36-402/608 HW #1 Solutions 1/21/2010

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.

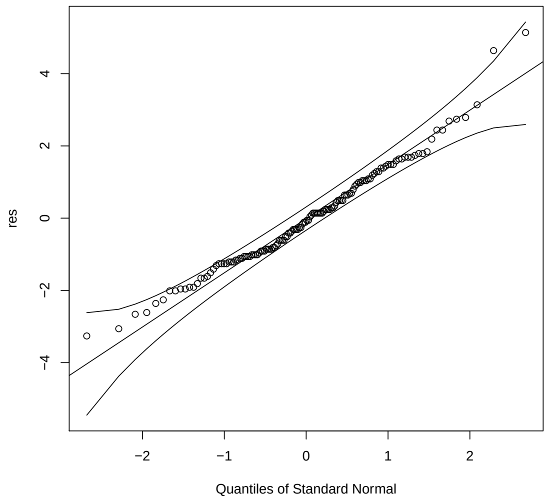
```
> with(sparrow, t.test(Weight[Survive=="Survived"],
                       Weight[Survive=="Perished"],
                       var.equal=TRUE))
        Two Sample t-test
 data:  Weight[Survive == "Survived"] and Weight[Survive == "Perished"]
 t = -2.6093, df = 134, p-value = 0.01010
 alternative hypothesis: true difference in means is not equal to 0
 95 percent confidence interval:
  -1.1399459 -0.1569291
 sample estimates:
 mean of x mean of y
  25.21250  25.86094

> with(sparrow, t.test(Weight[Survive=="Survived"],
                       Weight[Survive=="Perished"],
                       var.equal=FALSE))
        Welch Two Sample t-test
  data:  Weight[Survive == "Survived"] and Weight[Survive == "Perished"]
 t = -2.5703, df = 117.954, p-value = 0.01141
 alternative hypothesis: true difference in means is not equal to 0
 95 percent confidence interval:
  -1.1480287 -0.1488463
```

   (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.

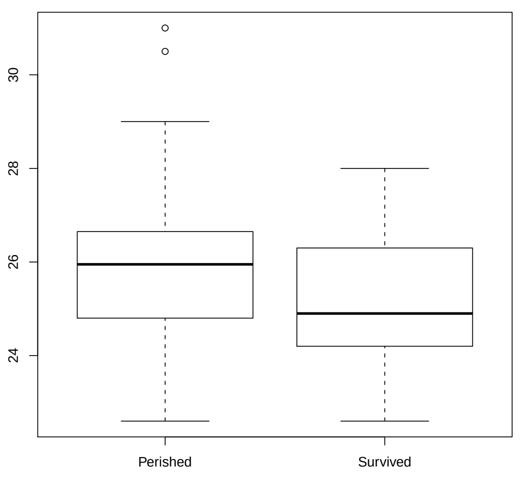
```
res = resid(lm(Weight~Survive, sparrow))
source("http://www.stat.cmu.edu/~hseltman/files/qqn.R")
qqn(res)
```



Only one point is outside the confidence band, so I would *not* worry about 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?

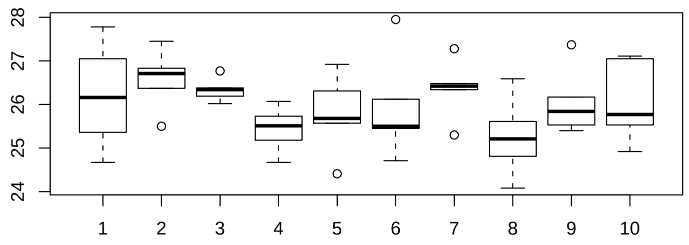
```
with(sparrow, boxplot(Weight~Survive))
```

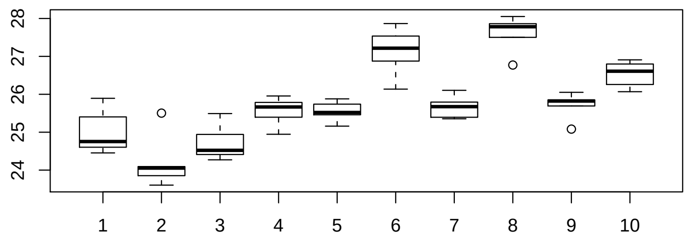


The ratio is only slightly greater than 1, so I am not worried about breaking the equal variance assumption.

(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?

```
fake = read.table("HW1FakeCor.txt")
par(mfrow=c(2,1))
with(fake, boxplot(WeightA~Nest))
with(fake, boxplot(WeightB~Nest))
par(mfrow=c(1,1))
```





Although the median to median variation is about the same in both plots, for the plot of WeightB, the variability within a nest is reduced, suggesting correlated 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)`.

```
> mdl = lm(Alar~Female+Weight, data=sparrow)
> summary(mdl)

Call:
lm(formula = Alar ~ Female + Weight, data = sparrow)

Residuals:
     Min       1Q   Median       3Q      Max
-13.2387  -2.6125   0.2613   2.8729  11.0747

Coefficients:
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  202.1958     6.1318  32.975  < 2e-16 ***
FemaleFemale  -4.8027     0.7271  -6.605 8.71e-10 ***
Weight         1.7553     0.2372   7.401 1.37e-11 ***
---

Residual standard error: 3.942 on 133 degrees of freedom
Multiple R-squared: 0.4961,     Adjusted R-squared: 0.4885
F-statistic: 65.47 on 2 and 133 DF,  p-value: < 2.2e-16
```

(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!)

The means model is

$$E(Y) = \beta_0 + \beta_F \text{Female} + \beta_W \text{Weight}$$

and using the internal coding that Female=0 is a male we get specific means models of

$$\text{Males: } E(Y) = \beta_0 + \beta_W \text{Weight}$$

and

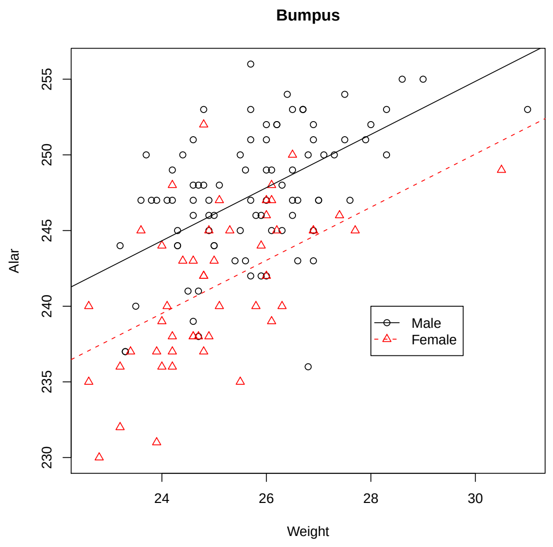
$$\text{Females: } E(Y) = (\beta_0 + \beta_F) + \beta_W \text{Weight}.$$

Matching the standard meanings of intercept and slope to the equations and identifying the corresponding positions of the estimates in `coef(mdl)` results in these assignments:

```
b0M = coef(mdl)[1]
b0F = coef(mdl)[1] + coef(mdl)[2]
b1 = coef(mdl)[3]
```

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

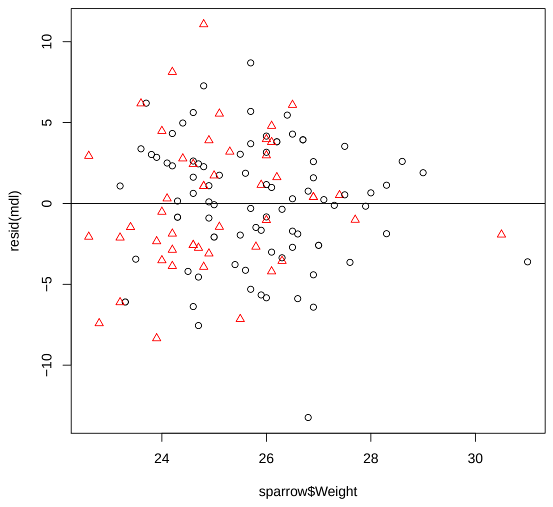
```
with(sparrow, table(Female, as.numeric(Female)))
with(sparrow, plot(Alar~Weight, pch=as.numeric(Female),
                   col=as.numeric(Female), main="Bumpus"))
abline(b0M, b1, col=1, lty=1)
abline(b0F, b1, col=2, lty=2)
legend(28, 240, c("Male", "Female"), col=1:2, lty=1:2, pch=1:2)
```



Note that without an interaction between Female and Weight, the fitted lines are forced to be parallel.

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.

The simplified means models are:

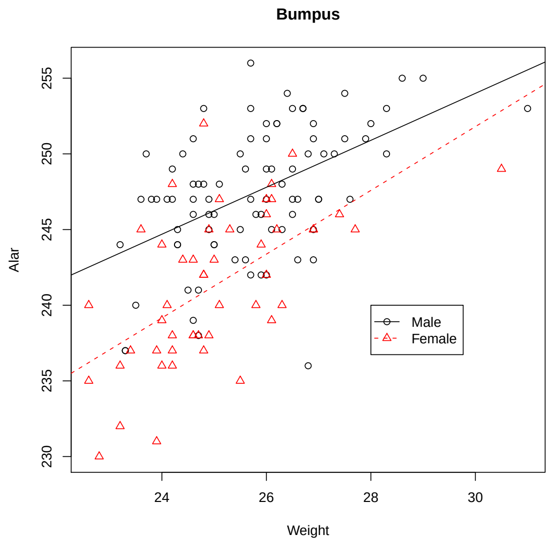
$$\text{Males: } E(Y) = \beta_0 + \beta_W \text{Weight}$$

and

$$\text{Females: } E(Y) = (\beta_0 + \beta_F) + (\beta_W + \beta_{F*W})\text{Weight}.$$

```
mdlI = lm(Alar~Female*Weight, data=sparrow)
summary(mdlI)
b0M = coef(mdlI)[1]
b0F = coef(mdlI)[1] + coef(mdlI)[2]
b1M = coef(mdlI)[3]
b1F = coef(mdlI)[3] + coef(mdlI)[4]

with(sparrow, plot(Alar~Weight, pch=as.numeric(Female),
                   col=as.numeric(Female), main="Bumpus"))
abline(b0M, b1M, col=1, lty=1)
abline(b0F, b1F, col=2, lty=2)
legend(28, 240, c("Male", "Female"), col=1:2, lty=1:2, pch=1:2)
```



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

```
> anova(mdl,mdlI)
Analysis of Variance Table

Model 1: Alar ~ Female + Weight
Model 2: Alar ~ Female * Weight
```

```
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1    133 2067.1
2    132 2047.3  1    19.810 1.2773 0.2605
```

This is the "added sum of squares F-test" for the null hypothesis that the extra interaction term does not add to the model fit any more than expected by chance. We conclude that we do **not** have good evidence of non-parallel fit lines.

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

```
> confint(mdlI)
                            2.5 %     97.5 %
(Intercept)           192.2381377 222.683434
FemaleFemale          -43.5018377   5.784645
Weight                  0.9619145   2.140502
FemaleFemale:Weight    -0.4166585   1.527374
```

The interaction term represents the slope difference (compare the male vs. female means models to see this). So we want [-0.417, 1.527].

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.

```
> mymat
      [,1] [,2]
 [1,]    3    1
 [2,]    5    3
 [3,]    7    5
 [4,]   10    7
 [5,]    3    9
 [6,]    5   11
 [7,]    7   13
 [8,]   10   15
 [9,]    3   17
[10,]    5   19
[11,]    7   21
[12,]   10   23
[13,]    3   25
[14,]    5   27
[15,]    7   29
[16,]   10   31
```

(a) column one

```
> mymat[,1]
 [1]  3  5  7 10  3  5  7 10  3  5  7 10  3  5  7 10
```

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

```
> mymat[c(2,8,5),]
     [,1] [,2]
[1,]    5    3
[2,]   10   15
[3,]    3    9
```

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

```
> mymat[,1]%%2==0
 [1] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE
[13] FALSE FALSE FALSE  TRUE
```

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

```
> mymat[mymat[,1]%%2==0, 2]
[1]  7 15 23 31
```

(e) the even numbered rows

```
> mymat[rep(c(FALSE,TRUE),nrow(mymat)/2), ]
     [,1] [,2]
[1,]    5    3
[2,]   10    7
[3,]    5   11
[4,]   10   15
[5,]    5   19
[6,]   10   23
[7,]    5   27
[8,]   10   31
## Or (but this won't work with less than 2 rows):
> mymat[seq(2, nrow(mymat), 2),]
## Or:
> mymat[seq(along=mymat[,1])%%2==0,]
```

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

```
> mymat[mymat[,2]==3*mymat[,1],]
     [,1] [,2]
[1,]    3    9
[2,]    7   21
```

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`

   ```
   tmp = read.table("easy.txt", header=TRUE)
   ```

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

   ```
   tmp = read.table("easy.csv", header=TRUE, sep=",")
   ```

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

   ```
   tmp = read.table("harder.csv", header=TRUE, sep=",", na.strings="NA")
   ```

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

   ```
   tmp = read.table("hardest.csv", header=TRUE, sep=",",
                    quote="\"", comment.char="")
   ```

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

   ```
   tmp = read.table("tabbed.txt", header=TRUE, sep="\t")
   ```

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.

```
# Function to do some standard EDA, analysis, and residual
# checks for simple regression.
# Input: x and y are explanatory and response vectors.
#        also the level for the confidence interval.
# Output: returns a list with useful information:
#         coef(ficients), nOK (number of usable data points),
#         p.values, b1 CI and level.
# Side effects: make EDA and residual plots if plots==TRUE.
#
mylm = function(x, y, plots=TRUE, level=0.95) {
  # 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")
  if (!is.numeric(level) || length(level)!=1)
    stop("level must be a single number")
  if (level<=0 || level>=1)
    stop("level must be in (0,1)")

  # 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)
  }

  # Additional info
  p.values = coef(summary(mdl))[,4]
  b1ci = confint(mdl, level=level)[2,]

  # Returning the coefficients and the number of usable data points
  return(list(coef=coef(mdl), nOK=sum(!is.na(x) & !is.na(y)),
              p.values=p.values, b1CI=b1ci, level=level))
}
```