Math 505 Lab 6

1. Write a function called `brown.forsythe` that accepts a vector of residuals e and a vector of variables x and returns the p-value for the Brown-Forsythe test for these vectors.

   The algorithm starts by calculating the median m of x and putting all e_i's for which $x_i \leq m$ into group 1 and all e_i's for which $x_i > m$ into group 2. The following code executes these steps

   ```
   m=median(x)
   e1=e[x<=m]
   e2=e[x>m]
   ```

   Test your function by running the following code, which should return a p-value of 1.61×10^{-18}.

   ```
   xtest=1:100
   etest=c(1:50,(1:50)*10)
   brown.forsythe(etest,xtest)
   ```

2. Import the file `Lab6Data.txt`, whose columns are the variables Y, X_1, X_2, and X_3. The goal of this lab is to perform diagnostics to assess the assumptions of normality and constancy of variance for a model predicting Y from the X_j's, to transform Y if necessary, to assess the transformed model using diagnostics, and to compare the original and transformed models via residual sums of squares.

   We begin by fitting a model and assessing it with diagnostics.

   (a) Fit `model = lm(Y~X1+X2+X3)`, compute the fitted values $\hat{Y}$, and the residuals e. You can use the command `Yhat=predict(model)` to obtain $\hat{Y}$.

   (b) Plot Y vs. $\hat{Y}$. If the model were valid, what would you expect this plot to look like?[1] Does the plot suggest the existence of curvature in the model?

   (c) Plot e vs. $\hat{Y}$. If the model were valid, what would you expect this plot to look like?[2] Does the plot suggest the existence of curvature in the model?

   (d) Now that we know curvature is present, there are two courses of action we can take: transform Y or transform the X_j's, or both. Generally, if there are problems with the errors, we should transform Y, and if the errors are ok, we should transform the X_j's. Let's investigate the errors.

   (e) Plot a qq-plot to check normality of the error terms using the `qqnorm` command.

   (f) Perform the Shapiro-Wilks test to check normality of the error terms using the `shapiro.test` command.

   (g) Based on the results in parts (e) and (f), do the error terms for this model appear to be normal?

   (h) Check constancy of error variance by plotting $|e|$ vs. $\hat{Y}$.

   (i) Check constancy of error variance by performing the test `brown.forsythe(e,Yhat)`.

(j) Based on the results in parts (h) and (i), do the error terms for this model appear to have constant variance?

(k) Does a transformation of Y appear to be necessary?

(l) Finally, calculate the residual sum of squares $\|e\|^2$. Note that this value is

$$\|e\|^2 = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2,$$

so it is similar to a prediction sum of squares (PRESS). It measures the sum of square errors between the predictions $\hat{Y}_i$ and the actual observations Y_i. This number is very large, so to put it in perspective, calculate $\frac{\|e\|^2}{\|Y-\overline{Y}\|^2}$. Assessing the model by this criterion is equivalent to using $R^2 = 1 - \frac{\|e\|^2}{\|Y-\overline{Y}\|^2}$.

3. Write a function `box.cox` that accepts a vector Y, a matrix X, and a vector `lambdarange` and returns the optimal value of λ from `lambdarange` for performing a Box-Cox transformation of Y.

   Here, `lambdarange` is a vector such as $(-3.0, -2.9, -2.8, \ldots, 2.8, 2.9, 3.0)$. For each λ in `lambdarange`, the algorithm will transform Y to obtain a vector W, which is then regressed on X. Assume that X does not have a column of ones, which allows use of the command `tempmodel=lm(W~X)` in the function. After regressing W on X, the residual sum of squares is computed, and the optimal value of λ is the one resulting in the smallest sum of squares.

   If `X=cbind(X1,X2,X3)` and `lambdarange` $= (-3.0, -2.9, -2.8, \ldots, 2.8, 2.9, 3.0)$, then `box.cox(Y,X,lambdarange)` should return 0.3. To refine this value, use

$$\texttt{lambdarange} = (0.20, 0.21, 0.22, \ldots, 0.38, 0.39, 0.40),$$

   yielding $\hat{\lambda} = 0.33$. Refining one more decimal place yields $\hat{\lambda} = 0.328$.

   Additional Hints:

   - Consult the lecture notes for more information on Box-Cox transformations.
   - To create the list $(-3.0, -2.9, -2.8, \ldots, 2.8, 2.9, 3.0)$, make use of the colon command, as in `4:9`$= (4, 5, \ldots, 9)$.
   - To calculate K_2, use the command `K2=(prod(Y^(1/n)))`. In other words, take the nth root of each Y_i before calculating the product, since taking the product first can result in an overflow (numbers that are too big for R).
   - It would be ok to have the function return a matrix listing the SSE for each value of λ, which could then be visually inspected to find the value of λ with the smallest SSE. However, the following hints can help you to return the optimal value of λ instead of a list.
   - If U is a vector, the command `sort(U)` returns U with its values sorted in ascending order. In particular, the first entry in `sort(U)` is the smallest entry in U.

- The command `sort(U,index.return=TRUE)` can be even more helpful, as shown in the following example.

```
> U=c(8,-3,4,1,10)
> U
[1]  8 -3  4  1 10
> sort(U)
[1] -3  1  4  8 10
> sortframe=sort(U,index.return=TRUE)
> sortframe
$x
[1] -3  1  4  8 10

$ix
[1] 2 4 3 1 5

> sortframe$x
[1] -3  1  4  8 10
> sortframe$ix
[1] 2 4 3 1 5
```

- After defining `sortframe=sort(U,index.return=TRUE)`, we get two variables.
  - `sortframe$x` is simply the sorted values of U.
  - `sortframe$ix` shows how the *indices* of U were permuted when U was sorted. For instance, -3 was the second entry in U (the index of -3 was 2). Sorting U moved -3 to the first position, so it moved the index of 2 to the first position. Similarly, sorting U moved 1 to the second position, so it moved its index, 4, to the second position.
- One consequence is that `sortframe$ix[1]` will provide the index of the smallest entry in U.

- This example shows how we can use R code to find which of three fruits has the smallest price.

```
> fruit=c("Apple","Banana","Cantelope")
> fruit
[1] "Apple"     "Banana"    "Cantelope"
> price=c(0.78,0.34,1.28)
> price
[1] 0.78 0.34 1.28
> sortframe=sort(price,index.return=TRUE)
> sortframe$x
[1] 0.34 0.78 1.28
> sortframe$ix
[1] 2 1 3
> cheapindex=sortframe$ix[1]
> cheapindex
[1] 2
> fruit[cheapindex]
[1] "Banana"
```

- Make sure to bring your laptop to the grocery store next time so you can use R to help you find the best prices. ☺
- The `index.return=TRUE` option is very powerful, as it allows you to sort an entire array of data on the basis of one variable as you can do in Excel.

4. Transform Y by defining $\tilde{Y}_i = Y_i^{0.328}$, for $i = 1, \ldots, 100$.

   (a) Fit a model `tmodel` by regressing $\tilde{Y}$ on X_1, X_2, and X_3, and find the corresponding fitted values $\hat{\tilde{Y}}$ and $\tilde{e}$.

   (b) Plot $\tilde{Y}$ vs. $\hat{\tilde{Y}}$ and $\tilde{e}$ vs. $\hat{\tilde{Y}}$. How do these plots compare to those from problem 2? Does curvature appear to exist in the transformed model?

   (c) Investigate normality of the errors for the transformed model.

   (d) Investigate constancy of error variance for the transformed model.

   (e) Do the errors for the transformed model appear to satisfy the assumptions of normality and constant error variance? How do your results compare to those from problem 2?

5. Now let's apply the results from the transformed model to the original variable Y.

   (a) First, create a vector of fitted values for Y by defining $\hat{Y}_i = \hat{\tilde{Y}}_i^{1/0.328}$, for $i = 1, \ldots, 100$, and create a vector of residuals by defining $e_i = Y_i - \hat{Y}_i$, for $i = 1, \ldots, 100$. These are predicted values and residuals for the original model, but they take advantage of the information from the transformed model.[3]

   (b) Plot Y vs. $\hat{Y}$ and e vs. $\hat{Y}$. Did the transformation appear to correct problems with the functional form? [4]

   (c) Finally, calculate $\|e\|^2$, $\frac{\|e\|^2}{\|Y-\overline{Y}\|^2}$, and $R^2 = 1 - \frac{\|e\|^2}{\|Y-\overline{Y}\|^2}$ as in question 2. Which model fits the data better/has a lower residual sum of squares?[5]

Notes

[1]If the model is valid, $Y \approx \hat{Y}$, with any difference being pure noise. Therefore, the plot would hug the line $y = \hat{y}$, with deviations from the line being noise.

[2]If the model is valid, e should resemble white noise, so the plot should hug the horizontal line $e \equiv 0$ with no trend in the deviations. Note that the plot of e vs. $\hat{Y}$ is simply the plot of Y vs. $\hat{Y}$ with the model trend removed, which is why the plot involving e is better at revealing problems in the model.

[3]The e_i's may *not* satisfy normality and homoscedasticity, because they are transformed versions of the $\tilde{e}_i$'s, which do satisfy these assumptions. The model satisfying these assumptions is the *transformed* model. However, the transformed model only allows us to predict $\tilde{Y}$, but what we really care about is $Y = \tilde{Y}^{1/0.328}$, which is why we are estimating Y with $\hat{Y}_i = \hat{\tilde{Y}}_i^{1/0.328}$.

[4]Since the plot of Y vs. $\hat{Y}$ hugs the line $y = \hat{y}$, we conclude that $Y \approx \hat{Y}$, that is, the $\hat{Y}$'s are doing a good job of predicting the Y's. Similarly, the lack of trend in the plot of e vs. $\hat{Y}$ indicates that the functional form has been corrected.

[5]Note that you can't use a model summary or the equation $R^2 = \frac{\text{Var}(\hat{Y})}{\text{Var}(Y)}$ to calculate R^2 on this problem, because our final model is actually nonlinear due to the Box-Cox transformation.