Biostatistics 208 Lab #2 1/17/19

The purpose of this lab is to give you practice in dealing with bivariate associations using exploratory methods and single-predictor linear regression.

DATA

We will use baseline data for a subset of the women in the HERS trial who did not have diabetes. Download `lab2.dta` from the official course website.

Double-click on `lab2.dta` to launch Stata and open the dataset, and start a log file to record your work. The variables on the dataset are as follows:

1. `glucose` : fasting plasma glucose in *mg/dL*
2. `exercise` : exercise 3 times/week, coded 1=yes, 0=no (with value labels)
3. `bmi` : body mass index (BMI) in kg/m^2

Observations for women with diabetes are omitted so that glucose, the outcome for this lab, is better behaved. Check to see whether its distribution looks reasonably normal using Q-Q and normal density plots:

```
qnorm glucose
kdensity glucose, normal
```

Note that the linear regression model requires normally distributed *residuals* for correct inferences (especially in smaller samples). Also, the conditional distribution of the outcome for any fixed predictor value(s) is normal. Thus, the check for symmetry/normality here doesn't replace looking at the residuals later. Rather, here we are looking for outliers, possible skewness and to get a sense of the distribution of the outcome variable before modeling. We'll cover transformations and residual diagnostics in a coming class.

PLOTS FOR COMPARING DISTRIBUTIONS

Before conducting any formal analyses, we'll use exploratory methods to view the distribution of fasting plasma glucose in the two exercise groups using both side-by-side boxplots and density plots. Note that these should be approximately normal for the assumptions of linear regression to hold (i.e. we only have two groups of individuals specified in our analysis,).

```
graph box glucose, over(exercise) name(boxplot, replace)
```

Note that the option `name(boxplot)` requests that this plot will be retained when other plots are requested. (Adding the `replace` option to the `name()` statement allows you to replace an existing plot with one of the same name.) While the pattern in the boxplots is often clear, it is not always easy to read the 5-number robust summary statistics (e.g. the median, quantiles, etc.) from the plot. Here are commands to obtain them:

```
bysort exercise: sum glucose, detail
```

A more presentable table is given by:

```
table exercise, contents(med gluc p25 gluc p75 gluc min gluc max
gluc)
```

Also, here is an easy way to get the mean and SD in each group specified by `exercise`:

```
tab exercise, sum(glucose)
```

Now let's use the `kdensity` function to get another view of these two distributions:

```
twoway (kdensity glucose if exercise==0) (kdensity glucose if
exercise==1), legend(order(1 2) label(1 "exercise=0") label(2
"exercise=1")) name(denplot, replace)
```

Question 1: Based on the above results, what do you conclude about the plausibility of the distribution of glucose being approximately normal? What assumptions required by the linear regression model are addressed by these results?

SIMPLE LINEAR MODEL WITH BINARY PREDICTOR

First, run a *t*-test for the difference in average glucose between women who do and do not exercise at least 3 times per week.

```
ttest glucose, by(exercise)
```

Now, fit the equivalent linear regression with glucose as the outcome and exercise as the predictor:

```
regress glucose exercise
```

The following statement gives the sum of the estimated coefficients for the intercept (`_cons`) and `exercise` from the regression that we just ran.

```
lincom _cons + exercise
```

Question 2: Using both the *t*-test and regression output (including the results of the `lincom` statement above), identify the following:

- mean glucose for individuals who exercise at least 3 times a week
- mean glucose for those exercising less than 3 times per week
- the difference in mean glucose between the two groups
- the 95% CI for the difference between the groups
- the *t*-statistic and corresponding *p*-value

Compare the results from the *t*-test associated with the regression output with the default *F*-test (upper right). What hypothesis is the *F*-test addressing? Note that once a model is fitted, the `testparm` command allows additional *F*-tests to be performed that address hypotheses about included variables (e.g. the hypothesis that the true values of particular coefficients are zero, implying that these variables do not contribute to estimating the average outcome). We will consider the `testparm` command in detail next week. In this case, using `testparm` with `exercise` just recaptures the results of the default test provided in the output:

```
testparm exercise
```

LINEAR REGRESSION WITH A SINGLE CONTINUOUS PREDICTOR

Use the `regress` command to run a linear regression of glucose on BMI.

```
regress glucose bmi
```

Question 3: Does there appear to be a relationship? Write a 2-3 sentence summary of the results as you would for a paper.

Using the following commands, obtain the fitted values and residuals using the `predict` command. They will be added to the dataset as new variables called `fitted` and `resid` respectively. Then, to get some practice with some utility commands `gen`, `egen`, and `display`, compute the residual sum of squares (RSS), and compare the result to the output given by the `regress` command. Note that the `gen` command does algebraic operations one observation at a time, while many `egen` commands, this one included, carries out an operation using the values of a given variable for all observations.

```
predict fitted, xb
predict resid, residuals
gen residsq=resid^2
egen rss = sum(residsq)
display rss
```

SCATTERPLOTS WITH LINEAR FIT AND LOWESS SMOOTHS

Use the following two commands to produce a scatter plot of the outcome `glucose` versus the continuous predictor `bmi`, add an unadjusted regression line to the plot:

```
twoway (scatter glucose bmi) (lfit glucose bmi), name(regplot1,
replace)
```

We can also check whether the linearity assumption appears reasonable in this case, using a Lowess smooth in place of the linear fit:

```
lowess glucose bmi, name(regplot2, replace)
```

Since this is a reasonably large dataset, try using a narrower bandwidth of .5:

```
lowess glucose bmi, bw(.5) name(regplot3, replace)
```

Now, produce a plot including both the regression line and the last Lowess estimate:

```
twoway (scatter glucose bmi) (lfit glucose bmi, lcolor(red))
(lowess glucose bmi, sort bw(.5) lcolor(green)), name(regplot4,
replace)
```

Question 4: Do you think the slight upward curvature apparent in the Lowess smooth is meaningful enough to consider additional steps to account for the nonlinearity in further analyses?

TRANSFORMING A PREDICTOR FOR INTERPRETABILITY

In some situations you may want to apply simple transformations to predictor variables, for example if it made more sense to present the increase in mean glucose for a 5-unit increase in BMI, rather than for a 1-unit increase. To do this, generate a new variable `bmi5`, which is just body mass index in units of 5 kg/m^2, then fit a regression model for outcome `glucose` with predictor `bmi5`.

```
gen bmi5=bmi/5
regress glucose bmi5
```

Compare the result of this regression fit to the regression with `bmi` as the predictor. What is similar and what is different, and why?

Now use the following commands to create a "centered" variable `cbmi`, equal to `bmi` minus its sample mean. Then, fit a regression model for glucose with `cbmi` as the predictor:

```
egen meanbmi = mean(bmi)
gen cbmi = bmi - meanbmi
regress glucose cbmi
```

Question 5: What changes in the results as compared to the models using `bmi` and `bmi5` as the predictor? What is the interpretation of the intercept in this model?

Note that the function `std`, used with the `egen` command, also centers a variable on its sample mean, so the new variable again has mean zero. However, this convenient function also rescales the variable to have standard deviation 1, which we may or may not want to do.

Try running the regression with this predictor, and compare the results to the three previous models.

```
egen stdbmi = std(bmi)
sum stdbmi
reg glucose stdbmi
```

THE `i.` SYNTAX FOR USING CATEGORICAL PREDICTORS IN REGRESSION

Note that because `exercise` in our previous example is coded as binary (0/1), no special steps are needed to include it in a linear regression model and interpret the results. Frequently, categorical variables are not coded in this way, and/or may specify more than two levels. In these situations, variables need to be recoded for regression results to retain the interpretation given above. To see this, create a new version of the `exercise` variable, with the value 1 and 2 indicating exercising <3 and ≥3 times per week, respectively:

```
gen exer2 = exercise + 1
tab exer2 exercise
```

The last command shows how the values of `exer2` map to the values of `exercise`. Now fit a regression using this new variable, and note the differences with the model fitted above using the binary version:

```
regress glucose exer2
```

The regression procedure is now treating exercise as a continuous predictor, and although the estimated effect of the change in glucose is the same as the previous result (because there is still a one-unit difference between the defined values), the intercept is different. The latter now does not give the mean among those exercising <3 times per week, but an estimate based on extrapolating the regression line to the zero value (i.e. the observed values for `exer2` are 1 or 2, and the 0 value is not defined). Fortunately, Stata allows categorical variables with arbitrary numeric values to be correctly included in regression models as binary predictors without explicit recoding by preceding their names with "`i.`" in regression commands.

```
regress glucose i.exer2
```

We don't need to worry about this for `exercise` because it's already coded as 0/1. However, suppose we want to fit a model that estimates the difference in average glucose between those exercising less with those exercising more? The following version of "`i.`" allows us to specify an alternate reference category:

```
regress glucose ib1.exercise
```

Introduction to simulation in Stata

Simulation is a useful tool for investigating the impact of assumptions required for valid application of statistical methods. The example below illustrates the effect of increasing sample size on the distribution of the estimated coefficient from a simple linear model including a single binary predictor. Recall (see 3.3.8.2 of the text) that regression coefficients from the linear models should follow a normal distribution as the sample size increases.

The program includes comments (in red preceded by an asterisk) explaining the commands. Given a sample size, the program generates an error (i.e. residual) variable `r` following a known non-normal distribution (in this case, a gamma distribution). Next, a binary predictor `x` with approximately equal number of observations in both categories is generated. These are then used to generate an outcome variable `y` using a linear model with specified values for the coefficients. The chosen model has coefficients $\beta_0 = 1$, and $\beta_1 = 2$. The model is then fitted and the coefficients stored. This program can be run (in this case 1000 times) using the `simulate` command (below).

Before running, let's take a look at an example of the chosen distribution of residuals. The gamma distribution is a family of distributions representing random variables with positive values (e.g. survival times). These typically are skewed, and the shape is controlled by the two parameters called shape and scale. These are specified as arguments to the `rgamma()` function (i.e. `rgamma(`*`shape, scale`*`)`).

```
* comparison of chosen distribution with a normal
set obs 1000
gen r = rgamma(1,0.1)
* center to have approximately zero mean
replace r = r-0.1
* plot
kdensity r, normal
```

Here's the program code (load it by cutting and pasting through the `end` statement):

```
* the program below (begins with the "program" command)
* takes the desired sample size "n" as the sole argument
clear all
program define regprg
args n
drop _all
set obs `n'
* use a skewed version of the gamma distribution to
* generate an error variable – the 2 arguments to the rgamma()
* function are the shape & scale parameters, respectively
gen r = rgamma(1,0.1)
* center the errors to have approximately zero mean
* (Note: the mean for the gamma = shape * scale)
replace r = r-0.1
* generate a binary predictor with approx. equal groups
gen x = rbinomial(1,0.5)
* generate the outcome using slope (intercept) of 2 (1)
gen y = 1 + x*2 + r
* fit model
reg y x
end
```

The statements below run the loaded program for a few different sample sizes and use the `kdensity` and `qnorm` commands to display the distribution of results across the 1000 repetitions. Note that the `simulate` command requests that the coefficients from the fitted regression are returned via the `_b` argument. The program name and sample size argument are specified after the colon. The `nodots` option eliminates unnecessary output.

```
simulate _b, reps(1000) nodots: regprg 10
kdensity _b_x, normal
qnorm _b_x
simulate _b, reps(1000) nodots: regprg 25
kdensity _b_x, normal
qnorm _b_x
simulate _b, reps(1000) nodots: regprg 50
kdensity _b_x, normal
qnorm _b_x
simulate _b, reps(1000) nodots: regprg 100
kdensity _b_x, normal
```

Question 6: Comment on the results. You can easily modify the program to try different versions of the gamma distribution by specifying alternate values for shape and scale.