving and sugars/servings. This is not the case! The point of the question was to compare analytic and bootstrap estimates!
- A few times I got the bootstrap statistics for the mean, but none for the sample standard deviation or the Gini coefficient of the standard deviation. This kind of defeats the purpose of doing the bootstrapping, doesn't it? The CI and *se* for the mean can be computed splendidly using formulae, but there aren't any (simple) formulae for the standard error of the sample standard deviation (yes, this exists – think carefully about it). and the Gini estimate of the standard deviation, which is WHY we were doing bootstrapping in the first place! So, I took marks off whenever I didn't see measures of variability for these other statistics.

The bootstrapping is a COMMON method for finding *se* for estimators without closed form solutions. The raison d'etre for Statisticians is to find SE. Finding estimates is easy... the hard part is getting a measure of precision for an answer

- So, there are two sets of statistics for the mean: those from bootstrap and those computed analytically. I took marks off whenever I didn't see the statement "these are similar". Small statement, and pretty self-evident, but it's an important feature to note. If they weren't pretty similar, t means that maybe the SEs for the sample standard devaition and Gini estimate for the standard deviation are also off.

- A few times I got the phrase "bootstrap SEs are slightly smaller/larger than those computed analytically". This is true, and I didn't penalize this, but it's important to realize that while the two SEs will be close, and sometimes the bootstrap *se* will be larger, and sometimes it will be smaller, but whether it's larger or smaller is an irrelevant fact, and attention should not be drawn to it. Don't forget that the bootstrap estimates are based on a SIMULATION and that a different simulation set could lead to different answers.

  One important part of bootstrapping is to actually see if the standard formulae for standard errors computed either via closed form estimators or via the inversion of the hessian in maximum likelihood are actually close to the true variability of the estimator over repeated samples. For example, MLE assumes that your model is correct (e.g. in the titanic, you assumed that people's survival were independent of each other and that the relationship with age was linear etc.) Often failure of the distributional assumptions in the likelihood or failure of the independence assumption leads to unbiased estimators, but reported standard errors that are too small. On way to asses this is via bootstrapping.

  Note that minor differences between the bootstrap *se* and the analytical *se* are likely just sampling artefacts.

- I read a few submission with claims like "the bootstrap *se* is smaller than the analytic *se*, so the bootstrap estimate is more precise." This is incorrect and it boils down to the difference between statistical precision and the meaning of the word precise.

  The precision of a point estimate relates to its standard error. The larger the standard error, the lower the precision of the point estimate. This is because confidence intervals generated using a larger standard error will be large themselves, and hence we have have less evidence that the true parameter is actually close to our point estimate. In the cereal question, since the analytic and bootstrap standard errors are close to each other, we can say that the precision as calculated using the bootstrap is similar to the analytic precision.

However, we cannot say that the bootstrap standard error is more precise than the analytic standard error. Why? My interpretation of the phrase "x is more precise than y" is that x is more accurate than y. In this case, the bootstrap standard error cannot be more accurate than the analytic standard error because the bootstrap standard error is an approximation of the analytic (exact) standard error.

Note the difference in meanings, in the statistical context, of the words precision and precise!

Part 2- Accident dataset

General comments from the marker:

- Most of the deductions for this part were for missing captions.
- Other deductions were for estimates that were way off.
- Some otherwise excellent write-ups got points off because they didn't mention that the whole hullabaloo was about predicting the Accident Index, which is a new variable measuring yadda yadda yadda. If you see a comment "What is the variable of interest??" that person is you.
- OK, so here is where I was harsher than I was in Assignment 4. changes with sample size. In fact, that's the very reason you're asked to do a regression on a log-log scale: the slope of roughly -0.5 would mean that se decreases as a factor of $n^(- 0.5)$. In human language, it means to get half the se we need to quadruple n. In fact, Carl's posted solution for Assignment 4 contained more or less this very information.

  So, any write-up that said the equivalent of "precision increases with sample size" got points off. Any write-up that described that relationship as "as $\log(n)$ increases by 1, $\log(SE)$ decreases by 0.5" also got points off. Both statements are 100% correct, but they are entirely uninformative. The former is not exactly news. The latter makes no sense.

Part 3 - Nest data

Some general comments from the markers:

- This question was very well done over all, with most students finding that there is evidence of an effect of butts weight as well as species on mean log(number of mites)
- Students lost marks (and didn't see a change in their results from the paper) if they kept the same subset of the data as the paper used.
- Students also lost marks for poorly formatted tables.
- Some people stated that the original paper was flawed because they didn't remove outliers. I take issue with that: there is nothing wrong about NOT removing outliers. It's always good practice to make **note** of them, but whether to remove them or not is a tricky problem, and the solution is often not obvious. In many cases scientists have good reason for not removing them from the study.
- As statisticians, we are free to play around with the data as much as we want – remove data points, try different models, fit this way and that, even generate "fake" data to improve our results. The power to infer anything under the sun via selectively removing data points is certainly exhilarating, but the final decision should really rest with the scientists who perform the experiments and draw conclusions from them.