**Biostatistics 208 Lab #1, 01/10/19**

The purpose of this lab is to review some exploratory data analysis methods, and introduce running and interpreting a simple linear regression using Stata. Follow the instructions, and remember to keep a log file of your results. Commands will appear in Courier font.

Before you start this lab, download the dataset `lab1.dta` from the course web site.

The dataset is from an HIV clinical trial, ACTG 333. The study measured HIV viral load at baseline and after eight weeks of treatment. In addition, the HIV-1 protease gene was sequenced for all study participants. The objective of the study was to link the changes in viral load to mutations in this gene at codons 10, 71, and 90. The 5 variables in the dataset are:

1. `logvlch`: change in log-10 HIV viral load from baseline to week 8

2. `tx`: treatment assignment (0=hard-capsule Saquinavir, 1= soft-capsule Saquinavir, 2= Indinavir)

3. `codon10`: mutation at codon 10 (1=present, 0=absent)

4. `codon71`: mutation at codon 71 (1=present, 0=absent)

5. `codon90`: mutation at codon 90 (1=present, 0=absent)

6. `logvl0`: baseline log-10 HIV viral load.

Before starting any analyses, first ask Stata to describe the dataset:

```
describe
```

## 1. NUMERICAL DESCRIPTION

A convenient place to start is the `summarize` command, which gives summary statistics for all numerical variables, including the number of observations, mean, standard deviation, minimum and maximum:

```
summarize
```

To get more detailed information about baseline HIV viral load, including the median, quartiles, and other percentiles, use the detail option. Check for any unusual values (remember that they are on the log-10 scale):

```
summarize logvl0, detail
summarize logvlch, detail
```

Note that the `tx` and `codon` variables are also included in the results of the first `summarize` command. This is because they are encoded as numbers with "value labels" as indicated by the results of the `describe` command. However, since they are categorical variables, the `tabulate` (or the abbreviated version `tab`) command should be used to obtain more useful descriptive statistics:

```
tabulate tx
```

```
tab codon10
tab codon71
tab codon90
```

Note that the summaries presented by `tabulate` show the value labels for these variables, but not the actual numeric values in the dataset. These can be viewed using the `codebook` command:

```
codebook codon10 codon71 codon90 tx
```

## 2. SINGLE VARIABLE GRAPHS

### 2.1 Histograms

Various graph commands can be used to depict the distribution of a variable. To get histograms of baseline log-10 HIV viral load as well as the changes in viral load from baseline to 8 weeks, type

```
histogram logvl0, name(lvl10_hist)
hist logvlch, name(lvlch_hist)
```

A graphics window appears with a histogram of the distribution of log-10 HIV viral load values. Note that using the `name()` option with a Stata graphics command keeps the named graph in memory, allowing you to view multiple plots without having to reissue commands. If this option is omitted, each new plot will overwrite the previous one in the default display window.

Use the `bin()` option to change the number of bars on the histogram. What number of bins appears to give the most informative histogram?

### 2.2 Boxplots

Recall that a boxplot gives a graphical representation of key features of the distribution of a continuous variable. The upper and lower limits of the box represent the $75^{th}$ and $25^{th}$ percentiles of the data, while the median ($50^{th}$ percentile) is included as a line through the box. The height of the box marks the interquartile range (IQR) of the observations, and the "whiskers" mark the largest (smallest) observations 1.5 times the IQR outside of the box in either direction. Points falling outside this range are typically represented as dots, and can be viewed as outlying observations.

To obtain a boxplot of the changes in log-10 viral load from baseline to week 8, type:

```
graph box logvlch, name(lvlch_box)
```

Can you read off the approximate 25, 50 and 75th percentile of the data? Do there appear to be outliers?

### 2.3 Q-Q Normal Plots

Histograms and boxplots are useful for assessing distributional shapes, but the `qnorm` command is best for assessing whether a variable approximately follows a Normal distribution. This is useful for linear regression, which relies on the assumption of Normally distributed (at least approximately) residuals for valid inferences. Try the `qnorm` command on both measures of log-10 viral load:

```
qnorm logvl0, name(lvl10_qq)
qnorm logvlch, name(lvlch_qq)
```

Recall that the diagnostic for Normality is that the plotted data should fall approximately on a straight line; Stata superimposes a diagonal line to help you make this judgment. Real data never fall exactly on a straight line; the pattern in this case reflects the short tails in the histogram.

How close is close enough to a straight line? This is a hard question and depends on what technique you want to use, how sensitive it is to the Normality assumption, the nature of the violation (short tails are relatively benign), and on the number of observations (you can get away with more serious violations in a big dataset). One useful exercise is to see what Q-Q plots look like when the data are really from a Normal distribution. You can do this by issuing the commands:

```
gen rannor=rnormal()
qnorm rannor
```

This generates a Normally distributed variable with the same number of observations as the original data set. Because of random variation, the Q-Q plot will not be completely linear, especially in small datasets. Do this a couple of more times to get idea if your Q-Q plot is close to that expected for a plot for exactly Normal data. You will first need to drop the `rannor` variable (or give the new one a different name).

```
drop rannor
gen rannor=rnormal()
qnorm rannor
```

Since the number of observations affects the appearance of the Q-Q plot, you can experiment with datasets of different sizes (e.g. $n$=25) by clearing out the workspace, and using the "`set obs`" command as follows:

```
clear
set obs 25
gen rannor=rnormal()
qnorm rannor
```

You will need to reload `lab1.dta` after doing this since the working dataset has changed. (You may need to issue the `clear` command first to clear the workspace.)

**2.4 Normal Density Plots**

A final tool for visualizing agreement between the distribution of a variable and a normal distribution is a density plot. The estimated density is essentially a smoothed histogram. The following commands plot the density for the variables `logvl0 & logvlch`, along with the corresponding normal density:

```
kdensity logvl0, normal name(lvl10_kd)
kdensity logvlch, normal name(lvlch_kd)
```

The normal density is selected to have a mean and variance that match those from the observed data. Discrepancy between the curves effectively illustrates departures from normality. Leaving off the `normal` option will return only the empirical density estimate for the requested variable.

Note that `kdensity` also allows you to specify the smoothness of the density of the selected variable using the `bw` option. This command is based on a technique called *kernel density estimation* that uses the specified bandwidth (`bw`) and a kernel function to produce what is essentially a smoothed version of a histogram. (The bandwidth is related to the number of bins in a histogram, and the kernel is used to provide smoothness.) The `kdensity` command allows different specifications of both the kernel and the bandwidth as options. For our purposes, the default choices of `bw` and kernel will usually suffice. If you like, experiment by changing the bandwidth to see how it affects smoothness. For example, the following choice returns a rougher-looking density for the log viral load change variable:

```
kdensity logvlch, normal bw(0.1)
```

**Question 1**: Comment on the normality of the variable `logvlch`.

You can also overlay more than one density plot on a single graph to investigate differences in distributions between subgroups defined by additional variables. For example, the following command plots densities for `logvlch` in the two groups defined by the variable `codon90`:

```
twoway (kdensity logvlch if codon90==0, color(blue)) (kdensity
logvlch if codon90==1, color(green)), legend(order(1 "codon90=0"
2 "codon90=1" ))
```

## 3. SIMPLE LINEAR REGRESSION

### 3.1 Simple linear regression with a single categorical predictor

Run a regression model for the dependence of the log-transformed observed change in viral load as a function of the indicator of the presence of a mutation at codon 90:

```
regress logvlch i.codon90
```

(Note that the "`i.`" prefix for the binary predictor isn't technically required here since the variable is coded as 0/1 – retaining it is a good practice in general.)

**Question 2**: Interpret both the coefficients and their accompanying significance tests and 95% confidence intervals.

**Question 3**: Interpret the *R*-squared statistic and the results of the *F*-test given at the top (right) of the model output.

**Question 4:** Review the assumptions required for using the linear model, and comment on their appropriateness in this example.

Note the linear regression model requires that the residuals follow a normal distribution. We can investigate this for the current model by saving the residuals in a variable and using a density or QQ plot to look at them:

```
predict res, residuals
kdensity res, normal
```

**Question 5:** What alternate analysis might be appropriate for assessment of the presence of an association between outcome and predictor if the assumption of a normal distribution for the

outcome variable was violated? Try running an alternate analysis (e.g. Wilcoxon rank-sum test, or a bootstrap CI for the association coefficient) and comment on the results.