**Lab Assignment Week 07**

**I. Objectives**: The objective of the first part of lab is to compare the *linear* model to a couple *nonlinear* models, i.e., models that fit curves other than a straight line through the mean of the data. Comparing specifications by examining model fit statistics is somewhat complicated when the dependent variable has been logarithmically transformed, and interpretation of regression coefficients is less straightforward, but some procedures may be applied to help with interpretation.

The objective of the second part of lab is to compare the *linear* regression model to the *Poisson* regression model when the dependent variable takes on integer values. After showing how to estimate the model, we will explore interpreting coefficients and generating predicted values.

**II. Datasets**: *OECD\_health.dta*, *crab.txt*

**III. Packages**: *foreign*, *white-test.R*, *sandwich*

**IV. Preparation**

1. Open RStudio by double-clicking the icon or selecting RStudio from the Windows Start menu.
2. Perform a Git pull to update the latest scripts and data

3) Open both R scripts by typing *Ctrl*+*O* or by clicking on File in the upper-left corner, using the dropdown menu, and navigating to the scripts in the subdirectory Lab 7 – March 29 in your Project Directory.

**V. Instructions for Part 1**

This lab starts with exploring the life expectancy data that we examined at the beginning of Lecture 13. Begin by running line 1-2 in the R script “*Lab 07 health.R*” to load the packages that you will need to open the datasets. Next, load the health data, as shown in line 4:

health <- read.dta(here("data","OECD\_health.dta"))

Alternatively, you can modify the script to specify the full path to the data:

> health <- read.dta("[full path to data]:/OECD\_health.dta")

If you type in the following two lines of code, you will see a scatterplot of average life expectancy against health expenditures, with a linear regression line. (**Lines 6 and 7 in the script show an alternative method of creating the same plot**):

> plot(health$health\_expend, health$life\_expect, pch=19)

> abline(lm(health$life\_expect ~ health$health\_expend))

Two potential outliers are far below the regression line; these are South Africa, whose expenditures are at the low end of the normal range but whose life expectancy is below 55, and the United States, whose life expectancy is in the normal range but whose expenditures are thousands of dollars above other countries. Run line 3 in the R script to subset the data, omitting South Africa:

> subset <- subset(health, country != "South Africa")

Next, run lines 5–6 to plot the data again without South Africa.

**The new lines 14-29 in the script illustrate the nonlinear nature of the data with two plots, using the *ggplot2* package. If you don’t have this package installed, uncomment the code in line 11 and install the package.**

Run line 31 to change the dimensions of the **Plots** window so that it will display a few plots side-by-side.

Next, run lines 32-33 to examine the normality of the dependent variable (*life\_expect*) visually. The left side displays a histogram of *life\_expect*; any deviation from a bell curve would indicate a non-normal distribution. The right side displays the normal quantile plot of *life\_expect* against the normal distribution; any deviation from a straight line indicates a non-normal distribution.

> hist(subset$life\_expect, freq = FALSE)

> qqnorm(subset$life\_expect); qqline(subset$life\_expect)

For a comparison, run lines 34-35 to examine the normality of the natural log of *life\_expect*.

> hist(log(subset$life\_expect), freq = FALSE)

> qqnorm(log(subset$life\_expect)); qqline(log(subset$life\_expect))

Do you notice any meaningful differences between the distributions of *life\_expect* and *ln*(*life\_expect*)?

You might like to see what the normal quantile plot and histogram look like for a variable that has better distributional properties. Run lines 36-37 for the normal quantile plot and histogram of our main independent variable (*health\_expend*).

> hist(subset$health\_expend, freq = FALSE)

> qqnorm(subset$health\_expend); qqline(subset$health\_expend)

Make sure to run line 38 to reset the dimensions of the **Plots** window.

Before we explore the linear regression results, run either lines 40-41 of the R script (without South Africa) or the following two lines of code for another scatterplot of the *log* of life expectancy against the *log* of health expenditures, with a linear regression line *utilizing the log-transformed variables*.

(**This code for survey question 1)**

> plot(log(life\_expect) ~ log(health\_expend), data = health, pch=19)

> abline(lm(log(health$life\_expect) ~ log(health$health\_expend)))

Does this relationship look more linear than the scatterplot of *life­\_expect* against *heatlth\_expend*?

Let us turn our attention to the bivariate relationships between health expenditures and average life expectancy. Begin by running line 43 to recode the health expenditures data, dividing by $1000. Next, run line 44 to estimate the linear model and display the results of the estimation.

> linlin <- lm(life\_expect ~ health.1k, subset); summary(linlin)

Before moving on, think about how you should interpret the regression coefficient. That is, complete the sentence, “A $1000 increase in health expenditures leads to a \_\_\_\_\_\_ years increase in life expectancy.”

Run line 45 to plot the residuals against the regressor. Do the residuals look well behaved? That is, do the residuals fit our expectations of zero conditional expectation given income and constant variance?

> plot(subset$health.1k, linlin$residuals, pch=19); abline(h=0)

It would not be unwise to run a Breusch-Pagan or White test for heteroscedasticity at this point. These tests should indicate that heteroscedasticity is a problem for the linear-linear regression model.

Even though *health\_expend* was well-behaved, run line 47 to estimate the simple regression of average life expectancy on the natural log of health expenditures (divided by 1000) and display the results.

> linlog <- lm(life\_expect ~ log(health.1k), subset); summary(linlog)

What was the R2 from linear-linear model? \_\_\_\_\_\_\_ What is the R2 from the linear-log model? \_\_\_\_\_\_\_

Are these two quantities directly comparable?

Next, look up the rules for interpretation that we covered in lecture 13. (Wooldridge’s Table 2.3 has been reprinted on page 4 to assist you.) Interpret the simple regression coefficient by completing the sentence, “A one-percent increase in health expenditures leads to a \_\_\_\_\_\_ years increase in life expectancy.”

Run line 48 to plot the residuals. Before moving on, you should examine whether these residuals look better behaved than the residuals from the linear-linear model.

> plot(log(subset$health.1k), linlog$residuals, pch=19); abline(h=0)

In addition to visually inspecting the residuals, you can and should test for heteroscedasticity. Sadly the *white.test* function (in the *white-test.R* script) does not work when we log-transform the variables in coding the regression. As shown with the code in lines 50-52, you could run a Breusch-Pagan or White test by creating the squared residuals and regressing those on *health.1k* and/or *health.1k*2.

Earlier in lab you plotted the natural log of life expectancy against the natural log of health expenditures. Run line 54 to estimate the simple regression *ln*(*life\_expect*) on *ln*(*health\_expend/$1000*):

> loglog <- lm(log(life\_expect) ~ log(health.1k), subset)

> summary(loglog)

What is the R2 from the log-log model? \_\_\_\_\_\_\_ Is this quantity directly comparable to the earlier R2’s? Why or why not?

Using the rules for interpretation that we covered in lecture 13, interpret the regression coefficient by completing the sentence, “A one-percent increase in health expenditures leads to a \_\_\_\_\_\_ percent increase in life expectancy.”

Run line 55 to plot the residuals from this model:

> plot(log(subset$health.1k), loglog$residuals, pch=19); abline(h=0)

You should observe that the residuals from the linear-log and log-log models look quite alike, and they also look quite unlike the residuals from the linear-linear model.

To answer a question that I asked a few paragraphs ago, you cannot directly compare the R2 of a log-log model against the R2 of either a linear-linear or a linear-log model, because the scale of the independent variable has changed. Pay close attention to the residual standard error. The fact that we manipulated the dependent variable dramatically altered the variation around the regression line. We will correct this now.

(For more on R-squared and transformed variables, and cautions about proper use of R-squared generally: <https://data.library.virginia.edu/is-r-squared-useless/> )

Lines 57-60 in the R script walk you through the steps required to compare the models fairly. Recall that the dependent variable in the regression is the natural log of life expectancy, so if you save the predicted values from the model you will have the predicted *log*(*life\_expect*). Line 57 exponentiates the residuals from the log-log model, and line 58 finds the average exponentiated residual. Line 59 creates new fitted values of *life\_expect* by multiplying the average exponentiated fitted value times the exponentiated residual. Finally, line 60 regresses test scores on the predictions based on the log-log model.

> expu <- exp(loglog$residuals)

> alph <- mean(expu)

> lifehat <- alph\*exp(fitted(loglog))

> logcheck <- lm(subset$life\_expect~lifehat); summary(logcheck)

If you chose to use the log-log model, then the Residual standard error and R-squared from the regression in line 60 should be used in place of line 54’s model fit statistics.

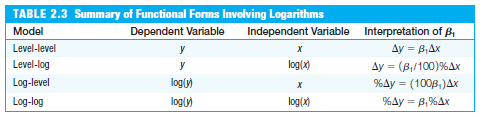
If you want to compare the R2’s from the log-log and level-log models, you can type the following code:

> summary(logcheck)$r.squared; summary(linlog)$r.squared

To compare the models, run line 65 to plot *life\_expect* against expenditures. Line 66 adds fitted values from the log-log model in red. Line 67 adds the curve from the linear-linear bivariate model in purple.

You have estimated three models: level-level, level-log, and log-log. Which has better properties (for instance, in terms of homoscedasticity)? Which seems to fit the data best? Which is easiest to interpret?

A last step before moving on will be to clear some items from memory. For the rest of the lab you will be using data on horseshoe crabs, so clear these data by typing rm(list=ls())



**VI. Instructions for Part 2**

This lab uses data on nesting horseshoe crabs. Begin by loading the horseshoe data, as shown in line 4 in the R script “*Lab07 crabs.R*”:

> crab <- read.table("C:/crab.txt")

The dataset contains data on 173 female horseshoe crabs, each of which has a male crab attached to her in her nest. The dependent variable is *Sa*, which is the number of “satellites” for each female crab; these are other male crabs living nearby.

Run lines 5-6 to name each variable and to remove the first column from the data frame, which simply identifies each observation. Explanatory variables that are thought to affect the number of male satellites include the female crab’s color (*C*), spine condition (*S*), carapace width (*W*) and weight (*Wt*). Let’s begin by examining whether the female crab’s width (*W*) affects the number of satellites (*Sa*).

Run lines 8-10 to attach the data and to view a scatterplot of satellites against carapace width, with a linear regression line.

> plot(crab$W, crab$Sa, pch=19)

> abline(lm(crab$Sa ~ crab$W))

You should observe that female crabs with a wider carapace tend to have more male satellites. However, there also are many female crabs with zero male satellites. Because the value of the dependent variable is truncated to non-negative values, this is almost certain to generate heteroscedasticity.

Run line 12 to estimate the linear regression of satellites on carapace width.

> lmodel <- lm(crab$Sa ~ crab$W); summary(lmodel)

Note that you can run line 13 to estimate the linear regression of satellites on carapace width using maximum likelihood (POLS 6482) instead of least squares (POLS 6481).

If you type the following code, you might be able to observe the predicted heteroscedasticity:

> plot(crab$W, lmodel$residuals, pch=19)

Before we turn our attention fully to the Poisson regression model, we ought to spend a little time examining model fit and interpreting the impact of carapace width. The R2 for the bivariate linear regression of *Sa* on *W* equals \_\_\_\_\_\_\_\_. Next, a one-unit increase in carapace width (*W*) increases the number of satellite mates (*Sa*) by \_\_\_\_\_\_\_\_; this effect (is/is not) \_\_\_\_\_ significant at the .05 level.

To carry out a comparison we’ll use later, calculate the predicted number of satellites for *W* = 25: \_\_\_\_\_\_ and for *W* = 26: \_\_\_\_\_\_ (i.e., multiply the slope coefficient times 25 or 26, and add the intercept).

No new packages are required to run the Poisson regression model, although some caution must go into specifying the model. Run line 15 of the R script to estimate the Poisson regression model

> pmodel=glm(crab$Sa ~ 1 + crab$W, family=poisson(link=log))

Run line 16 to view the Poisson regression model estimated parameters. You will notice that there is *not* an R2 measure presented; we will address this shortly. In the meantime, focus on the estimated coefficient for carapace width (*W*), which equals \_\_\_\_\_\_\_\_; is the effect statistically significant?

> summary(pmodel)

Run line 18 to add the fitted values from the Poisson regression model to the data frame, which is necessary before exploring model fit. (If you wish, you may run line 19 to view the fitted values.) More importantly for our purposes, run line 20 to regress the actual values of *Sa* on the fitted values from the Poisson regression, and to display the R2 from this regression, which equals \_\_\_\_\_\_\_\_. (This process should look familiar to you – it parallels how we assessed model fit for log-linear models!)

> print = data.frame(crab, pred = pmodel$fitted)

> fits = lm(Sa~pred, print); summary(fits)$r.squared

In lecture, we will address predicted values at some length. Run line 22 to display the linear predictors, which fill in the value of *Wi*  into the equation , for each observed value of *Wi*. Next, run line 23 to display the exponentiated linear predictors, i.e.,

> pmodel$linear.predictors

> exp(pmodel$linear.predictors)

Note that I did not type in the previous paragraph and equation, because that would confuse what the Poisson model does. The exponentiated linear predictors, i.e., , equal *µ*(*x*). In order to generate actual predictions, we substitute *µ*(*x*) into the Poisson function:

, *k* = 0, 1, 2, …

where *µ*(*x*) is specific to a given value of *x*. So, if we set the value of *x*, we can generate a value of *µ*(*x*), and then we can generate a vector of predicted probabilities [, , , …]. The idea of comparative statics is: changing the value of *x* changes the vector of predicted probabilities.

In order to clarify how this works, run line 25 to estimate the null model, which has only an intercept. This model treats every female horseshoe crab as if she is identical to all other female horseshoe crabs. If you run line 25, you should see that the null model estimates = 1.0713. If you use your calculator, you should find that *e*1.0713 = 2.919075 ≡ *µ*. Because we are using the null model, this *µ* is unconditional on *x*. Finally, if you substitute *µ* = 2.9191 into the Poisson distribution, you should find the following:

= \_\_\_\_\_\_\_\_

= \_\_\_\_\_\_\_\_

= \_\_\_\_\_\_\_\_

You can continue to carry on for as many (or as few) values of *k* as you like.

With the process for generating predicted probabilities hopefully clearer, let us go back to the bivariate model which accounts for the female horseshoe crab’s carapace width (*W*). The mean and median values of *W* are 26.3 and 26.1, respectively. The first quartile of *W* is 24.9. So let us focus on the effect of increasing carapace width by 1 unit, from 25 (roughly the first quartile) to 26 (roughly the median).

Step 1. Calculate both = \_\_\_\_\_\_\_\_ and = \_\_\_\_\_\_\_\_.

You can run lines 29-30 to check your calculations, but you should also write the equations in this space:

Step 2. Calculate *µ*(25) = = \_\_\_\_\_\_\_\_ and *µ*(26) = = \_\_\_\_\_\_\_\_.

You can run lines 31-32 to check your calculations, but you should also write the equations in this space:

Step 3. Calculate the probabilities of zero satellites if *W* equals 25 and if *W* equals 26:

= = \_\_\_\_\_\_ vs. = = \_\_\_\_\_\_.

The calculations in Step 3. are made easier because . Again, you can run lines 33-34 to check your calculations, but you should also write the equations in this space:

Step 4. Calculate the probabilities of one satellite if *W* equals 25 and if *W* equals 26:

= = \_\_\_\_\_\_ vs. = = \_\_\_\_\_\_

You can run lines 35-36 to check your calculations, but you should also write the equations in the space below; also, this is the last time that the R script contains the code for these calculations…:

Step 5. Calculate the probabilities of two satellites if *W* equals 25 and if *W* equals 26:

= = \_\_\_\_\_\_ vs. = = \_\_\_\_\_\_

Step 6 through Step ∞ would involve calculating as many predicted probabilities as you wish. In any case, if you adopt this form of interpretation, then changes in the value of your explanatory variable are changing the *probabilities* of different numbers of your dependent variable.

Another way of interpreting the coefficients resembles interpretation of the log-linear regression model. We actually use the same equation for interpreting : % Δ*y* = 100·[exp( Δ*W*)–1].

In words, as *W* increases by 1 **unit**, *Sa* should increase by 100·[exp()–1] **percent**. So, we need to exponentiate , subtract 1, and multiply by 100.

Recall that the estimated slope was = 0.16405. Since is positive, will be greater than 1. Therefore, –1 will be positive, so that increases in *W* increase the expected number of satellites.

Calculate = \_\_\_\_\_\_. As width increases by 1 unit, e.g., from 25 to 26, the exponentiated value should be greater by the same ratio. Run line 38 to prove that = \_\_\_\_\_\_ = .

(Line 39 uses the 2nd and 6th observations, which have *W* = 26 and *W* = 25, respectively, to show this.)

If you subtract 1 from = , you get \_\_\_\_\_\_\_. This tells you that the mean value if *W* is 26 is predicted to be \_\_\_\_\_\_% higher than the mean value if *W* is 25.

For a comparison, go back to the linear regression model and calculate = \_\_\_\_\_\_\_\_ and = \_\_\_\_\_\_\_\_ based on that model. The ratio of the latter number to the former number equals \_\_\_\_\_\_\_. If you subtract 1, this tells us that the predicted number of satellites for one additional unit of carapace width is \_\_\_\_\_\_ greater (which is equal to , of course).

Let us conclude our investigation of the coefficient for carapace width by looking at the confidence intervals with ordinary standard errors and with robust standard errors. Type the following code or run line 41 to examine the confidence interval for . Then, run line 42 to examine the confidence interval for – 1. As long as the lower and upper bounds for the confidence intervals both exceed 0, then carapace width has a significant and positive effect on satellites.

Note: You may need to uncomment and run line 43 if you do not already have the package *sandwich* installed.

Either run line 44 to load the *sandwich* package, or run line 51 to load the *car* package, both of which we have used previously. Next, either run line 45 to use the *sandwich* package to create a new variance-covariance matrix of the estimators, or run line 52 to do the same with the *car* package. Both line 45 and line 52 name this new matrix *robvar*.

Run line 46 to create the standard errors of the intercept and the slope, calculated as the square roots of the diagonal elements of the *robvar* matrix. Run lines 47-48 to bind together the coefficient, the new standard errors, the *p* value, and the confidence intervals. Run line 49 to display these in a way that should resemble usual regression output (at least in Stata…). Go back to earlier results in the lab, and compare the standard errors to what was displayed after line 15, and compare the confidence interval to what was displayed after line 40. Because the robust standard error is about 50% larger, the confidence interval is larger, but it still excludes zero by a wide margin.

To clear the **Environment**, type rm(list=ls()) or click on the broom icon.

To clear the **Console** window, type Ctrl-*l*

