Stat-340/341/342 – Assignment 8
2015 Spring Term

Part 1 - Breakfast cereals - Intermediate

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

- take a continuous variable and create a categorical version of it;
- create dot plots, box plots, notched-box plots, and bar charts using *ggplot()*
- improve plots through jittering;
- use the *t.test()* function and compute the *se* of the difference in means;
- do a single factor ANOVA using the *lm()* function;
- extract estimated marginal means using the *lsmeans()* function from the *lsmeans* package and make a suitable plot.
- why *se* and ci's differ for marginal means when computed using summary data from each group separately or from a pooled analysis.

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 the coding for missing values as in previous assignments.

5. Make a histogram of the grams of fibre across the cereals using the appropriate geometry from *ggplot*. You may have to adjust the breakpoints as seen in the previous assignment. What does the histogram tell you about the grams of fibre/serving?

6. Find the five number summary using the *summary()* function. Do the results of the summary function match what you see on the histogram.

7. A common analysis method is to divide a continuous variable (such a *fiber*) into categories based on quartiles. Create a categorical variable, *fiberclass*, using the *recode()* function from the *car* package taking the values of *high* and *low* with splitting the values at 2. Why do I use alphanumeric values for the classification rather than numeric codes of 1 and 2?

   You should always practice defensive programming by checking that you've done the classification properly. For example, create side-by-side dot plots (as you did in previous assignments) of the continuous variable fibre by your categorical variable to show that you have done the division properly.

   Check your recode to see in which class the value of 2 is assigned.

8. Convert *fiberclass* variable to a factor, but order the levels appropriately. Hint: how does *R* order factor levels by default and how do you change this? Hint: check the help for the *factor()* function and refer to the previous assignments. You can check the ordering of the factor levels using the *levels()* function applied to a variable that is defined as a factor. You can also check to see that it is an ordered factor by using the *str()* function as you did on previous assignments.

9. We wish to examine if the mean calories/serving varies by fibre class. Begin by making a side-by-side dot-plot with an notched box plot overlaid (see previous assignments). The plot looks very odd! Why? What do you conclude from this plot?

   There appears to be some potential outliers in one of the groups – we will include these for now.

10. Next, compute the mean calories for each fibre class along with its standard error and a 95% confidence interval. You could dig out your textbooks and do the computations "by hand" (like you did in the previous assignment), but it is always better to let the machine to the arithmetic for you.

    Use the *ddply()* function with your own bespoke function. Your function should fit a simple intercept only model using the *lm()* function applied to the subset of the data from the *ddply* function. The formula in the call will look something like $y \sim 1$.

    Your code will look something like:

```
fiberclass.report <- ddply(cereals, "fiberclassF",
    function(x){
      # Compute some estimates of the mean response for each fiber class
      # by fitting an intercept only model
      fit <- lm(..... , data=x)
      ....
    })
fiberclass.report
```

Then extract the estimates, the *se*, and 95% confidence intervals using the appropriate method functions (e.g. *coef()*, *vcov()*, etc.).

You should get a summary table that look like:

```
  fiberclassF     mean  mean.se       lcl      ucl
1         low 105.2083 2.490934 100.19722 110.2194
2        high 104.8276 5.148549  94.28126 115.3739
```

11. Make a bar chart of the estimated mean from each fiber class along with the 95% confidence interval using the *ggplot2* package. You can add error bars to bar charts using the *geom_errorbar()* function. What do you conclude from this plot.

12. Now use the *t.test()* function to do the formal hypothesis test of equality of population mean calories between the two fiber classes. Read the help file and use the formula argument in much the same way as was shown in previous assignments.

    Print the created object. What are the null and alternate hypotheses being tested, the test-statistic, and the *p*-value. Interpret the *p*-value. What do you conclude?

13. Use the *str()* function on the object and you will see that it is simple list object with named components.

    Notice that the object created by *t.test()* contains a 95% confidence interval for the estimated difference in means, but contains neither an estimate of the actual difference in means, nor a *se* for the estimated difference in means. Sigh[1]. We need to compute these "by hand" and add it to the created object.

    Estimate the difference in means using something like

    ```
    fiberclass.ttest.object$estdiff <- sum( c(1,-1)*fiberclass.ttest.object$estimate)
    ```

    Do you understand how the difference was computed?

    If you do a *str()* on the object, you will notice that you have successfully add a new item to the object, but when you print the object, it doesn't

[1] *R* is free, but not cheap.

show. Sigh! The reason is that the fitted object has a special class called *test* (try *class(fiberclass.ttest.object))*, and the print method for this class ignores everything except that created by the *t.test()* function. You could modify this print method, but that is a lot of work.[2]

How do we estimate the *se* of the difference in means? Recall that the t-statistic is defined as the $t = \frac{estimated\ difference}{se\ difference}$. So if we know the t-statistic (given already in the object), and the estimated difference in means (computed above), you can use these two value to compute the *se* of the difference in means. Do this and add it to the object.

Hint: Look at some of the sample programs at `http://people.stat.sfu.ca/~cschwarz/Stat-650/Notes/MyPrograms/` that do a t-test to see how this is done.

Print the object and estimated difference in means (with se) using something along the lines of

```
print(fiberclass.ttest.object)
print(fiberclass.ttest.object$estdiff)
print(fiberclass.ttest.object$estdiff.se
```

In this day and age, not providing an estimate of the difference and the associated standard error automatically is simply old-fashioned and out of touch with modern statistical practice. Shame on *R*!

14. Now use the *lm()* function to again compare the mean number of calories/serving between the two fiber classes except that now include the *fiberclass* factor in the model.

    Use the *anova()* method to get the formal test. What are the null and alternate hypotheses, the test-statistic, and the p-value. What do you conclude.

    Note that the p-value from the *t.test()* function and from the *lm()* function are not quite exactly the same (in this case they differ in the 4^{th} decimal digit). In general, they could differ more. Why are the two results NOT EXACTLY the same? Hint: what assumptions about the standard deviations of the number of calories/serving in each of the fiber groups for the two methods?

15. Use the *summary()* function to see the actual coefficients from the model. Do the coefficients actually have any easy interpretation? IF YOU KNOW WHAT YOU ARE DOING, you could use these estimates directly. But again, to protect yourself, you should always use a method that works regardless of the underlying parameterization used by the statistical package.

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

16. You should NEVER use the results from the *summary()* function applied to a model object to estimate effect sizes in ANOVA type models. Rather use the *lsmeans* package introduced in a previous assignment. Load the package, and compute the individual marginal means.

    Compare the estimates, se and confidence limits for the marginal means to those you computed (and plotted previously). Can you explain why the estimates are the same, but the estimated *se* and ci are different? Hint: what assumptions about the standard deviations of the number of calories/serving in each of the fiber groups for the two methods?

17. Use the estimated marginal means from the *lsmeans* package to create a bar chart of the means and add the confidence intervals. Compare the two bar charts. Again, why are the plots not identical. See hint from previous part.

18. Use the *pairs()* method from the *lsmeans* package (recall that a *method* is just another name for a specialized function that is used to extract information from model fits) to estimate the difference in the marginal means. Compare the results to the output from the *t.test()* function and the *summary()* function applied to the *lm()* fit. Again, the results do NOT MATCH EXACTLY. Why? Hint: refer to the previous hints.

    The sign of the difference may also vary - why? How can you tell the order of the difference?

19. Finally, we want to produce diagnostic plots of the fit of the *lm* model. Base *R* has a plot method to produce such plots from a *lm()* object, but this is not currently available in the *ggplot2* package. However,visit `http://rpubs.com/sinhrks/plot_lm` for instructions on how to install the *ggfortify* package to add to functionality of the *ggplot2* package.

    Alternatively use:

    ```
    source("http://www.stat.sfu.ca/~cschwarz/Stat-650/Notes/MyPrograms/schwarz.functions.r"
    sf.autoplot.lm(fiberclass.lm)
    ```

    to download some of my own functions with similar functionality.

    Is there any evidence of problems? For example, what are some points labels in the residual plot? Why does the normal qq plot appear "stepped"?

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.
  - A statement about the comparison of interest. An estimate of the difference in mean, its standard error, and the p-value from the relevant hypothesis. What do you conclude?
  - The two plots (created using *ggplot*) comparing the means and an explanation of why the plots are not identical.

  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

In this assignment, we will examine the circumstances of personal injury road accidents in Great Britain in 2010. The statistics relate only to personal injury accidents on public roads that are reported to the police, and subsequently recorded, Information on damage-only accidents, with no human casualties or accidents on private roads or car parks are not included in this data.

Very few, if any, fatal accidents do not become known to the police although it is known that a considerable proportion of non-fatal injury accidents are not reported to the police.

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

- import date-time values;
- compute the odds ratio for fatalities for individual hours using the *plyr* package
- compute the odds ratio for fatalities for individual hours using a modeling approach
- use the *lsmeans* package for *glm()* objects.

1. Return to the *Accidents* data file from earlier in the course.
2. Read the data into *R* using the *read.csv()* function with appropriate options. Print out the first few records and look at the structure of the data frame.

   There are over 150,000 records, so manually scanning the entire data is simply not feasible, nor desirable.
3. Notice that the *date* and *time* have been read in as character data. We need to convert them to the internal *R* representation of a date-time, the number of second since midnight 1970-01-01. Convert the date and time character data to an internal *R* POSITct objects using the *as.POSIXct()* function. Note that you will have to *paste()* the date and time together into an appropriate string before conversion.

   Use the *summary()* function to find the minimum, maximum, and other statistics about the date-time of the accident. Are there any values outside of 2010?

   Are there any missing values? Look at the records with missing values to see why the date-time value cannot be computed and is missing.

4. Extract the hour of the accident using the *format()* function as you did in a previous assignment. Note that the *format()* function returns a character representation and you likely want to convert this to a number for ease of use using the *as.numeric()* function.

5. Use the *recode()* from the *car* package to create a variable taking the values *yes* and *no* to indicate if the accident involved a fatality. Again, why should you use alphanumeric code rather than numeric codes? Then convert this to a factor because the *glm()* function cannot deal with alphanumeric responses and expects a factor where the FIRST LEVEL of the factor is a FAILURE. In this case, it will be a non-fatality. You likely want an ordered factor – recall how you specified an ordered factor in previous assignments.

   Practice defensive programming. Use *xtabs()* function to look at the cross classification of the severity index and your classification as fatal or not fatal. Also, use the *str()* function to verify that you've got the levels done properly.

6. Create a simple table using *xtabs* of the fatality status by the hour of the accident.

7. Get an estimate of the log-odds along with its standard error and confidence intervals using the *ddply()* function as you did in a previous assignment and question. Hint: Once again fit an intercept-only model to each hour's data within the *ddply()* function and then use the methods to extract the various estimates. These estimates will be on the log-odds scale.

   Note that you may have to delete the rows from the accident data frame that have a missing value for the *hour* of the accident./p-valu

   Convert these estimates and ci from the log-odds scale to the odds scale and then to the probability scale just like you did in previous assignments.

   This can all be done in the *ddply()* call.

8. Plot the odds of fatality and the 95% confidence interval by the hour of the data using the *ggplot()* function. What do you conclude based on the plot?

   Hint: be very careful about how the hour of the day is treated. It starts at 0 which is the FIRST level of the factor and so must appear as such on the plot.

9. As you saw in the first part of this assignment, modeling can sometimes be used to estimate the individual estimates with the added advantage that hypothesis testing occurs at "no additional cost", but sometimes the model makes further assumptions (e.g. equal standard deviations over all treatment groups) that can give you slightly different results than the individual computations.

Use the *glm()* function to model the fatality rate across the hours of the day.

10. Use the *summary()* method to examine the results of the model. You did convert Hour to a factor didn't you? What happens if the hour is NOT declared as a factor – is this what you want?

    Why are there only 23 coefficients (other than intercept) printed when there are 24 hours in a day? What is the interpretation of these coefficients?

    Again, NEVER use the estimates reported by the *summary()* function applied to a model object that contains factor variables because the coefficients reported depend on the internal coding of the model and could change.

11. Use the *anova()* function to examine if there is evidence of a difference in the log-odds of fatality among the various hours. The use of the term *anova()* is a bit-misleading because you are NOT comparing means here – with logistic regression you are comparing the probability of membership in a category by comparing the log-odds of membership.

    You will first notice that no p-value s are reported by the *anova()* function. Read the help on *anova.glm()* function which is the particular method called by *anova()* function when applied to a *glm* object, to see that you can request different types of computations for the p-value. In more advanced classes, you will learn about these different methods. For now, select the likelihood ratio method of computing the p-value.

    What hypothesis is being tested? What are your conclusions?

12. We now want to get the log-odds of a fatality for each hour. We will use the *lsmeans* package even though this model is not an analysis on the mean response, but on a log-odds response.

    Apply the *lsmeans()* function to the fitted object and verify that the log-odds are reported. Just as in *SAS*, the term *lsmeans* is used generically to represent the marginal estimates.

    Use the *lsmeans()* function to get the marginal estimates (of the log-odds) and associated standard errors and large sample confidence intervals as you did previously.

13. Which hour has the highest estimated odds of a fatality? If you look at the plot of the odds of a fatality along with their associated 95% confidence intervals, it is difficult to say exactly when this occurred (why?).

    Use the *cld()* method on the *lsmeans* object created above. What do you conclude?

14. If you compare the two tables of results, you will see that the estimates are all the same, but there are slight difference in the confidence intervals

because of the profiling vs. large sample methods for computing confidence limits.

In this case, there is NO equivalent of the standard deviation parameter that you saw in the previous assignment, so concerns about using a pooled or separate estimate of the standard deviation over treatment groups are not applicable. So, if there is no standard deviation in the model, why are there slight differences in the results from the two approaches?

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 plot of the odds of fatality by hour of the data and your conclusions from this. At what hour is the odds of a fatality highest and how certain are you of that result?

Part 3 - Practice for exams

Here are some small practice questions that could appear on the term test or exam. Nothing to hand in, but compare your answers to the answer key.

Remember you will have a copy of *R* reference card (`http://cran.r-project.org/doc/contrib/Short-refcard.pdf`) and the ggplot reference card (`http://people.stat.sfu.ca/~cschwarz/Stat-R/CourseNotes/R-manuals/R-ggplot2_ref_card.pdf`) available on term tests or final exams.

1. **Find the *se* and large sample confidence interval for a binomial proportion in a SRS using formula** Write a function that takes the sample size and number of success and returns the estimated proportion of successes, the standard error of the sample proportion, and the large sample confidence interval for the population proportion. The confidence level should default to the 95% level, but should be able to be changed. I'll start you off

   ```
   pstat <- function( sampsize, nsuccess, confleve=0.95){
   ...
   }
   ```

   Recall that the standard error of a sample proportion is found as

   $$se_{\widehat{p}} = \sqrt{\frac{\widehat{p}(1-\widehat{p})}{n}}$$

   where $\widehat{p}$ is the sample proportion and n is the sample size.

   The large sample confidence interval is

   $$\widehat{p} \pm z_{\alpha/2} se_{\widehat{p}}$$

   where $z_{\alpha/2}$ is the appropriate quantile from the normal distribution. The *qnorm()* function returns the appropriate quantile from a standard normal distribution.

   Here are some test cases for you to try:

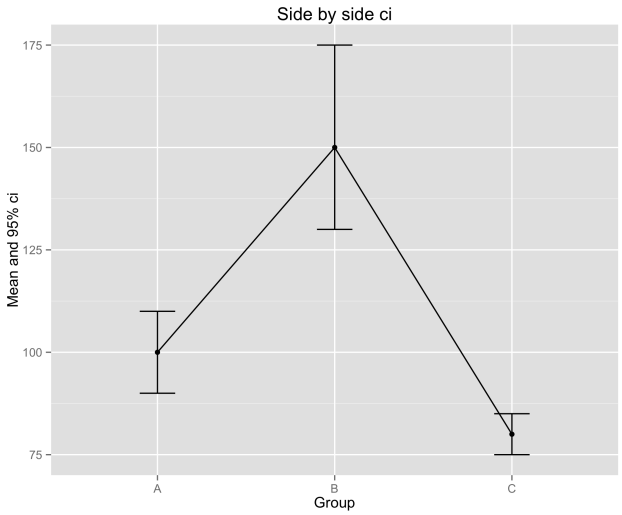
   ```
   > pstat( 1000, 300)
         phat    phat.se        lcl        ucl  conflevel
   0.30000000 0.01449138 0.27159742 0.32840258 0.95000000
   > pstat( 1000, 300, conflevel=0.90)
         phat    phat.se        lcl        ucl  conflevel
   0.30000000 0.01449138 0.27616381 0.32383619 0.90000000
   ```

2. **Draw a graph using ggplot** Write a code that return a plot where the means are connected between groups and each group's confidence limit are displayed. You may assume that you have a dataframe with variables *group*, *mean*, *lcl*, and *icl* in the data frame.

   Here is some test data:

```
> mydf
  group mean lcl ucl
1     A  100  90 110
2     B  150 130 175
3     C   80  75  85
```

   You should get



3. **Read in and extract information from dates** Given a dataframe with a vector of character strings representing dates, convert these to a date variable, and extract the year, weekday (i.e. 0 to 6 representing Sunday through Saturday) and the julian day (the number of days since the first of the year). This doesn't have to be a function – just straight code is all that is needed. Add the new variables to the data frame containing the date in character form.

   Here is some test data

```
> mydf
       InDate
1 14/mar/2014
2 15/mar/2014
3 16/mar/2014
4 14/mar/2013
```

```
5  3/mar/2013

> str(mydf)
'data.frame': 5 obs. of  1 variable:
 $ InDate: chr  "14/mar/2014" "15/mar/2014" "16/mar/2014" "14/mar/2013" ...
```

and some results from the above

```
> mydf
       InDate       Date Year WD  JD
1 14/mar/2014 2014-03-14 2014  5 073
2 15/mar/2014 2014-03-15 2014  6 074
3 16/mar/2014 2014-03-16 2014  0 075
4 14/mar/2013 2013-03-14 2013  4 073
5  3/mar/2013 2013-03-03 2013  0 062
>
```

Comments from the marker

General comments from the marker:

- There were no for-loops in the *R* code which was great to see.
- *R* packages were loaded throughout scripts. It's common *R* style (from my own experience) to load all required packages at the beginning of your script, instead of interspersed throughout your script when you need a certain function. The main reason for this is that there are lots of packages in *R* and if you run your scripts on different machines, it's often the case that the new machine will not already have all your favourite packages installed. If you load them at the beginning of your script, it's really easy to see which packages your script requires so that you can install them if necessary. This is also helpful for other people if they will be running your script on their machine.

  On that note, here's a tip for loading packages in *R*. Say you need to use *plyr*, *ggplot2*, *reshape2* and *lsmeans* in your script and you are running your script on a school computer, so you don't know if they're already installed. You can use the following lines at the beginning of your script to check if these packages are installed, and if not, install them.

```
# Make a vector that contains the names of the packages that you'll use
required.packages <- c("plyr","ggplot2","reshape2","lsmeans")

# Define a function that checks if the package package.name is installed,
# and if not, installs it
checkAndInstall <- function(package.name){
  if(!package.name%in%rownames(installed.packages())){
    install.packages(package.name,
                     dep=TRUE,
                     repos="http://cran.r-project.org")
  }
}

# Apply the checkAndInstall function to each element of your vector required.packages
sapply(required.packages,checkAndInstall)

# Load your packages
library("plyr")
library("ggplot2")
library("reshape2")
library("lsmeans")
```

Part 1 - Cereal data

Some general comments from the marker:

- Most of the marks were deducted for not having a proper caption/legend for the plot.
- There were also some issues with estimating the mean for the two groups - some submissions showed a larger difference in the mean than there should've been in the plots. This is likely a result of cereals with EXACTLY 2 grams of fiber being assigned to the wrong group. You should ALWAYS carefully check the results of any recoding that you do to ensure that it is done correctly.
- Most students understood the difference between the two standard error calculations which was great.

I'd like to stress the importance of a good introduction. Yes, you need a brief description of the data, but you're not obligated to say the same thing in every writeup. In fact, it's better when you don't : instead try to mention the parts of the data that are actually important in the context. For instance, in this case I don't need to know that "missing values were recorded as -1", or that "shelf the cereal was displayed on was recorded".

Secondly (and this is important!) – you need a sentence that states why you bothered to do what you did. What question were you trying to answer? If you can, it helps if you write that in the form of an actual question.

To give you an idea, here is one example of an excellent introduction from a student:

> "The cereal data set contains information on 77 brands of cereal. For each cereal brand, nutritional information such as calories, protein (g) and fibre (g) was given, as well as manufacturer name and cereal type. Cereals were classified as having high fibre content if their fibre content was greater than or equal to the median of the sample (2 g). All other cereals were classified as low fibre. Does the mean calories per serving differ between the two fibre classifications?"

Pretty much everybody by now can compare two means and report the t-statistic and the p-value and make correct inferences about them, and that's

really great, I'm very pleased with that. I don't think I've dealt with any write-ups where that wasn't done correctly. A couple of times I had to subtract point for an incorrect conclusion, but I think at this point most of those blunders are due to the awkward language of hypothesis testing that students keep subjecting themselves to.

I'm pretty sure that Carl has already mentioned this in class, but I'll mention it again. Hypothesis testing language is confusing, and should NOT be used in write-ups! There's never any reason to use it outside of your intro-to-stats midterm. In fact, for most of the people who require the services of statisticians the word "hypothesis" has a completely different meaning and connotations. It's far, FAR better to say "there was no evidence that the two means were different ($p = 0.98$)" than to say "the p-value is 0.98, which is bigger than 0.05, we thus fail to reject the null hypothesis which is that the two means are the same." The former is in plain English and would make sense even to people who don't know/care what a t-test is. The latter is gobbledy-gook, EVEN if you manage to say it correctly.

I assigned two points to being able to give me the reason for the two plots being different. Any mention of standard deviation got people 1 point, but unless I heard that one of the methods used pooled variances and another didn't, I didn't give out full points. The most common mistake was stating that the difference in the degrees of freedom amongst the two methods made a difference in the CI's. That's not really the case – rather, the different degrees of freedom are just another symptom of the different assumptions inherent in the two methods.

There were a few times where I've subtracted points for passages that made no sense to me. I'd like to mention that whenever you communicate about statistics, strive for simplicity above all. Avoid using technical terms that you haven't defined previously; avoid sounding like a textbook; avoid QUOTING the textbook (or Wikipedia), and most importantly: make sure that you understand 120% of the technical lingo you're introducing. In this particular case, the assignment did not require you to use any fancy statistical theory, nor long complicated explanations.

Here is an example of some excellent student work. Note how it contains no fancy language:

> The following are two plots for comparing the means. The plots are not identical because they were made from analyses that make different assumptions of the standard deviation. The left plot used

information from the unequal variance t-testl, which does not assume that the standard deviations are the same between two fiber classes; white the right plot was generalized by an ANOVA model, which assumes equal standard deviation between two fiber classes."

Part 2 Accidents data

- Some students justified their claim that the odds of fatality are the highest at 2 a.m. by citing a hypothesis test. I think the hypothesis test they speak of would be testing whether odds of fatality are different depending on the hour of the day. Based on the plot, we can see that there is evidence of odds differing for different hours. However, this test tells us nothing about specific hours of the day. This is why you need to do a multiple-comparisons analysis (the *cld()* method) to compare that hour to the other hours of the day
- A few students said that the analysis was for the probability of a fatality, but it's actually the odds of a fatality.
- Most points were deducted for not giving a caption/legend for the plot.

I expected to see three elements in this part of the assignment:

1. A plot of fatality odds vs. hours, appropriately referenced and labeled, with CIs. This was done well for the most part. A couple of people plotted the non-fatality odds, and the log odds – that wasn't penalized as long as it was correctly labeled and referenced. A number of people had trouble plotting CIs and means and having them not be offset by an hour. Those lost a mark.
2. A mention of important trends in the plot: significantly higher odds of a fatality during early morning than in the afternoon, bigger CIs at night than during the day. This was skipped by some people, and that's not good. Any time you include a plot in your write-up, you have to point out the main features of that plot, and by far the most noticeable feature was the sharp drop at around 7 or 8 in the morning in the odds of fatality. Only a few people mentioned the wide CIs during the night and narrow CIs during the day. Only one solitary person conjectured that it may have been because most accidents actually occur during the day – good job there!
3. A mention of what the hour associated with the greatest odds of fatality was, and whether we were sure that was the actual "peak". I didn't need anything fancy here – just a mention that CIs overlapped a lot and that

we weren't sure that 2 a.m. was the actual peak. I nonetheless got a lot of answers which did not make sense to me. Once again, simple is good. The rule of thumb of scientific communication – If your mom can't understand your write-up, it's too complicated.

More comments

I agree with everything that the marker said about stating the results of hypothesis testing, and I even go further.

NEVER USE THE FOLLOWING PHRASES in reports

- accept the null hypothesis
- reject the null hypothesis
- fail to reject the null hypothesis
- accept the alternate hypothesis
- results are statistically significant

Using these phrases brands you as a statistical robot (a.k.a. a trained monkey) who blindly follows a recipe without understanding the purpose of the ingredients. When I hire a student to work on a project with me, I always ask then to explain the result of a two-sample t-test. If they use any of the phrases above, they fall to the bottom of my list of people to hire.

Hypotheses are merely convenient mathematical fictions – indeed they are ALWAYS false!! For example, if you compare the calories/serving across the different shelves, do you really expect that the mean amounts of calories/serving in the population are equal to a million decimal places!! So you see that that hypotheses MUST necessarily be false and is only a convenient mathematical fiction.

Statistics is all about collecting evidence. So you should say phrases such as:

- ... there was no evidence that the mean calories/serving varies among the shelves (p=.35).

- ... we were unable to distinguish among the mean calories/serving across the shelves (p=.35).
- ... there was strong evidence that the mean calories/serving varied among the shelves (p=.002).

By saying that there was no evidence, we are saying that given the data (and the overlap in the confidence intervals), there was enough overlap in the confidence intervals that we were unable to distinguish the means.

Statistical "significance" is NOT the same as biological importance, i.e. just because you detect a difference in the means, does not imply that the difference is of any practical value.

Refer to Sections 2.4.5, 2.4.6 (especially the graph in this section), and 2.4.7 (esp. the Cherry and Johnson paper) in my course notes for more details.