36-402/608 Homework #1 due 10:30AM 1/21

1. t-test (20 points)

   Use fullBumpus.R to set up the data from fullBumpus.txt (both at Blackboard/Assignments). For this problem, analyze the full dataset together – don't break down by the Group variable.

   (a) Perform two t-tests to see if the weight of the bird differs by survival status, trying both var.equal=TRUE and var.equal=FALSE. (The latter "adjusts" for unequal variance.) Turn in your two R statements and the corresponding output.

   (b) With reasonable sample sizes, the t-test is quite robust to (unaffected by) moderate amounts of non-normality. Nevertheless, it is a good idea to check for normality of errors by examining the residuals with a quantile-normal plot. To get residuals for this problem, the easiest method is to re-run the analysis as a simple regression using `res = resid(lm(Weight~Survive, sparrow))`. A nice version of quantile-normal plots with confidence bands is from Brian Junker. To load it, enter `source("http://www.stat.cmu.edu/~hseltman/files/qqn.R")`. Then create the plot using `qqn(res)`, but don't turn it in. State whether or not you think that the plot shows evidence of sufficient deviation from the reference line to suggest a troublesome degree of non-normality.

   (c) The t-test is only moderately robust to unequal variance. Unlike the statistical significance of the mean difference, equal vs. unequal variance is easily judged on a side-by-side boxplot. Make a side-by-side boxplot comparing the weight distribution of surviving and perished sparrows. As a rough rule of thumb if the ratio of the IQRs is between 0.5 and 2.0, there is no cause for concern about unequal variances. Roughly what ratio (Survive to Perish, say) do you see?

   (d) The t-test is non-robust to correlated errors. We'll examine this by simulation next week. Correlation is either serial (adjacent subjects are correlated) or by some other grouping, e.g., by nest in this example. The intuition is that, if birds in the same nest are highly correlated in their weights, then there is really not much more information gained by sampling several vs. one bird per nest, but the t-test "thinks" that you have a much larger "n" and therefore inappropriately reduces the estimate of the standard error, resulting in falsely low p-values and falsely narrow confidence intervals. To get a feel for this, load HW1FakeCor.txt and make side-by-side boxplots of weight by nest for both WeightA and WeightB (considered as alternate realities). Which one corresponds to correlated (within-nest) errors?

2. Regression (20 points)

   Now we will pretend that the goal of the bird analysis was to model wing length ("Alar") using gender and Weight (without interaction) as explanatory variables.

   (a) Turn in the R command to store the lm() result in a variable called "mdl". Turn in the result of `summary(mdl)`.

   (b) Turn in assignment statements of the form `b0M=`, `b0F=`, and `b1=` which obtain the estimates of the intercepts and slope from "mdl" using the `coef()` function. To do this, you need to think about the structural model for the regression, and how it simplifies when "Female" is no longer considered to be a variable, but rather is held constant at 0 (male) or 1 (female). (Do not try to do this by fitting two separate regressions!)

   (c) Make and turn in a single plot summarizing the data and model as follows:

```
with(sparrow, table(Female, as.numeric(Female)))
with(sparrow, plot(Alar~Weight, pch=as.numeric(Female),
                   col=as.numeric(Female), main="Bumpus"))
abline(b0M, b1, col=1, lty=1)
abline(b0F, b1, col=2, lty=2)
legend(28, 240, c("Male", "Female"), col=1:2, lty=1:2, pch=1:2)
```

   If this does not look right, go back to step b.

   You don't need to turn anything else in, but it is worthwhile examining the residual vs. X plot for this model using:

```
plot(resid(mdl)~sparrow$Weight, col=as.numeric(sparrow$Female),
          pch=as.numeric(sparrow$Female))
abline(h=0)
```

   (d) Now repeat the whole process with "mdlI" being the interaction model. You'll need to redefine b0M and b0F, and now introduce b1M and b1F for the separate slopes. Turn in the single plot summarizing the data and the interaction model, with a legend.

   (e) Run `anova(mdl,mdlI)` and make a claim whether or not we have good evidence of non-parallel slopes.

   (f) Run `confint(mdlI)` and turn in the 95% CI for the difference of slopes (female - male).

3. Array indexing in R (20 points)

   Assign `mymat = matrix(c(rep(c(3,5,7,10),4), seq(1,32,2)), nrow=16)`. Turn in expressions to get the output listed below. Your statement must work in general, not just for this specific matrix.

(a) column one

(b) the matrix of rows 2, 8, and 5

(c) a logical T/F vector telling whether each row has an even number in column one (check `?`%%``)

(d) a vector showing the values of column two that corresponds to an even number in column one

(e) the even numbered rows

(f) the rows where column two is exactly three times column one

4. Reading in data in R (20 points) Use various parameters of `read.table()` to read in each of the files listed below as a variable named "tmp". If you did it correctly, the associated command will give the corresponding results. Turn in just the `read.table()` statements. (

   (a) easy.txt: `mean(tmp$Age, na.rm=TRUE) # [1] 99`

   (b) easy.csv: `mean(tmp$Age, na.rm=TRUE) # [1] 99`

   (c) harder.csv: `mean(tmp$Age, na.rm=TRUE) # [1] 99`

   (d) hardest.csv: `nchar(as.character(tmp$Comment)) # [1] 7 10 11 13 9 13`

   (e) tabbed.txt: `mean(tmp$Age, na.rm=TRUE) # [1] 99`

5. Function writing in R (20 points)

   Starting with the given function, add each additional bit of functionality, then test the function, before adding the next bit. Turn in the final, complete function. Remember to update the comments in the code.

   As a start set:
   `x = 1:8; y = 2 + 3*x + rnorm(8, 0, 1.5)`. Then use mylm.R to define:

```
# Function to do some standard EDA, analysis, and residual
# checks for simple regression.
# Input: x and y are explanatory and response vectors.
# Output: returns a list with useful information.
# Side effects: make EDA and residual plots if plots==TRUE.
#
mylm = function(x, y, plots=TRUE) {
  # Some input checking
  if (!is.numeric(x) || !is.null(dim(x)))
    stop("x must be a numeric vector")
  if (!is.numeric(y) || !is.null(dim(y)))
    stop("y must be a numeric vector")
  if (length(x) != length(y))
```

```
    stop("x and y must be the same length")

  # The main analysis
  mdl = lm(y ~ x)

  # The optional plotting
  if (plots) {
    par(mfrow=c(2,1))
    plot(y~x)
    abline(mdl) # add the fitted line
    plot(resid(mdl)~fitted(mdl), xlab="Fitted values", ylab="Residual values")
    abline(h=0)
  }

  # Returning the coefficients and the number of usable data points
  return(list(coef=coef(mdl), nOK=sum(!is.na(x) & !is.na(y))))
}
```

Then test the function with:

```
mylm(x, y)
dev.off()
mylm(y, x, plot=FALSE)
mylm(x, c(y,NA))
mylm(c(x,NA), c(y,NA))
mylm(x*c(1,1,NA,1,1,NA,1,1), y)
mylm(list(x), y)
```

(a) Examine `coef(summary(lm(y~x)))`, then add code to also include the p.values in the output.

(b) Examine `confint(lm(y~x))`, then add code to also include the 95% CI for the slope.

(c) Check the help information for the confint function, then add code to allow the user of `mylm()` to input a "level" which defaults to 0.95 but can be changed to get a different sized confidence interval. Include a check that the value entered is between 0 and 1 (exclusive) and include the "level" in the output list.