UWEO StatR201 – Winter 2013: Homework 1

Due: Thursday January 24, before Class

Assaf Oron, assaf@uw.edu

Instructions:

- Please submit online in the class dropbox. Accepted formats: *.rmd (R Markdown) and *.pdf. **If you submit an *.rmd file, there's no need to submit plots separately if the plotting is done by the submitted code (even when the question says "submit plots").**
- **Starred (*) questions and question-parts are not required.** You may submit them if you choose, or do any part of them without submitting.
- **Grading is determined chiefly by effort, not by correctness.** If your submission shows evidence of independent, honest effort commensurate with the amount of homework assigned – you will receive full credit.

1. At the start of Lecture 2, I suggest the keywords **Assumptions, Properties, Behavior (A-P-B)**, to use when evaluating and implementing statistical models. In free prose or bullet points, try to summarize what we have learned in Lectures 1-2 about the 'A-P-B' of multiple linear regression. Please try to go beyond what was already learned in StatR101, and add a brief explanation/example to clarify each point. **No need to write more than 1 page total.**

For example: in StatR101 we learned that the random part of *y* is **assumed** to be i.i.d. (independent identically distributed) Normal. In Lecture 2, we learned that the **properties** of regression estimates make them relatively robust to deviations from the Normality part of this assumption. And so forth.

Important note: the trick in modeling is to recognize and keep in mind the *implicit* assumptions, beyond the obvious ones like the example above. We discussed some of these in Lecture 2.

R Trick/Annoyance

2. Download the latest version of `'enhancedGraphic.r'`.

a. As a student suggested, **density plots** on the diagonal might be more helpful than histograms. Replace the histograms with density plots, and check it out using a dataset – submitting both the code and the plot. **Hints: copy the `'panel.hist'` function (included in the file) and use it as a template. The *x* and *y* ranges are controlled by the 'par' statement in the beginning of the function. Either change the *y* range to fit your plot, or normalize the density's *y* values to max out at 1, as done in `'panel.hist'`. *If you get stuck after 30 minutes of trying, send a question.***

b*. Add an option to draw `'qqnorm'` curves on the diagonal, instead of density plots.

c*. The 'boston' dataset has 12 variables, making the panes rather small. Add an option to split the display into several pages, with user input ('press any key...' etc.). for advancing a page. **However,** make sure that the *y* variable (or modeled outcome – in the 'boston' case, home values) *still shows as the last variable in every page.*

<u>Diagnostic Descriptives</u>

3. The Variance Inflation Factor (VIF) collinearity diagnostic is defined in Lecture 2.

a. Write a function to calculate the VIF from scratch for all the covariates.

b. I found two instances of R functions calculating the VIF, in different packages. Download them, compare the results to each other and to your version using the 'boston' dataset, and state your justified opinion regarding which of the two is better for use (i.e., look at the actual code, documentation, function parameters and features, package features, etc.).

4. Document and complete the work done in class (Lecture 2) on the 'boston' dataset:

- Missing values and their location in the dataset.
- Following this and the pairs plot, make-justify-and-implement decisions on data cleaning and transformations.
- Now examine outliers (in home values) and influential points (in the covariates). Highlight any potential trouble data points.
- Using the VIFs, and the pairs plot information, make initial observations and suggestions regarding any covariate that appears to be less useful and can be excluded up front.

<u>Using Lattice</u>

5. Use the lattice command `xyplot`, to examine potential interactions between covariates of any dataset of your own choosing, taken from anywhere. **There is no need to try out all covariate pairs; rather, use your general knowledge about the covariates' meanings for the dataset you use, and show about 2-3 interesting plots.**

For continuous covariates, the `cut` function is useful to break the range into parts. Note that `cut(x,quantile(x))` will cut the covariate into its 4 quartiles.

Example of looking at the modification of the "var1" effect by "var2":

```
xyplot(y ~ var1 | cut(var2,quantile(var2)),data=mydata)
```

Example of looking at the modification of one variable's effect by **two others** – one continuous and one categorical/discrete ('var3'):

```
xyplot(y ~ var1 | factor(var3) + cut(var2,quantile(var2)),data=mydata)
```

Another way:

```
xyplot(y ~ var1 | cut(var2,quantile(var2)),groups=var3, data=mydata)
```

Notes:

- The function decides on the pane arrangement, based on your command and your plotting window's shape. If you don't like it, you can directly control the arrangement using `layout=c(columns,rows)`.
- The plotting symbols can be modified using the usual `cex,pch,col` (don't use 'col' when

using 'groups'!). The title and labels can be set as in `plot`; **please do give informative titles!**

- When using groups, you can show the legend via the option `auto.key=TRUE.`

**** Starred option: add fitted trendlines to the panes.**

6*. Regression simulation.

In Lecture 2, I argue that regression is fairly robust to non-normality. We may or may not have time to test this via simulation in class. In any case, you can test it at home:

a. For the Null case (x,y independent)

b. For a univariate case

c. For a bivariate case, with two mildly correlated covariates (you can still focus on only one of the covariates, to save work).

For each case, generate an **ensemble** of 1000 instances of `y=true model + random noise.` Use sample sizes of n=10 and n=100. In the non-Null case, use noise of a magnitude around ½ of the effect size. Calculate:

- The empirical **bias**: (ensemble average of effect estimate) – (true value)

- The **interval coverage**: in what proportion of the 1000 runs, does **the true value** fall inside **the estimated 95% confidence interval around the effect estimate?** In other words: for each run take the estimate, add and subtract the SE * (the 97.5% t-distribution multiplier), and check if the resulting interval contains the true value.

Noise distributions to use:

- Normal (as reference; this one should behave nicely with near-zero bias and near-95% coverage).

- t with 3 d.f. (use `rt`)

- Exponential (scale=1), subtracting its expectation (1) to remove bias. (i.e., `rexp(n)-1`)

- Cauchy (`rcauchy`), which is equivalent to t with 1 d.f. **Note: this one *should* behave strangely.**

Another note: if you submit an *.rmd file, for this question show only the code in a disabled format, and a table (or simple plots) with the results, after obtaining them offline. I don't want a lengthy simulation to start each time I knit your file :)