Stat-340 – Assignment 3 – 2015 Spring Term

Part 00: Real life vs. the life of a student

Learning to program in SAS is like learning to play an instrument. No amount of talking or demonstration by the instructor will help you learn to play the instrument – you have to do the actual practice.

The questions that appear in the assignment are designed to "mimic" the types of analyzes that you will do as a real statistician. As you can see, they are NOT the toy problems you find in textbooks where all the assumptions are satisfied, the objectives are clear, and 10 lines of SAS code will provide the answer. Real statisticians have messy data, objectives are often not clear cut, several attempts at an analysis are required, and you need to communicate the results to someone.

Similarly, real statisticians do not have someone that "checks" their work – you need to develop "self-checking" skills so that you don't do silly things. So practices such as using *MISSOVER*, printing out the first 10 records of files, using *Proc Tabulate* to check transformations, using graphs to view your results are all useful so you don't make a silly errors and loose credibility in the eyes of your employer.

A statistical package and written communication skills is also a key skill in getting a job after you graduate. For example, I recently visited the `monster.ca` website and here the results of several queries:

Searching for "statistician" - no listed jobs
Searching for "sas" - 56 jobs
Searching for "spss" - 19 jobs
Searching for "statistical package" - 13 jobs
searching for "data analyst" - 375 jobs a statistician can do these jobs
Searching for "regression" - 85 jobs
searching for "anova" - 0 jobs
searching for "written communication skills" 1000+ skills

Part 01: Review and preparing for term test

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

At least one of the questions (or a variant) from this part will be on the term test and/or final exam! You should try and "solve" the first two problems without copying and pasting past code so that you get practice in creating a *SAS* program from scratch as you would have to do in a testing situation. For the third problem, you need to understand how to use the *index()*, *substr()*, and *input()* functions to deal with unusual data.

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. You will notice that missing values will be generated because of problems in the data; it is NOT necessary to try and fix the problems in the data. However, you should understand why missing values are generated.

1. **Multiple records for each observation**
   Sometimes data spans more than one record in the input file. For example, look at `http://www.stat.sfu.ca/~cschwarz/Stat-340/Assignments/Assign03/assign03-part01-ds01.txt`. There is information on members of my family (not necessarily real values) with the (suggested) variable names on the first line, but the actual data values for each family member split over 2 records for each family member. You cannot use *Proc Import* to read the data into *SAS* because the data for one observation is on multiple records. Do you understand the difference between a *variable*, *observation*, and a *record*?

   If data fall on two (or more records), you simply use the slash (/) to tell *SAS* to move to the next record to continue to read variables. Notice that ALL observations must have the same number of records in the dataset. (It is possible to deal with cases where the number of records varies per observation, but this is not covered in this course). For example, here is a *SAS* code fragment:

```
data .....;
    infile ...;
    input v1 v2 / v3 v4;
run;
```

which reads variables *v1* and *v2* from the first record of the observation and then moves to the second record and reads variables *v3* and *v4*. All four variables will be placed on the SINGLE observation in the *SAS* dataset.

Write *SAS* code to reading the dataset at the previous URL (using the URL option to read from the web directly) and print the dataset. Compare your printed dataset to the original data to make sure that you have read it in properly. Do you understand the output and can you explain the "features" of the displayed output.

2. **Creating a plot of side-by-side confidence intervals**
Look at `http://www.stat.sfu.ca/~cschwarz/Stat-340/Assignments/Assign03/assign03-part01-ds02.txt`. This file has information on summer earnings for students in different faculties (made up data). The data consists of the faculty name, the sample size, the mean earnings/students, and the standard deviation. All earnings data are in dollars.

   Read in the data set using the *URL* option. In cases of a simple random sample, the estimated standard error of the sample mean is found

$$SE_{\overline{Y},SRS} = \frac{s}{\sqrt{n}}$$

   where n is the sample size and s is the sample standard deviation. **This formula is ONLY valid for simple random sampling from an infinite population and if the estimator is the sample mean. Different sampling designs and different estimators have different formula for the standard error.**

   Compute the estimated standard error of each sample mean using suitable data step commands. An approximate 95% confidence interval is found as $estimate \pm 2SE$ (this is true for virtually all estimators in large samples regardless of the sampling design). Find the upper and lower bounds for the confidence intervals.

   Create the standard side-by-side confidence interval plot as you have done in the last two assignments using *Proc SGplot*.

3. **Special numeric values.**
Sometimes, special codes are used in numeric data to represent special conditions. Look at: `http://www.stat.sfu.ca/~cschwarz/Stat-340/Assignments/Assign03/assign03-part01-ds03.txt` where you will find a set of data on the water quality at several dates (made up data). There are several variables, the date of the sample, the coliform level, the turbidity, and the low-level ozone readings.

Consider the coliform readings. In some cases, the readings were below the detection limit, For example the value "<100' indicates that the coliform level was less than the minimum count that could be detected (a value of 100). The detection limit can change over time. This is an example of censored data which is very common in ecological and medical studies.

With censored data, it is quite common to convert the input data into two variables. One variable is the value and the second variable is a censoring indicator variable. If the value is NOT censored, report the actual value and set the censoring indicator variable to "no". If the value is censored, report the detection limit, and set the censoring indicator variable to "yes".

In order to do this, we read in the values from the data records into a CHARACTER variable. We then examine the values of the character variable to see if the value includes a "<" symbol. If there is no symbol, we then convert to the value to a number and set the censoring indicator to "no". If there is a "<" symbol, we need to skip over the symbol, read the rest of the digits, and set the censoring indictor to "yes".

There are three VERY useful *SAS* functions to help you do this, the *index()*, *input()* and the *substr()* functions. Read the *SAS* help information on these functions. Be careful to read the help information on the *input()* FUNCTION and not the *input* statement. One of the examples for the *input()* function shows how to read in numbers that include commas. [Hint: This is a good exam question!]. Notice that the syntax of *SAS*'s *substr()* function DIFFERS from that of *C++* or *Java*.

We start by reading in the data as character using a code fragment similar to (don't read in the rest of the values yet):

```
data ....;
    infile ....;
    length ccoliformw$30.;  /* the character input coliform value */
    input SampleDate:yymmdd10.  ccoliform $;
    format SampleDate yymmdd10.;
run;
proc print data=...
   title2 'Water quality data before fixups';
run;
```

Notice that the character variable starts with 2 c's (i.e. character coliform) to remind me what I'm doing.

We now need to check if there is a "<" in the values of the *ccoliform* variable. The *index()* function returns a value of 0 if the searched for

string is NOT found, and the position of the searched for string if it is present. So we can use code:

```
data newds03;
   set ds03;
   length ccoliform  detectlimit $30.;
   if index(ccoliform,'<') > 0 then do;  /* a '<' is found */
      lc_coliform = "yes";  /* left censored coliform value */
      whereless = index(ccoliform,"<");
      detectlimit = substr(ccoliform,whereless+1);
      coliform = input(detectlimit,  f30.0);
   end;
   if index(cocoliform,"<") = 0 then do; /* a '<' is NOT found */
      lc_coliform = 'no';
      coliform = input(ccoliform, f30.0);
   end;
run;
proc print newds03;
   title2 'Water quality data after fixups';
run;
```

Let's look at the code in more detail. First consider the case where NO < is found and *index(ccoliform,"<")* returns the value of 0 – the second *if* block. In this case, the *input* function reads the value of the character variable *ccoliform* directly using the informat of *f30.0* which means that a number up to 30 digits long with typically 0 decimal points. [Notice that if the number actually contains a decimal point, it will override the 0 specified in the *f30.0* so there is NO danger in using this even with data that contains decimal places.]

Second, consider the case there a '<' is found, the first *if* block. The *index(ccoliform,"<")* returns the POSITION of the '<' (usually in the first position, but never assume that data is proper!). This is stored in the variable *whereless*. Hence the first digit of the detection limit must START at the next position (*whereless+1*) until the end of the input string. The *substr()* function EXTRACTS the data starting at the first digit until the END of of the *ccoliform* variable (that is the default action when the third argument of the *substr()* function is not given and puts this into the character variable *detectlimit*. (See also the first example in the help file on the *substr()* function). Then the *input()* function reads the detection limit value as a number.

Notice that I did NOT use the *if-then-else* style of coding. Generally speaking, try and AVOID the *if-then-else* style as it is too prone to logic errors. For example, consider the code block

```
if sex = 'f' then do;
   y= 1;
   end;
else do;
   y = 2;
end;
```

If the variable *sex* takes the value of 'f', then Y is set to 1. If the variable *sex* takes the value of 'm', then Y is set to 2, which is likely what the coder intended. But never assume that the data is correct. What happens if sex takes the value 'F' (i.e. an upper case *f*). Now Y also takes the value of 2, which is likely not what is wanted.

A safer code block is;

```
y = .; /* initialize to missing */
if sex = 'f' then do;
   y = 1;
end;
if sex = 'm' then do;
   y = 2;
end;
```

Now, if the variable *sex* takes a value other than 'f' or 'm', the value of Y will be missing which seems more sensible.

SAS has a *select* construct which may be useful in these situations, but this is not part of this course.

It is also easy to make other types of logic errors. Consider, for example, the turbidity variable (see below). It can contain either a "$<$" or a "$+$" and different actions must be done in each case. You may be tempted to use the code block:

```
if index(cturbidity, "<") >0 then do;
   Y = 1;
   some other  actions
   end;
if index(cturbidity,"+") >0 then do;
   Y=2
   some other actions
   end;
else do;
   Y=3
   some other actions;
end;
```

What is the value of Y if the variable *cturbidity* contains the value '<100'. You may be surprised to learn that $Y = 3$, and not $Y = 1$. What went wrong?

Because the variable *cturbidity* contains a '<', the first *if* block is executed and Y is set to 1. *SAS* checks to see if the *cturbidity* value contains a "+" (the second *if* statement). It does not, and so invokes the *else* clause which is not what is wanted.

A much safer code block is set variables to missing;

```
if index(cturbidity,"<") > 0 then do; /* contains a < */
   some actions;
end;
if index(cturbidity,"+") > 0 then do; /* contains a + */
   soem actions;
end;
if index(cturbidity,"<") = 0 AND
   index(cturbidity,"+") = 0 then do; /* no < or + */
   some actions if no < or +
end;
```

Don't forget to check your recoding using *Proc Tabulate* or *Proc SGplot* to make sure you done the coding correctly. For example, you could something like:

```
proc tabulate data=.... missing;
   class sex y;
   table sex, y*n*f=5.0;
run;
```

to make a table showing the value of Y for each value of *sex*.

Normally, you would then use the *DROP* statement to drop the intermediate variables *whereless* and *detectllmit*. And, *SAS*perts would put all of this into ONE expression and dispense with the intermediate variables.

Go back and again look at `http://www.stat.sfu.ca/~cschwarz/Stat-340/Assignments/Assign03/assign03-part01-ds03.txt`. The third column of values refers to turbidity. Turbidity (a measure of clearness of the water) can be censored on either sides (but not simultaneously on both sides). A value of '<100' indicates that the reading was less than 100. A value of '50+' means that the reading was more than 50. In both cases, the exact value is not known.

Use the above program as a template to read in and dissect the third column of values. Some hints to get you started:

- Again use a character variable to read the raw value
- If there is neither a '<' nor a '+', then the data value is not censored.
- If there is a '<', then you are back in the previous case.
- If there is a '+', then you want the digits BEFORE the position of the '+'.
- No values will have both a '<' and a '+' (it doesn't make sense).
- You should create TWO indicator variables for censoring (*lc_ turbidity* for **l**eft-**c**ensored and *rc_ turbidity*) for **r**ight-**c**ensored with appropriate values.

Finally, go back and again look at `http://www.stat.sfu.ca/~cschwarz/Stat-340/Assignments/Assign03/assign03-part01-ds03.txt`. The fourth column of values refers to the ground level ozone. Here you have a lower and upper bound separated by a dash ('-'). Now you want to read in those values, and create two new variables *lower_ ozone* and *upper_ ozone*.

4. **Split-Apply-Combine**. The *Split-Apply-Combine* paradigm is fundamental to may data analyses. Look at: `http://www.stat.sfu.ca/~cschwarz/Stat-340/Assignments/Assign03/assign03-part01-ds04.txt` where you will find final grades from two years of a course. The variables are *year*, *sex*, *name*, and *final grade*.

   Write *SAS* code to compute the average grade in each year. Put the final results into a nicely formatted table with nice headers. The table should include the year, the number of student, the average grade with at most one decimal place, and a lower and upper confidence limit for the mean. Hint: Use *Proc Means*.

   Write *SAS* code to compare the mean grades of males and females from the class. Put the results into a nicely formatted table with nice headers that includes the year, the estimated difference in the means between the two sexes, the standard error of the difference, the p-value for the test of hypothesis that the means between the two sexes are equal. Format the p-value using the *SAS* p-value special format - check the help pages for details. The estimated difference in means and its standard error should have one decimal place.

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.

There is no write up for this part.

Part 02 - Road Accidents with Injury - Intermediate

We return back to the road accident database. A small number of accidents involved deaths of one or more drivers in the accident. Does the sex of the drivers affect the fatality rate, e.g. does the fatality rate differ in two car accidents if two female, one female and one male, or two male drivers are involved?

Additional data on the sex of the drivers was extracted from `http://www.data.gov.uk/dataset/road-accidents-safety-data` and information about the sex and age of the driver is available at `http://www.stat.sfu.ca/~cschwarz/Stat-340/Assignments/Accidents/road-accidents-vehicles-2010.csv`.

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

- how to skip columns when reading in large datasets;
- summarize data and count the number of observations, and the number of missing values;
- delete observations from a *SAS* dataset;
- use *Proc Tabulate* to compute percents in a table;
- merge two datasets using a common variable;
- receive an introduction to *Proc Genmod*, an analogue to GLM for categorical data;
- extracti information from *SAS* output and making tables and plots.

1. Download the accident and vehicle *.csv files into your directory.
2. Read in the accident information. From the accident dataset, you will need the AccidentID, the accident severity, and the number of vehicles. A quick and dirty way to select only some of the variable in a dataset is to do something like:

```
data accident_info;
   infile ....;
   length dummy $1 accidentid $20;
   input accidentid dummy dummy dummy dummy dummy  /* skip 5 columns */
           accident_severity number_of_vehicles;
   drop dummy;
```

This will "reuse" the variable (*dummy*) to skip over columns we don't want, and then drop it from the final dataset;

Select only accidents where two cars are involved (e.g. exclude single vehicle accidents and accidents where more than two vehicles are involved).

Look at the *Accident_Severity* variable and create a derived variable *fatality* that takes the values of *yes* and *no* as appropriate.

Don't forget to deal with missing values (see the code book) appropriately.

3. Print out the first 10 records of the accidents datafile. Hint: Use *split="_"* on the *Proc Print* statement to make a more compact display.

4. It is always a good idea to check your recoding by using a code fragment similar to:

```
proc tabulate data=accident_info missing;
   class Accident_Severity fatal ;
   table Accident_Severity, fatal*n*f=comma8.;
run;
```

What does the *f=comma8.* do?

5. Examine and read in the vehicle information. Make sure that the accident id variable name is the same and it has the same length as in the accident information dataset. Use a *Data Step* and not *Proc Import* to read the data. All you need to keep are the accident id, the sex of the driver variable and the derived variable *female* (see below).

   Create a derived variable in the vehicle dataset (e.g. *female*) that is 1 or 0 if the driver of the vehicle is/is not a female. If the sex of the driver is unknown (or missing), also set the *female variable* to missing.

6. Print out the first 20 records of the vehicle information dataset. Hint: Use *split="_"* on the *Proc Print* statement to make a more compact display. Notice that you will have multiple records for the same accident if more than one vehicle is involved.

7. It is always a good idea to check your coding by using a code fragment similar to:

```
proc tabulate data=vehicle_info missing;
   class sex_of_driver female;
   table sex_of_driver, female*n*f=comma8.;
run;
```

8. We need to reduce the vehicle information dataset to get one record per accident. Use *Proc Means* to count the number of drivers in each accident, the number of female drivers involved in each accident, and the number of drivers with missing sex information by analyzing the *female* variable by accident id. Your code fragment will look something like:

```
proc means data=vehicle_info noprint;
   by accidentid;
   var female;
   output out=vehicle_summary n=ndriver sum(female)=nfemale
              nmiss(female)=nmisssex;
run;
```

What does the *nmiss()* option do in Proc Means? Print the first 10 records to ensure that you have summarized the data correctly.

9. Process the vehicle summary data set to select only those accidents with 2 drivers (why) and where the number of drivers with missing sex is 0 (why?). What assumption are we making about the missingness mechanism for the sex of the drivers?

10. Print out the first 10 records of the reduced *vehicle_summary* dataset to verify that everything is working properly. Be sure to cross check with the raw vehicle information.

11. We now want to merge the accident information and vehicle summary data by accident id using code similar to:

```
proc sort data=ds1; by CommonVariable; run;
proc sort data=ds2; by CommonVariable; run;
data both;
   merge ds1 ds2;
   by  CommonVariable;
run;
```

12. Print out the first 20 records of the merged dataset. Because we excluded some accidents (involving less than 2 or more than 2 vehicles), not all accident id's have matching ids. What happens in this case?

13. There are almost 100,000 records in the joined dataset, so manually scanning the entire data is simply not feasible, nor desirable. *Proc Tabulate* is your friend!. *Proc Tabulate* can compute percentages in tables, but the syntax is not obvious and it often takes some practice to get things correct.

    Try a code segment that looks something like:

```
proc tabulate data=mergeddata missing;
   class nfemale fatal;
   table nfemale, n*f=comma8. fatal*f=comma8.  fatal*pctn<fatal>*f=5.1;
run;
```

    What does this code do? What does the *comma8.* format do? What does the *pctn* keyword do? What do you conclude based on this table.

    You will want to dump this table out into RTF for the final report.

14. *Proc Genmod* is the "equivalent" to *Proc GLM* when looking at categorical data. We want to see if the fatality rates varies by the number of female drivers involved in the accident. You could use *Proc Freq* but, as you will see in a minute, *Proc Genmod* allows to get all the pairwise comparisons among the groups.

    Use the code:

```
proc genmod data=mergeddata descending;
  title2 'examine if fatality rate is the same across number of females';
  class nfemale;
  model fatal = nfemale/ dist=binomial link=logit type3;
  lsmeans nfemale / cl diff ilink oddsratio;
  ods output lsmeans=myodds;
run;
```

    This procedure models the ODDS of a fatal accident (similar to what happened in the Titanic dataset) and information is presented on the log-odds (log(ODDS)) for the individual groups, or the odds ratio (for the pairwise comparisons between the groups). For historical reasons, *SAS* still used the LSMEANS statement, and the output has column headers with the word MEANS, despite it NOT BEING A MEAN! (Confusing? Definitely).

    Look for the Type 3 test in the output – this tests that hypothesis that the odds of a fatality is the same regardless of the number of females involved in the accident. What do you conclude based on these results.

    Never report a naked p-value. If you detect an effect, where does the effect lie? Look for the output comparing the "Difference in female least square means". As noted above, these are NOT comparison of means, but of odds. Look at the companions of the odds of fatality when there are 0 females involved vs. 1 female involved. Because the estimated odds ratio is larger than 1 (and the 95% confidence interval does not cover 1), it indicates that the odds of a fatality are greater with fewer females involved.

15. The ODS statement dumps out the estimated odds of a fatality (along with a confidence interval) by the number of females. Print out this dataset. Again, despite being labelled as *mu*, it really is the proportion that is being reported (but you have to look carefully to see if this is the proportion of death or survival. Create a plot of the proportion (along with the 95% confidence interval) by the number of females involved in the accident.

    What do you conclude from this report?

    Dump this out to RTF for the report.

16. The results are pretty clear that the fatality rate is lower if more females are involved in an accident! Why do you think this results has occurred?

You can adjust for many confounding variables by doing a more sophisticated analysis using GENMOD. For example, you could try and adjust by the speed limit or age of the driver. You may see such analyses in more advanced courses.

Hand in the following using the online submission system:

- Your SAS code.
- A PDF file containing the the output from your SAS program.
- A one page (maximum) PDF file containing a short write up (minimum 1.5 line spacing) 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.
  - The summary table showing the number of fatalities and the percent of fatalities and the plot showng the odds ratio of fatality by number of females in the accidents. A brief explanation of your findings.

Part 03 - Cigarette Butts in Bird Nests - Challenging

We will continue with the analysis of the Cigarette Butts experiment that you started in Assignment 2.

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

- use *Proc Import* to read Excel spreadsheets directly;
- use *Proc Transpose* to transfer from long to wide data formats;
- use *Proc Univariate* for paired data;
- use *Proc Ttest* for paired data;
- use *Proc GLM* for a single factor RCB ANOVA;
- see that the above 3 methods give IDENTICAL results;
- merge data sets together to combine information from two different sources;
- extract information from *Proc Reg* for subsequent plotting

Let us begin.

1. Review how the experiment was conducted where a thermal trap was used to attract ectoparasites as outlined in the paper. Draw a picture of the experimental setup. This is known as a Paired Experiment. What is the primary advantage of a paired design vs. an unpaired design. How could have run this phase of the experiment as a completely-randomized-design (CRD)? (Hint: in a CRD, each nest would be measured only once, either using the smoked or unsmoked butt).
2. Download the Excel file from the previous assignment again.
3. Use *Proc Import* to import the data directly from the second worksheet, the *Experimental* worksheet. What does the *dbms=* option do? What does the *replace* option do? Why are these good practices?
4. Examine the log file to see how SAS creates the variable names. Print out the first 10 records to ensure that the data has been read properly.
5. Print out the first 20 records. Notice that there are 2 records per nest (why?).

6. We will now analyze this experiment in 3 (equivalent) ways. Note that because the response variable (number of ectoparasites) are smallish counts, a better analysis would use a Poisson distribution and generalized linear models, but we will use the standard normal theory ANOVA model.

   While the results from the three methods of analysis below are the same, it is sometimes easier to analyze paired data in one format than another format so the choice comes down to personal preference.

   We will first analyze the difference in the number of ectoparasites attracted in each nest. In order to analyze the difference, we need to get the number of mites attracted under the treatment and control settings onto ONE observation so we can compute a difference. This is called converting data from the LONG format to the WIDE format (why do you think it has this name?).

   Proc Transpose does this conversion for us. Try

```
proc transpose data=xxxx_long  out=xxxx_wide;
   by nest;
   var number_of_mites;
   id treatment;
run;
```

   Because you are using a *By* statement, don't forget to sort the data appropriately beforehand.

   Print out the first 10 records of the WIDE format dataset. Do you see what has happend? (This would make a good exam question.). What do you think would happen if not every nest has every value in the long-format, i.e. if some nests were missing the treatment or control group? (Again, a good exam question).

7. Compute the difference in the number of ectoparasites attracted ($experimental - control$) in the transposed dataset and print out the first 10 records so you can verify that the difference has been computed properly.

8. Before doing any statistical analysis, it is a good idea to have a look at the data to see if there are any outliers or unusual plots. Use *Proc SGplot* to draw a histogram of the difference with a superimposed kernel density estimate (see previous assignments).

9. We wish to test the hypothesis that

$$H_0 : \mu_{treatment} = \mu_{control}$$

   Do you understand what the μ's refer to? Why is it silly to test a hypothesis of the form

$$H_0 : \overline{Y}_{treatment} = \overline{Y}_{control}$$

The hypothesis of equality of two POPULATION means is equivalent to

$$H_0 : \mu_{treatment} - \mu_{control} = 0$$

or

$$H_0 : \mu_{diff} = 0$$

Why?

Use *Proc Univariate* on the difference variable to both estimate the difference in the means (along with a se and a 95% confidence interval (don't forget to specify the *CIBASIC* option)) and to test the hypothesis (use the *MU0* option). Don't forget to specify NEST as the ID variable so you can identify the extreme observations.

You can also get a normal quantile-quantile plot using *Proc Univariate*. Use the

```
qqplot diff / normal;
```

statement within the procedures. Look at the QQ plot. Why are the points "stepped"? What do you conclude from looking at the QQplot?

What is the test-statistic (the t-value), the estimated difference in the means, the SE of the estimated difference in means, the 95% confidence interval for the difference in the means, and the p-value for the test of the hypothesis.

Refer back to the scientific paper, in the paragraph just before section 4. The authors' state:

> Traps containing cellulose from the smoked butts attracted significantly fewer ectoparasites from traps with non-smoked butts ($F_{1,54} = 43.13, p < .0001$).

The F statistics with 1 degree of freedom is the same as the (t statistic)^2. Do you get this?

Note that the authors' conclusion is actually stated incorrectly. The proper way to report the results should be along the lines of:

> There is strong evidence that traps containing cellulose from the smoked butts attracted fewer ectoparasites ON AVERAGE from traps with non-smoked butts ($F_{1,54} = 43.13, p < .0001$) with an estimated difference in the means of 1.18 (SE 0.18) parasites.

Why is it necessary to add the phrase "ON AVERAGE" to this statement? Why should you ban the word "significantly" from your statistical vocabulary? Why is it necessary to report the estimated effect sizes along with the p-value?

10. We will now use *Proc Ttest* to conduct a Paired t-test. Refer to Section 6.4 in my course notes at `http://people.stat.sfu.ca/~cschwarz/CourseNotes/` for more details. Why is this a paired experiment?

    You will again analyze the widedata set using code similar to:

    ```
    proc ttest data=xxxx_wide;
       paired control*experimental;
    run;
    ```

    Proc Ttest gives much of the output we created separately when we analyzed the differences our selves. Compare the results to your earlier output. Do you get the same results?

11. Finally, we will now use *Proc GLM* to analyze the data using a randomized complete block analysis. Use code similar to

    ```
    proc glm data=expinfo_long plots=(residuals diagnostics);
       title2 'GLM for RCB analysis';
       class nest treatment;
       model number_of_mites = nest treatment;
       lsmeans treatment / cl diff adjust=tukey stderr ;
    run;
    ```

    Look at the Type III tests and at the line labelled as TREATMENT. You should get the same F value as in the paper. Find the estimated difference in the mean, its standard error, the 95% confidence interval etc. You should again get the same values as in the previous two analyzes.

    Output the marginal (LSmeans) to a dataset and create a plot of the marginal means (along with a 95% confidence interval) in much the same way as in previous assignments.

12. Does the difference in the number of mites attracted to the resistors depend on the weight of the butts in the nest. We will need to merge the information from the (wide) dataset containing the difference in the number of ectoparasites attracted with the dataset (from the previous assignment) containing the weight of butts in each nest.

    Import the first worksheet as you did in the previous assignment.

    In order to merge two datasets, you must have a variable that is common to both. In this case the variable is the nest id. Sort both datasets by nest number, and then merge them together using code similar to:

    ```
    data both;
        merge nestinfo xxxx_wide;
        by nest;
    run;
    ```

Print out the first 10 records of the merged dataset to see that it has been merged together properly.

13. Make a scatterplot of the difference vs. the weight of butts. What do you conclude based on the scatterplot?

14. Do a regression of the difference in the number of ectoparasites attracted vs. the weight of butts in the nests. What do you conclude and why? Is there any indication of problems in the diagnostic plots?

    Output the fitted values from the procedure and create a scatterplot of the difference vs. the weight of butts overlaid with the fitted line, and a 95% confidence interval for the line (see previous assignments).

Hand in the following using the online submission system:

- Your SAS code.
- A PDF files containing the output from your SAS program.
- A one page (maximum) PDF file containing a short write up (minimum 1.5 spacing) of the results of the experiment. Report the p-value for the test of the hypothesis, the estimated difference in the means and the se of the difference. Report the final fitted regression line (with measures of precision) of the difference vs. the weight of butts, and your conclusion from the regression analysis. Include both the plot of the regression model (difference vs. butt weight) and a side-by-side ci plot of the means (with ci's) for the experiment.

Whew!

Comments from the marker

Some general comments:

- Code style is getting better and better, but many students are missing easy marks by not having proper spacing, indentation, comments, and a header. In particular, comments are lacking. They need not be exhaustive but a few informative comments go a long way.
- Reporting 5 or more decimal places
- Having no name - writeups without names in the headers received 0.

Part 01

New this year, so no previous comments

Part 02 - Accidents

Some general comments from the marker:

- Lots of people did not have the correct table. I'm not sure what went wrong. In many tables, it looked like the first row (meant to be missing values) was a total of the other rows.
- Not including a row for missing values in the table.
- No formal hypothesis test for whether probability of fatality is related to number of female drivers involved in the accident.
- Incorrect interpretation of results (saying probability of fatality did not depend on the number of female drivers)
- Everyone reported odds of fatality when they are actually probabilities
- I did, however, penalize people who jumped to conclusions, claiming things like "females are safer drivers" without mentioning that it is but one of the many possible reasons for the trend in the data. Saying things like this is one of the more serious infractions a statistician might make. :(

  CJS: The hallmark of a good statistician is to ask "why could this have happened" and think carefully through the consequences of your "hypothesis" relative to the data. Is females were "safer" drivers, then they would

be involved in fewer collisions. However, our analysis conditioned on being in an accident and looked at the subsequent fatality rate so we don't know if 10% of female drivers were in accidents or 20% of female drivers were in accidents. So females could have fewer accidents, but once you are in an accident, the "safeness" of the driver is not relevant.

Part 03 Nests

Some general comments from the marker:

- Not estimating the difference in means for the controlled and experimental groups.
- Not including regression line or interpretation.
- Not including standard errors of estimates.
- Testing wrong hypothesis (number of mites depends on buttweight).
- Including the wrong plots.
- Poor coding style.
- Not using the *ods output lsmeansCL=* in *PROC glm* which gives the confidence limits for the estimates which are used in later parts of the estimates.
- Hardly *anybody* mentioned what the treatment was, or even which variable was being measured. Most people only said that they were comparing a "treatment" and a "control" group, without any elaboration. I can count on one hand the number of people who actually bothered to explain what the experiment was trying to measure.

  CJS- Again, you will be involved in LOTS of different projects so you must understand the projects throughly.
- I didn't get the impression that anyone understood the reason for the regression of the difference against the weight of butts. I got plenty of correct conclusions, but very little to tie those conclusions up to the first test. A lecture on controlled experiments might be in order...

  CJS - Some of this is covered in Stat-350 and Stat-430.
- I took marks off where people interpreted the results of regression as "the difference in butts increases by 0.0016 as the weight of butts increases". The SE and the anova should make it clear that there is no evidence of a relationship.

To add to the misery, I think a lot of people thought the paired test and the regression were both testing the same thing. Plenty of assignments were very confusing reads :(

CJS - Again, think CAREFULLY.