STATA

**Getting Started**:

1. Go to Stata prompt and click on “Intercooled Stata”

 2. In the command line type: set mem 5000k (then press “enter”)

 Note: for very large files: set mem 500000k (press “enter”)

 then type: set matsize 150 (press “enter” – allows 150 variables)

 3. Click on “file” (upper left corner)

 4. Click on “open”

 5. Click on “down arrow” to use either the “c” or “a” drives

 6. Click so that the desired drive reveals its files

 7. Click on the file you want to load

 8. To execute an operation (e.g., ordered probit) type the following in the

command line: regress (now click on the variable names as listed on the left side of page beginning with the dependent variable – note that using that replacing regress with “fit” will deliver many useful diagnostics - in regression and dichotomous logit/probit a constant is automatically included). Then press “enter.” Other estimators: logit (for the odds ratio – instead of the log of the odds ratio that logit yields - replace logit with logistic), probit (for marginal effects replace probit with dprobit), oprobit, ologit (ordered probit/logit), mlogit (multinomial logit), nlogit (nested logit) and tobit. If using tobit, after the last independent variable type a comma and the letters ll and ul (e.g., tobit ratio ada pover, ll ul). This censors the model at the lower limit and upper limit (i.e., uses the lowest and highest values as the censoring points). You can select a censoring point [i.e., ll (17)]. If censoring is only one side, you may use just one censor point (e.g., tobit ratio ada pover, ll) After running a tobit model, type quadchk (if these results differ greatly from the tobit results it means that you probably shouldn’t use the tobit results). After probit or logit commands you can eliminate all the convergence output by placing “nolog” at the end of the command line: probit vote ada99, bush00, nolog

**Reading an Excel file into Stata:** Make sure that any “missing data” is

denoted with a period (“.”), which is Stata’s missing data code (to do this read the “Recoding and Missing Data” section later in this document). Save the Excel file as a “tab delimited text” file to the “c” drive (or whichever drive your saving the data on - the tab delimited option is in the lower part of the gray box that will appear when you save the file (clicking on “Save as” in the upper left corner). You do not need a file extension. Note: a gray box in the middle of the screen will say that the file is not saved as an Excel file. Click on “yes.” Go into Stata. In the command line type: set mem 5000k and press “enter.” In the command line type: insheet using “C:/senate1.txt” (you need both the quotation marks and the txt on the end). Save as a Stata file by clicking in the upper left corner on “file” and then “save as” and make sure you remove the “\*” and have .dta as a file extension. Thus to save as a file called senate you would need the file name box to read: senate.dta (no asterik).

# Converting a Stata file into Excel: read the data file into Stata; click on

# “Window” (top of the screen); click on “Data Editor”; click on upper left corner to highlight and keep highlighting as you move right; click on “Edit” (top of screen); click on “copy”(or control “c” – i.e., “Ctrl c”); then go into Excel and “paste” (or control “v” – i.e., “Ctrl v”). Check to see that dots (i.e., “.”) representing missing data are in the new Excel file. If not, follow

 “Recoding and Missing Data” section ahead.

**Scientific Notation:** To convert data in scientific notation into regular numbers

you need to specify the number of places to move the decimal for the variables whose scores you’re changing. For three variables and moving the decimal point 11 places to the right type:

format k12ed welfare medical %11.0f (the “0” is number zero)

**Importing a Non-Stata data file (e.g., R or Excel) into Stata:**

1. Download the R data file to your computer. You won’t be able to open it but it will appear in the download files.
2. Go to: <https://gallery.shinyapps.io/rioweb/>
3. Click on “Browse” and then to the right of “Browse” enter the file name for the downloaded Non-Stata data file.
4. Choose Stata as the “Output Format.”
5. Click on “Download” and Stata should open the data file. If this doesn’t work then try to download the file in Excel and then save the file as a tab delimited text file (.txt) and read it into Stata.

**Adding User Written Ado Files to Stata:**

1. To add an ado file (e.g., user written commands) first check to see if it’s already available through Stata. In the command line type: findit clarify (clarify is the name of package you’re trying to import).
2. If the ado file doesn’t appear under the “findit” option then try to download the file from the internet. Often such files will be in a “zip” file which you will need to open. Double clicking on the zip file should open it.
3. You should be able to highlight the files you need to move to Stata.
4. Look for a gold file on the bottom of the screen, click on it and look for directory “windows c:” on the left-side of screen. You may see a gold folder titled “ado” and/or a gold folder “plus”. If so, double click on it and use a “control v” to put the highlighted files in the plus folder. If this doesn’t work do the following: click on the gold folder on the bottom of the screen, then click once on “windows c:”, then double click on “Users”, then double click on “CD”, then double click on “ado”, double click on “plus” and you should be able to move the copied files into “plus” (i.e., via “control v”).
5. Go into Stata and in the command line type: sysdir This will show what folders you have. Among those that appear should be a “personal” folder.
6. To see which ado files you have type: ado dir
7. To activate the user written file (example relogit) type the following in the Stata command line: which relogit
8. Most likely there is a help file that the author the file included to help you. It can be useful to then access this file by typing: help relogit
9. To see if ado files are the current version type: adoupdate If there is a later version type: adoupdate, update

**Long Command Lines**: If the command you are entering is so long that it will

not fit on one line then type /// at the endpoint of the first line and continue the command on a second line (i.e., the line below).

**Reconfigure Screen:** (1)click on “Prefs”; (2) click on “Manage Preferences”; (3)

# click on “Load Preferences”; (4) click on “Factory Settings.” If the something doesn’t appear as you expect, just run several equations, it will clear up. You may need to exit and then go back into Stata.

# Place STATA Results in a Word File: Using the mouse, highlight the Stata

# results you want to transfer into Word. Then use “ctrl c” (control c) to copy the Stata results. Go into Word. Use “ctrl v” (control v) to bring the Stata results into Word. You can “lineup” the Stata results by making sure the font is “currier new” 9 (i.e., font size or number 9). To change the font, highlight the Stata results you have placed in Word. With the highlight on, change the font to “currier new” 9 (i.e., 9 “point”). Now save as a normal Word file.

**Command File or “Do File” – Creating the file:** (1) Go into Stata; (2) load a

dataset; (3) Click on “Window” (at the top of the screen); (4) Click on “Do-File”; (4) Click on “New Do-File”; (5) Write the commands that you want to save (you can include loading the data file) – for example:

version 9

use c:\taxreceiveratiodata.dta

probit grh85 ccus86 par86 medinc85

probit grh87 ccus88 par87 medinc87

**Note:** you could input an Excel spreadsheet that had been saved as

as “tab delimited” text file, as opposed to the “use” and then Stata

 file as above, by replacing the “use” line with:

 insheet using “c:/taxreceiveratiodata.txt” (doesn’t appear to be case

sensitive)

**Further Note**: for commands requiring more than one line, type /// at

The end of the first line and continue the command on the line below.

(6) save by clicking on “File” under “Untitled” in the middle of the screen

**To make changes in your “Do-File” or to use one, or more, commands out of the file do the following:** (1) Click on “Window” (at the top of the screen); (2) Click on “Do-File”; (3) click on “New Do File; (4) Click on “File” under “Untitled” in the middle of the screen; (5) Click on “Open” and double click on the Do-File you want and the commands in the file will appear – you can either change/add commands or highlight existing commands and “copy” a command and then “paste” it in the Stata command line. If you want to replace a term in a do file (e.g., replace “union” each time it occurs with “demcont”) do the following: (1) highlight the commands with the term you want to change; (2) “cut and paste” the highlighted commands into a new do file; (3) in the new do file click on “edit”; (4) click on “find”; (5) click on “replace”; (6) in the appropriate boxes type the work you want to replace and the word to replace it with; (7) highlight the revised commands in the new do file; and (8) paste these new commands from the new file into the old do file.

**To run an entire “Do-File**”: (1) Click on File in the upper left side of the screen; (2) Click on “Do” and then double click on the Do-File you want to run.

**Show Last Command**: “page up” key (“page down” for reverse direction)

**Turn off “More”:** to avoid having to press “more” to see additional results type

“set more off” in the command line or in your do file

**Stopping Results:** to stop results from “whizzing by” press the “Ctrl” key and

while the “Ctrl” key is depressed press “break” (left-side of key pad near the “numbers lock” key

**Saving Results:** to save results in Word type before any results appear type:

 log using f:\T1Results.log (T1Results is the file name on the f drive)

After the last results type: log close. You should then find this file on the f drive. Highlight all results and switch the font to courier new 9 and it will “line up” correctly.

**Nolog – Omit Unwanted Lines Showing Iterations:** maximum likelihood

estimates often show many interations in order to obtain convergence.

To avoid showing these add “nolog” to the end of the command line and after a comma. Not usable for least squares estimators (i.e., rreg but not reg). For example: rreg minwag2 demcont repcont, nolog

xtprobit stminwl2 demcont repcont if year<1980, vce (robust) nolog

**Sampling:** to randomly select 10.5% of the cases from your dataset type:

 sample 10.5 You can save this smaller sample (e.g., for “Small Stata”).

**Finding Variable Descriptions:** after reading in a Stata dataset type: describe

If the person put variable description in the dataset, this command should produce them.

**Select Cases by Scores on a Variable:** logit nafta avmich if divrk>25 (only

uses cases where “divrk” is greater than 25; use >=25 for 25 or greater) If selecting by a particular year you need to use two consecutive equal signs. Thus to list the scores on variable “race” for 1990 type: list race if year==1990 (two consecutive equal signs). To select on two variables at the same time use “&”: logit nafta avmich if party==1 & south==0 Missing data are a great problem with this procedure. You can use two commas if you have multiple sets of instructions after the last independent variable. For example, to specify lower and upper censoring points in tobit and use only observations where variable “state” =1:

tobit ratio ada85 par85, ll ul, if state==1

You can also select cases by scores on a variable using the “keep” command prior to statistical analysis. For example, to select cases scoring “2” on variable “brown” (with possible scores of 1, 2 and 3) you could use the following command: keep if (brown==2). To use cases with scores of 1 and 2 on brown type: keep if (brown== 1 & 2). To use cases with a score of less than 10,000 on a continuous variable “income” type:

keep if (income <10000) or for 10,000 and less type:

keep if (income <=10000). Make sure you don’t save the data because you’ll lose all the dropped observations. If you are using a “do” file to prevent permanent loss of data than have the last command reinstall the original dataset.

**Select Cases by Observation:** In a data set with 100 observations, to use

observations 1, 25-29 and 34-100 type: drop in 2/24 (press “enter”)

 drop in 30/33 (press “enter”) and then run regression

**Selecting Years for Regression:**

regress nt1 demcont repcont if year> 1947

 regress nt1 demcont repcont if year> 1947 & year < 1977

If using either fixed effects or random effects regression you need to specify the subset of years you are using prior to the comma rather than at the end of the command line. Thus,

xtreg nt1 demcont repcont if year > 1977, fe

“fe” for fixed effects and “re” for random effects.

**Deleting Observations from a Dataset**: Read the dataset into Stata. In a

dataset in which “year” was a variable and I wanted to take data from the years 1985, 1987, 1988, 1991, 1992, 1995, 1996, 2000 and 2005 from a dataset that was annual from 1880 to 2008 I did the following:

 drop if year<1985

 drop if year==1986

 drop if year==1989

 drop if year==1990

You get the picture. I don’t know how to drop consecutive years (e.g., 1989 and 1990) in one command. When you’ve deleted all the years you don’t want you will be left with those you do want. Then, using the data editor you can cut and paste the new dataset into Excel. You might “google” both “drop” and “keep” in Stata. There may be easier ways to do this than that described above.

**Creating a Variable by Taking the Highest Score from Two Other Variables:**

To create a variable showing the applicable state minimum wage (either the state’s minimum wage – minwagsh or the federal minimum wage – minwagfh) use the following commands:

gen newminwage=minwagsh

replace newminwage = minwagfh if minwagfh > minwagsh

replace newminwage = minwagsh if minwagfh < minwagsh

**Stacking Data – How to Change it:** To change a dataset stacked by state (e.g.,

observations 1-20 are 20 consecutive annual observations on state #1 with observation 21 being the first observation on state #2) to a dataset stacked by year (e.g., observations 1-50 are the scores for 1985 on each of the 50 states – stnum = state number): sort year stnum

**Transpose Rows and Columns:** xpose, clear Since the xpose command

eliminates letter names for variables, it might be useful to use the xpose command on a backup file while having a primary file containing the variable names. You could then transpose the variable name column or row in Excel and cut and paste the variable names into the xpose file.

**Show scores on a particular observation:** to show the score on variable dlh for

the 19th state (stcode is the variable name) and the year 1972 (year as the variable name) type: list dlh if stcode==19 & year==1972

**Entering a Series of Independent Variables:** You can enter a series of

consecutive independent variables by subtracting the last independent variable from the first. For example, suppose you have 50 state dummy variables that appear in consecutive order in your Stata dataset (e.g., ala, ak, etc. through wy). Instead of entering each independent variable by name, you could type: ala-wy and every variable beginning with “ala” and ending with “wy” would be entered.

**String Variables to Numeric Variables:** Stata cannot read variables which

appear in the Data Editor in “red” (i.e., string variables - letters or numbers that are in red). To convert a “red”/string variable, d3, to a numeric variable (also named “d3”) you need to convert the string variable to another variable (“place” in the example ahead), set this new variable equal to d3 and delete the created variable (i.e., place). The word “force” is part of the process and not a variable name. Proceed as follows:

destring d3, generate(place) force

 drop d3

 gen d3=place

 drop place

**Mathematical Procedures**: addition +; subtraction -; multiplication: \*; division /

**Recoding a Variable:** To convert a score of 1 into 2 and vice versa on variable

grhcum type: recode grhcum (1=2) (2=1) You can also use multiple procedures with nonconsecutive changes in one command. For example if you have a variable “vote” with 4 categories of responses (0, 1, 2, and 3) and want to have 0, 1 and 3 read as 0 while 2 becomes read as 1 type the following: recode vote (3=0) (1=0) (2=1) To put combine two consecutive categories use “/” thus to get 1 and 2 read as “1” type: recode cons (1/2 = 1). To recode a variable that ranged from 0 to 1 (and assumed any value in between) into just 0 and 1, I typed the following:

recode demcont (.0001/.5 = 0) (.5001/1.0 = 1) Note: the recode command does not recognize “<” and “>.” To create and recode a variable that comprises several variables (e.g., Democratic control includes both houses of the legislature plus the governorship):

gen lhdempc= lhdempro

recode lhdempc (.0000/.5 = 0) (.5001/1 = 1)

gen uhdempc= uhdempro

recode uhdempc (.0000/.5 = 0) (.5001/1 = 1)

gen demcont = lhdempc + uhdempc + demgov

recode demcont (.0001/2.999 = 0) (3 = 1)

To recode nonconsecutive numbers use multiple commands. In creating state dummy variables from a state number variable (“stnum” coded 1 through 50) I did the following: gen al=stnum then with a second command: recode al (1=1) (else=0) To recode a percentage variable, “cons,” into three categories (e.g., 0-33=1, etc.) type: recode cons (0/33=1) (34/66=2) (67/100=3) or you can accomplish the same operation as follows: gen cons1 = 1 if cons < 34 (press “enter”)

 replace cons1 = 2 if cons >= 34 (press “enter”)

replace cons1 = 3 if cons >= 67 (press “enter”) (double check which way the arrows point – think through if it should be “<” or “>”)

 **Note:** if you are using the above commands for one value you need two

consecutive equal signs. Thus, gen cons1=1 if cons ==34 (would be if you wanted a score of “1” on cons1 to equal a score of 34 on cons).

**Recoding to Percentiles:** There is a command called the "xtile" command that will recode the observations based on which percentile (ranges)  of a distribution the data are in.
xtile lownetinc5 =  networthlow\_06, nq(5)
Where the new variable would be "lownetinc5" , the old variable would be "networthlow\_06" and the number of categories for the new variable would be indicated by the nq (5) if you wanted 5 categories.

**Absolute Values:** to recode a variable with positive and negative values

 to all positive values type: gen margin= abs(diff) This converted

 negative values on “diff” to all positive for new variable “margin.”

Converting electoral “margin” into two categories (less than 3% = 1 and greater than 3% = 0), I did the following: (1) gen margin3less=.

(2) replace margin3less=1 if margin<.03001

(3) replace margin3less=0 if margin>.03

**Percentage Change Variable**:

 tsset stnum year, yearly (Note: “tsset” means time series data)

 gen income1 = (pcdinc-pcdinc[\_n-1])

gen income2 = income1/pcdinc[\_n-1]

gen pcdincpch=income2\*100

**Mean Annual Score with Panel Data**: to generate an annual mean score on

variable “minwag2adj” the following worked:

tsset stnum year, yearly

egen MEAN=mean(minwag2adj), by(year)

tabstat MEAN, stats(count mean sd p50 min max) by(year)

alternative method:

bysort year: egen aminwag2adj = mean(minwag2adj)

**Converting Monthly Data to Annual Data/Annual Average and Annual High**

**Score:** for a dataset with the following variables: year, month (numbered 1 through 12 – i.e., 12 entries for each year) and minwage (the state’s minimum wage in that particular month of that particular year) to convert monthly minimum wage data to an annual average minimum wage the following commands were used:

collapse (mean) minwage, by(year)

tsset year

To obtain the highest annual score for monthly data for variable minwage (dataset has separate variables for month and year):

# Moving Average Variable: Variable “nonwhpch2” is a 5 year moving average of

# variable “nonwhpch” (i.e., year 6 is the average of the previous 5 years with each year weighted equally). To do this I typed:

tsset stnum year, yearly

tssmooth ma nonwhpch2 = nonwhpch, window(5)

# To transform state ideology from an annual variable

# (stideoan) to a moving average where years 1-4 were the same score (because our the data series we needed extend four years prior to the beginning of the time period for which data were collected) and years 5 and beyond were the average of the current year plus the immediately preceeding 3 years (each year equally weighted) type:

#  xtset stnum year (note: stnum = state number and year=year)

#  tssmooth ma stdeoan\_ma=stideoan , window(3 1 0)

# [note: the “3” is the command immediately above is the number of previous values used, “1” means we are using the current value as well as the previous 3 values in computing the moving average while the “0” means we are using no future values of this variable to create the moving average]

To generate a 12 year moving average of Democratic control (assuming you have already generated the Democratic control variable – e.g., 1= Democratic governor + Democratic majority of both houses of the state legislature in year “t” – how to do this is explained later in this file)

type:

tsset stnum year, yearly

gen demcont12 = (demcont[\_n-12] + demcont[\_n-11] + demcont[\_n-10] + demcont[\_n- 9] + demcont[\_n-8] + demcont[\_n-7] + demcont[\_n-6] + demcont[\_n-5] + demcont[\_n -4] + demcont[\_n-3] + demcont[\_n-2] + demcont[\_n-1])/12

**Interpolation:** this command works even if you have multiple data points to

interpolate between. I used the following to get the percentage of the state population who are African-American annually by state from 1910 to 2010 when I had one score per decade (i.e., variable “afapop” - a score for each state for 1910, 1920, etc.). The data were stacked by state (i.e., variable “stnum” - observation #2 was state#1 in year#2). The following is what I typed:

tsset stnum year, yearly

bysort stnum: ipolate afapop year, generate(afapop\_ip)

The variable “afapop\_ip” is the interpolated variable. If you aren’t using panel data (e.g., the US annually over 50 years) then you wouldn’t use the tsset command and would just type:

ipolate afapop year, generate(afapop\_ip)

**Interpolation if Several Years Missing in a Series Across Units (e.g., TSCS):**

ipolate missing years of 1983-85 for top 1 variable spanning 1917 to 2011 for each of the 50 states. The commands below create a loop for each state.  I guess the num is for the state numbers, and some use virgin islands, dc, etc.  So it is just saying to interpolate for each state, going through the loop.  It interpolates based on previous values, so even if there is no state number 44 for example, it still does it correctly.

for num 1/56: ipolate nt1 year if stnum==X, gen(newtop1\_5X)

gen newtop1i=newtop1\_51 if stnum==1

for num 2/56: replace newtop1i=newtop1\_5X if stnum==X

drop newtop1\_51

**Tables – How to Have Stata Make Them**

For tables with multiple columns try:

 eststo clear

 reg tax cons party stinc

 est sto m1

 reg tax cons stinc

 est sto m2

 **esttab m1 m2, se replace stats(r2 r2\_a r2\_p aic bic N** ll **cmd) starlevels(\* .10 \*\* .05 \*\*\* .01)**

The defaut is t statistics. So, if want those rather than the standard errors just remove “se” from the above line. The above will get standard errors, n, etc. It looks better if you transfer the table from a Stata screen and cut and paste it into Word. In the above esttab command r2\_a is adjusted R2 while r2\_p is pseudo R2. Aic and bic are the Akaike Information Criterion and bic the Bayesian Inofrmation Criterion while ll is log likelihood. To do it differently you’d replace the last line above with:

esttab m1 m2, se using "F:/ChrisDennisRegression.rtf" replace stats(se ll N) starlevels(\* .10 \*\* .05 \*\*\* .01)

While you can use the “rename” option in Stata to change variables you put in the table this option does not recognize time-series operators (i.e., will not change the name of variables such as ld.gini). So, it’s probably easier to just have the table made with variable names in the estimation model and then change them after you’ve cut and pasted the results into Word. To make tables including post-estimation results see the discussion of “Long Run Multipliers” later in this document. The following is the reply by Stata Technical Support to an email I sent about incorporating long run multiplier results in a Stata made table.

From Stata Technical Support: The option -post- with -nlcom- would post estimation results to the standard b and V matrix in -ereturn list-. Then you could use -eststo- and -esttab, append- to append the table to the regression results. Here is an example:

eststo: xtreg ln\_w grade age c.age#c.age ttl\_exp c.ttl\_exp#c.ttl\_exp tenure ///

c.tenure#c.tenure 2.race not\_smsa south, fe

esttab . using result.rtf, se replace stats(r2 r2\_a r2\_p aic bic N ll cmd) ///

starlevels(\* .10 \*\* .05 \*\*\* .01)

eststo: nlcom \_b[age]/(-\_b[ttl\_exp]), post

esttab . using result.rtf, append

Additionally, you could also manually get the \_b, \_se, and pvalue (t

statistics is used in this case) from -nlcom- and use -estadd-. For

example: eststo myxtreg: xtreg ln\_w grade age c.age#c.age ttl\_exp c.ttl\_exp#c.ttl\_exp tenure ///

c.tenure#c.tenure 2.race not\_smsa south, fe

nlcom \_b[age]/(-b[ttl\_exp]), post

scalar b1=\_b[\_nl\_1]

scalar se1=\_se[\_nl\_1]

scalar p1 = 2\*ttail(e(df\_r), abs(\_b[\_nl\_1]/\_se[\_nl\_1]))

estadd scalar myb1 = b1: myxtreg

estadd scalar myse1 = se1: myxtreg

estadd scalar myp1 = p1: myxtreg

esttab myxtreg, se replace scalars(r2 r2\_a r2\_p aic bic N ll cmd myb1 myse1 myp1) ///

starlevels(\* .10 \*\* .05 \*\*\* .01)

**t test comparing time periods:** gen period1v2=.

replace period1v2=1 if year >= 1917 & year <= 1947

replace period1v2=0 if year >= 1948

Now a ttest to compare whether inequality is statistically different between the two periods.

ttest newtop1i, by(period1v2)

**Square Root:** square root of variable pop or spop type: gen spop = sqrt(pop)

(no space between sqrt and (pop)

**Logarithms:** to convert variable X to a log: gen lnx = ln(x)

**Descriptive Statistics -** type**:** summarize cons party stinc (to get the median –

i.e., 50th percentile – type: summarize cons party stinc, detail)

If you have missing data and want the statistics based on only those cases used in a regression do the following: (1) run the regression/logit, etc.; (2) after receiving the results type the following in the command line: estat summarize

**Z Scores:** to standardize cons type: egen consst = std (cons) (i.e., the new variable name is consst or: (1) download “center” command: ssc install center (2) to standardize cons type: center cons (which produces a new variable called c\_cons) (3) then type: summarize cons (to obtain the standard deviation) (4) then type: gen consst=c\_cons/standard deviation from step 3

**Frequencies:** to obtain the frequencies on one variable (e.g. ada94) type:

 tabulate ada94 You can add the mean and standard deviation by:

 tabulate ada94, sum(ada94) You can obtain frequencies over a limit

 range of observation (e.g., 1 through 10) by:

 tabulate ada94 in 1/10

**Addition/Summation Over Time:** given a dataset stacked by state (e.g.,, obs.

1-47 are 47 consecutive years for Alabama, with obs. 48 being the first year for Alaska, etc.) and variable year telling the year and stnum the number of the state, to find the mean on variable lhdem over the 1985-2002 period for each state type:

tabulate stnum if year>1984 & year<2003, summ (lhdem)

Note: If you want to make every observation for a particular year have the average value, 63.21, for variable statepop for that year type:

gen statepop = 63.21 if year==1985

**Cross Tabulation and Measures of Association:**

type: tabulate grh85 par85, row column all (“row” and

“column” yield row and column percentages, “all” yields statistics – Kendall’s tau, gamma and Cramer’s V - you can ask for either row or column percentages, or as above, both – if you want Fischer’s Exact test, add “exact” after “all”). If an error message says “too many values” you may need to recode one or both variables. For a three variable table either: tabulate tax1 cons1 if party==1, row column all

or you need to “sort” by the control variable. For example, to use the two variables above controlling for party type: sort par85 (press “enter”)

by par85: tabulate grh85 grh87, row column all exact (press “enter”)

**Correlation:** correlate tax cons (to correlate tax and cons – can add more

variables) To see statistical significance try: pwcorr tax cons, sig

**Partial Correlation:** pcorr tax cons party stinc

**Kendall’s tau:** ktau tax cons (can add more variables)

**Spearman rank correlation:** spearman tax cons (can add more variables)

**Gamma:** see “cross tabulation and measures of association” above or tabulate

tax cons, gamma or tab tax cons, gam

Note: you can only use two variables at a time and you may need to recode before obtaining a gamma statistic – you can have 5 categories per variable but I don’t know how many more categories are allowed

if you use the procedure listed at the beginning of “Cross Tabulation and Measures of Association” you can avoid recodes.

**Regression:** regress tax cons party stinc (note: dependent variable is “tax”)

 You can replace “regress” with “reg” in the command line above.

**Robust Regression:** rreg tax cons party stinc (note: dependent variable is “tax”)

Note: R-squared is not shown in the output. There may problems in using it. After the program reaches convergence, it goes through one more step in which it creates pseduovalues of the dependent variable using the final set of weights, a scaling factor and a couple of other values. It then uses the pseudovalues as the response variable in an OLS regression. The ereturn values, such as e(r2), e(r2\_a), etc, are left over from that OLS regression model. According to Street, Carroll and Ruppert these auxiliary values that are left over from the pseudovalue regression are not meaningful and should not be used. UCLA has a user written command (rregfit) that, when typed after the robust regression results, will yield goodness of fit measures. To download this command type: findit rregfit

**Regression Without a Constant:** reg tax cons party stinc, nocons

**Seemingly Unrelated Regressions**: If the dependent varibles are tax and

money and the independent variables are party gdp and income type:

sureg (tax money = party gdp income)

As above, but obtain small-sample statistics and use small-sample adjustment

sureg (y1 y2 = x1 x2 x3), small dfk

Also perform Breusch–Pagan test

sureg (y1 y2 = x1 x2 x3), small dfk corr

Model of y1 as a function of x1 and x2 and y2 as a function of x2 and lag of x2 using tsset data

sureg (y1 x1 x2) (y2 x2 L.x2)

**Cronbach’s Alpha:** Cronbach's Alpha examines reliability by determining the

internal consistency of a test or the average correlation of items (variables) within the test. In Stata, the alpha command conducts the reliability test. For example, suppose you wish to test the internal reliability of ten variables, v1 through v10. You could run the following:

alpha v1-v10, item In this example, the item option displays the effects of removing an item from the scale. If you want to see if a group of items can reasonably be thought to form an index/scale you could also use Cronbach’s alpha. For example: alpha a3e a3g a3j a3o, c i The “alpha” score in the “Test scale” row (“alpha” is in the far right column and “Test scale” is a row) should be about .80 (maximum is 1.0) to show a high degree of reliability of the components. However, William Jacoby said the .80 threshold is very high. He would’ve gone lower to .70 (but would never use a scale with a reliability below .5 because you’d have more error variance than substantive variance). If the variables are measured on different scales you may want to standardize them. If so then add “s” to the above command (i.e., alpha a3e a3g a3j a3o, c i s). Since the score for a variable in the “Test scale” column is what the “Test scale” number would be if that variable were deleted, you can maximize the score in the “Test scale” row by deleting any variables whose score in the “alpha” column is greater than the alpha in the “Test scale” row. You can make the scale into a variable by typing: alpha a3e a3g a3j a3o, c gen(anscale) Note: “anscale” is arbitrary (you can pick any name you want – this will now appear as a variable). If you want to exclude those respondents that had a particular score on a variable (e.g., using scores 1 and 2 on variable “petition” but excluding 3) then do the following: alpha a3e a3g a3j a3o if partition==1&2, c i s) For a better understanding see “Intermediate Social Statistics: Lecture 6. Scale Construction” by Thomas A.B. Snijders – saved as adobe file: StataMokkenCronbach.

**Factor Analysis:** You could factor analyze a group of variables (principle

components method by typing: factor a3e a3g a3j a3o, pcf

Look for eigenvalues greater than 1.0 (signifying that the variables in the factor explain more of the variance than individual variables). The entries in the “factor” column are the correlations of that particular variable with the underlying factor. The number in the “cumulative” column tells how much of the variance in the variables you have factor analyzed are explained by all of the factors at the point (i.e., the first entry tells how much of the variance is explained by factor 1 while the second entry tells how much of the variance is explained by factors 1 & 2 together). The score in the “uniqueness” column tells how much of the explained variance is unique to that variable. For example, a score of .61 would indicate that 61% of the variance explained by that particular variable is not explained by the other variables. If you then type “rotate” (default approach – varimax with orthogonal factors – i.e., the factors are not correlated with each other) it will maximize the fit of the dominant variables on that factor. This setting is recommended when you want to identify variables to create indexes or new variables without inter-correlated components. To create new variables (after running “factor” and “rotate”) type: predict factor1 factor2 (you can use whatever names you want to the right of “predict”). They will now appear as variables.

polychoric a3a- a3o (need to add this command by typing findit polychoric) Note: tetrachoric for dichotomous variables

**Mokken Scaling: “**Mokken scaling is an iterative scale-building technique, and

as it is non-parametric is especially suitable for skewed and binary items. It is based on Guttman scales, which are unidimensional, ordinal scales of binary items along a continuum. A positive answer to one item of a certain ‘difficulty’ indicates that all other items of lesser difficulty have also been

answered positively. For example, a positive response to one particular (rare) item indicates that other (more common) items have also been endorsed. Mokken scaling can also use polytomous items, and is a probabilistic version of Guttman scaling. Loevinger’s H-coefficient is used for interpretation. By convention, 0.3 ≥ H < 0.4, 0.4 ≥ H < 0.5 and H ≥ 0.5 indicate weak, moderate and strong scales respectively. Higher

H values indicate higher item discrimination power, and thus more confidence in ordering of respondents. The H-value equals [1 – (observed Guttman errors/predicted Guttman errors)]. Expected Guttman errors are the probability that the items are chosen by chance, while observed Guttman errors are the number of times items are endorsed as if not in an

ordered sequence. Therefore, a coefficient of ≤ .4 demonstrates a scale with items with a 60% rate of Guttman errors. Following a recommended procedure, which involves increasing the coefficient value until the most interpretable solution is found, items that demonstrate poor discriminability are excluded from the scale. Results can be compared to factor

analysis. In general, factor loadings larger than .5 result in H-coefficients greater than .3. Reported scales are ordered in terms of difficulty, ie. the most infrequently endorsed items feature at the top.” (Frank Doyle, et. al., “Exhaustion, Depression and Hopelessness in Cardiac Patients: A Unidimensional Hierarchy of Symptoms Revealed by Mokken Scaling,” Royal College of Surgeons in Ireland, 2011, pp. 29-30). “Loevinger coefficients Mokken (1971) proposed to measure the quality of the pair of items i; j by the Loevinger coefficient Hij = 1 Observed Nij (1; 0)

Expected Nij (1; 0): The ‘expected’ value is calculated under the null model that the items are independent. If no errors are observed, Hij = 1;

if as many errors are observed as expected under independence, then

Hij = 0. For example with two items with means \_Xi: = 0:2; \_Xj: = 0:6,

for a sample size of n = 100 the expected table is

 Xjh = 0 Xjh = 1

 Xih = 0 32 48 80

 Xih = 1 **8** 12 20

 40 60

There are 8 errors in the above table. Now suppose the errors were reduced to just 2 (i.e., 2 in the cell which contains 8). Then Hij = 1 - (2/8) = 0:75. Thus, a good scale should have Loevinger H coefficients that are large enough for all pairs i; j with i < j. Rules of thumb that have been found useful are as follows: Hij < 0:3 indicates poor/no scalability;

0:3 < Hij < 0:4 indicates useful but weak scalability;

0:4 < Hij < 0:5 indicates medium scalability;

0:5 < Hij indicates good scalability.

Similarly, Loevinger’s coefficients can be defined for all pairwise errors for a given item (Hi ) and for all pairwise errors for the entire scale (H).

Although you can run the procedure without specifying a value for Loevinger’s H, you can set levels as in the following command (“c” is the value set). msp a3f a3h a3i a3k, c(.4)

Below are some additional commands that can be used:

msp a3a-a3o, c(.4)

 msp a3a-a3o, pairwise c(.4)

loevh a3a-a3o, pairwise

Mokken scaling can be used in a confirmatory way, with a given set of

items (where the order can be determined empirically) as well as in an exploratory way. In the exploratory method, a set of items is given,

and it is tried to find a well-scalable subset. This is done by first finding the pair with the highest Hij as the starting point for the scale; and by then consecutively adding items that have the highest values of the Loevinger coefficients with the items already included in the scale. This procedure can then be repeated with the remaining items to find a further scale among those. The reliability can be estimated also from the inter-item correlations. The Mokken scaling module is not part of the normal Stata program and must be downloaded. In the command line type: findit msp

**Multidimensional Scaling:** Assume we have information about the American

electorate’s perceptions of thirteen prominent political figures from the period of the 2004 presidential election. Specifically, we have the perceived dissimilarities between all pairs of political figures. With 13 figures, there will be 78 distinct pairs of figures. Rank-order pairs of political figures, according to their dissimilarity (from least to most dissimilar). Multidimensional Scaling (MDS) tries to find a set of k points in m-dimensional space such that the distances between pairs of points

approximate the dissimilarities between pairs of objects. (adapted from William Jacoby and David Armstrong, “Multidimensional Scaling 1”, used at the Measurement, Scaling and Dimensional Analysis short course at the 2011 ICPSR Summer Program – available from William Jacoby’s website). The following command generates the mds default estimation:

mds a3a-a3o, id(partic)

Note: you need to specify an “id” – the name of the variable that gives the observation number. In the command line above, the variable “partic” gives the number assigned to each participant in the study.

**View Scores on a Variable**: list (then click on variable names and press “enter”)

**Graphs/Plots**: run a regression prior to each step below.

to graph residuals by predicted values: rvfplot

 graph residuals by an ind. variable cons: rvpplot cons

plot cons by tax type: plot cons tax

leverage vs. residual sq. plot: lvr2plot

added variable plots (useful for uncovering observations exerting disproportionate influence): avplots

box plots: graph box cons party tax (shows median, 25th, 75th percentile)

histogram of cons: graph twoway histogram cons

scatter plot of tax (on y axis) and cons type: scatter tax cons

###  >>> to get a graph of variables restax and rescon with both dots and

 a regression line type:

graph twoway lfit restax rescons || scatter restax rescons

### Interaction Term: gen nsnt=nsa\*nt1 (+ - / for other mathematical procedures)

### Dummy Variables: Automatic Creation: if you have a variable entitled “year”

### and want to create dummy year variables type: xi i.year or just i.year. To delete this variable type: drop \_I\*

### Interaction Variables: Automatic Creation: to create an dummy variable for

### year and gender type: xi i.year\*i.gender To drop type: drop \_I\*

### Residuals and Predicted Values

 1. run main equation: fit tax cons party stinc

 (I believe you can replace “fit” with regres, logit, probit, etc. – “fit” is just for regression – I don’t think “fit” works with logit, etc.)

 2. predict yhat

 3. gen res=tax-yhat

 4. list tax yhat res

**Outliers**: use the “Residual and Predicted Values” discussion immediately above

to generate the residuals. After doing this you need to see which residuals have the largest absolute size. So, following the discussion above (and assuming your data are organized by state number – variable stnum) type the following in the command line: list stnum res This will show you the value of each residual. Supposing you’re data are by state, your going to drop the largest three residuals and those are for states 15, 17 and 23. Use the drop three successive time. drop if stnum==15 Then repeat this for states 17 and 23. Then re-estimate your results. Be careful NOT to save the data because you will have lost the data for those 3 states. Always have a backup file will all the data in case you accidently save the reduced dataset.

# Stepwise: allows you to specify significance levels and re-estimate the model

# deleting variables that are less significant than the selected threshold. For example: stepswise, pr(.2) hierarchical: regress tax cons party stinc

would mean Stata would estimate the model with all three independent variables and then re-estimate excluding any independent variable that was not significant at the .20 level. Can use with probit, logit, etc.

# Regression Diagnostics/Outliers/Leverage Points: Run a regression with

# regress. Now in the command line type: dfbeta (you’ll see each DF and the name of each independent variable – type “list” and then the name of the independent variable you are interested in). For other diagnostics run a regression. For standardized residuals type: predict esta if e(sample), rstandard (in the command line). You will see “esta” appear in the variable list. Now type: list esta and you will see the values. For studentized residuals do try the following after a regression: Predict estu if e(sample), rstudent (estu will now appear as a variable). For Cooks distance type: predict cooksd if e(sample), cooksd after running a regression. Two user written commands can be useful in identifying outliers: mmregress (will indentify outcome regression outliers and predictor high leverage points - in the command type: findit mmregress – and look for a link such as <http://fmwww.bc.edu/RePEc/bocode/> to download) and smultiv (for robust regression and principal components analysis - same downloading process). Just replace regress or reg with mmregress. To find leverage points estimate regression and in the command line after it runs type: lvr2plot

**Multicollinearity:** after running a regression using regress, type: vif (or: estat vif)

 in the command line. Subtract the number in the 1/VIF column from 1

to obtain the percentage of variation in that independent variable which is explained by all other independent variables. In the VIF column, numbers above 30 indicate high variance inflation (i.e., high multicollinearity).

Doesn’t work in probit/logit. Since at this point you’re only interested in multicollinearity, re-estimate a probit/logit equation in regression and then follow the procedure above. You can also download a user written command “collin” that provides additional information on multicollinearity.

In the command line type: findit collin (and look for a link such as <http://fmwww.bc.edu/RePEc/bocode/> to download). With independent variables cons party stinc the command would be: collin cons party stinc

**Autocorrelation/Correlogram**: **regdw** (replaces regress command and

executes Durbin-Watson test). The data need to be dated – for example, if your data are annual and you have a variable called year, then before you do a regression type: tsset year and press “enter” (or after running regression with “regress” type dwstat). The command **corc** (replaces regress command and executes Cochrane-Orcutt correction for first-

order autocorrelation – note data must be dated, see regdw discussion above). You can save the first observation by using the Prais-Winsten (replace “regress” with “prais”). Prais-Winsten uses a different correction for autocorrelation than Cochrane-Orcutt. To obtain Cochrane-Orcutt results with Prais-Winsten put corc at the end of the command line

(e.g., prais minwag2 demcont repcont massecon, corc). You can obtain a correlogram and specify the number of lags. For example, to obtain a correlogram for the variable “top1” with 12 lags type: corrgram top1, lags(12). If you have a lagged value of the dependent variable as an independent variable you Durbin’s h test. You can download by typing: help durbin h (it will lead you to the downloadable “ado” file). After downloading this file then run your regression and afterward type: durbinh

**Long Run Multipliers**: To obtain long run multipliers and put the results in an

easy to read format type the following:

eststo: nlcom (\_b[l.demcont]/-\_b[ld.minwag2adj]) (\_b[l.repcont]/-\_b[ld.minwag2adj]) (\_b[Ld.minwaggapadj]/-\_b[ld.minwag2adj])

(\_b[l.massecon]/-\_b[ld.minwag2adj]), post

Notice at the bottom of the table of results you should see something

like: (est2 stored) Since the long run multiplier results are post-estimation you need to change the Stata table making procedure. Do the following: (1) use the esttab command to put the non-long run results in a table as discussed in the previous section of this document (“Tables: How to Have State Make Them”); (2) then use the esttab command again but this time use the stored nlcom results to make a second table (e.g., esttab est2, etc.); (3) cut and paste the results from the nlcom table into the first table.

# Heteroscedasticity: run regression replacing regress with fit and then, as a next

# command, type: hettest and press enter. If you have significant heteroscedasticity, use the “robust’ estimation option. Thus, for a “robust” regression type: rreg tax cons party stinc

# Ramsey RESET Test: regress tax cons party stinc After this equation is

# estimated in the command line type: ovtest Note: the RESET test is only a test of whether the model is linear in the original variables. It cannot pick up the influence of other variables.

# Lagged Independent Variable:

You can lag a variable one time period by typing “l.” in front of the variable. Thus, l.ussr should be a one period lag of ussr while l2.ussr would be for two time periods back. You can also do this by typing: gen xussr = ussr[\_n-1] which will create a new lagged variable: xussr. Remember that your data must be dated (see regdw discussion under Autocorrelation above). Lagging will cost one data point, when you run the regression it running it on your sample minus the first observation. There is an “underline” before the “n-1.” You can also lag independent variables by multiple periods. For example, to lag demcont two periods type:

L(1/2)demcont or current period and two period lag: L(0/2)demcont

# First Differences: To create a variable that tells the difference in scores from

# one time period to the next (e.g., 2003-2002), type “d.” in front of the variable. Thus, to difference ussr type d.ussr

# Unit Root Tests: For non-panel data: (1) tsset the data (e.g., tsset year, yearly);

# (2) type dfuller and then the name of the variable you want to test for a unit root. For example: dfuller top1 You could test the first difference of this variable for a unit root as follows: dfuller d.top1 Remember, the null hypothesis is that the IS a unit root. So, if you can reject the null, you can conclude there isn’t a unit root. For panel data: (1) tsset the data (e.g., tsset stnum year, yearly); (2) use either the Im, Pershan and Shin test (e.g., xtunitroot ips top1) or the Hadri’s Lagrange Multiplier test (e.g., xtunitroot hadri top1). The Hadri test requires a strongly balanced panel. So, the test can’t always be used. You can also include lags and a trend variable (e.g., xtunitroot ips top1, lag(aic 4) trend. You need the “xtfisher” command which can be downloaded (findit xtfisher). This is a test to see if a variable is stationary. The null hypothesis is that it has a “unit root.”

# You can use either the Dickey-Fuller or pperron options. For example,

 xtunitroot fisher demcont, dfuller lags(1)

 xtunitroot fisher demcont, pperron lags(1)

Rejecting the null hypothesis of a unit root means the variable is stationary. With error correction models a useful test is to see if the residuals are stationary. If so, you don’t need to worry about some important problems. After estimating the model type: predict yhat (to obtain the residuals). Then type: xtfisher yhat if yhat==1 This will yield results for no lags. To specify lag length change the command to read: xtfisher yhat if yhat==1, lag(2) (for two time periods) If you reject this with confidence then the residuals are stationary. The results should say:

Ho: unit root

 chi2(0) = 0.000 (i.e., less than .001 of a unit root)

#  Prob > chi2 =

#  To test for cointegration in panel data download the xtpedroni command

If you are using an error correction model you use the Westerlund cointegration test: xtwest minwag2 demcont repcont, lags(1)

# Equality of Regression Coefficients: use the suest (seemingly unrelated

# regression) post-estimation command. Estimate model 1, then model 2, and it forms the seemingly unrelated variance-covariance matrix (estimates) for the combined set of coefficients. With that, you can test if some coefficient from model 1 is equal to some coefficient from model 2. An example from panel data for the impact of variable growst (state economic growth) on variable low25 (percentage of income going to the poorest 25% of tax units over the 1913-1978 period vs. the 1979-2003 period for an ECM model (i.e., first differenced dependent variable with a lagged value of the dependent variable as an independent variable: (1) tsset stnum year, yearly; (2) run:

# reg d.low25 l.low25 d.growst l.growst if year<1979

estimates store low25early

 reg d.low25 l.low25 d.growst l.growst if year>1978

estimates store low25late

suest low25early low25late, vce (cluster stnum)

test [low25early\_mean=low25late\_mean]

test [low25early\_mean]d.growst=[low25late\_mean]d.growst

test [low25early\_mean]l.growst=[low25late\_mean]l.growst

# You need to run a separate test for each independent variable. If you are using differenced and lagged values of a variable as an independent variable you would need two tests per variable (i.e., the differenced version of the variable is one variable and the lagged version of the same variable is a second variable).

# Standardized Coefficients: in regression just add ,beta to the end of the

# command line – thus: regress tax cons party stinc, beta

#  To obtain standardized coefficients for probit, logit and multinomial logit

first estimate the desired equation. After the results appear type the following in the command line: listcoef (You may need to download this option from Stata - it will work but may not be built into the package – to download you need to be connected to the internet and type the following in the command line: ssc install listcoef – if that doesn’t work try findit listcoef - then you need click on the appropriate link). If you are interested in the relative value of coefficients, use the coefficients in the “bStdXY” (i.e., the coefficients in this column should be identical to what you receive with the “beta” command in regression). Additionally, two bStdXY coefficients have virtually the same ratio as do the same two bStdX coefficients).

# Marginal Effects: run regress, probit or logit. Then in command line type: mfx

# In probit you can also get the marginal effects of each independent variable by replacing probit with dprobit

**Comparing Models in Probit/Logit (i.e., nesting – like F test in regression**

**for the equality of two R squareds) –** from page 144 of J. Scott Long and Jeremy Freese, Regression Models for Categorical Dependent Variables Using Stata, 2nd. ed. –

probit involvem repcont demcont ablegal fund1 catholic

estimates store fullmodel

probit involvem ablegal fund1 catholic

estimates store smallmodel

lrtest fullmodel smallmodel

**“Weights”/Downloading Data –** If you are downloading data and Stata refuses

to accept the file by saying “weights not allowed” or something like that put quotations around the file name. Thus, if the file name is test, then in the command line type: use “test” (press enter) Put the parentheses around everything expect the word: use (thus use “C:/test” not “use C:/test”)

**Word Responses into Numerical Responses**: If you the responses are words

(e.g., strong agree, etc.) and you want to convert them to numerical values, one suggestion is to cut and paste the dataset into Excel and use the “Find and Replace” option – you can ask Excel to “find” words and then “ replace” with numbers. You can see if there is a numerical code that the words translate into (e.g., strong agree becomes “1,” etc.) by the following procedure: (1) click on “Data” at the top of the screen; (2) click on “Data Editor”; (3) I think you can choose either “Edit” or “Browse”; (4) click on “Tools”; (5) click on “value labels”; (6) click on “Hide all value labels” – numbers should appear at this point. There is a way to permanently convert words into numbers. Go to data editor (choose “edit”, not “browse”) and select (i.e., highlight) the variable (one variable at a time) you are interested in (e.g. q1). Right click the mouse and choose “Value Labels” then choose “ Assign Value Label to Variable 'q1’” and finally, choose “None.” This will erase the labels for the variable and leave the numeric values.

**Merging Datasets**:

Dear Stata Technical Support,
 I=92m a Stata version #11 user (serial number: 30110511993).  I'm trying t=
o merge two state files (statedata.dta and kellywitco.dta).  Statedata is annual data stacked by state from 1880 to 2008 while kellywitco.dta is annual data stacked by state from 1975 to 2006 (e.g., annual data for 1975 through 2006 for Alabama, then 1975 through 2006 for Alaska, etc. - both datasets are stacked this way but the statedata.dta series is a much longer time frame). The states are in the same order in both data sets.  The variable "year" is common to both datasets as is the variable "stnum" (for the number of the states and the state numbers are the same in both data sets).  Could you write out
the specific commands I should use to merge them?  If you give "general directions" I'll never be able to figure it out.  Please write out the commands.  Thanks for your help.

                        Chris Dennis

From: Stata Technical Support [tech-support@stata.com]
Sent: Thursday, August 02, 2012 8:57 AM
To: Chris Dennis
Subject: Re: Merging Datasets (clearer version of message)

Dear Chris,

Below is the solution I propose. The first part is me creating a fake data set
that fits your description and helps exemplify what will happen when the idea
is implemented. It has a variable found across data sets and one that is only
found in kellywitco. For caution I named my files statedataepg and
kellywitcoepg, you will have to eliminate the epg component at the end of the
file name to implement the solution. The solution is just the last two lines of
what I present below:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FAKE DATA \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

quietly {
clear
set obs 129
gen year = \_n + 1879
expand 50
bysort year: gen state = \_n
generate landline = 0
label var landline "mean # of landlines per household"
replace landline = .5 if (year >=1892 & year <1900)
replace landline = 2.5 if (year >=1900 & year < 1999)
replace landline = .5 if  year >=1999
sort state year
save statedata3, replace
keep if year>1974
generate modern = runiform()
label var modern "only in second data"
save kellywitco2, replace
clear
use masterepg
save statedataepg, replace
clear
}

**Robust Regression**: **replace “regress” with “rreg”**

# Omitted Variables Test: run a regression with “fit” instead of regress and then,

# as a next command, type: ovtest (executes Ramsey Reset).

# Predicting Scores from a Combination of Scores on the Independent

# Variables that are *Not* in the Data Set: Since you will need a data set that contains the values on the independent variables, (1) start by going into Excel and creating a file that contains the variable names and the desired scores on the independent variables (remember that row #1 contains the variable names – thus row #2 will be the first row with scores on the independent variables). [Note on Excel: (1) “ctrl z” (i.e., control z) will “undo” what you last did in Excel; (2) to repeat a score down a column: click once and “drag” to highlight and use “ctrl d” to duplicate the number.]; (2) save as both an Excel file and a “tab delimited text file” – thus, click on “save” and save as an Excel file to the “c” drive then click on “save as” and highlight “tab delimited text file” and save to the “c” drive. (3) go into Stata

# and bring in the tab delimited text file you just created by typing: insheet using “C:/deterp.txt” (you need the quotation marks – deterp is the name of the file) (4) now save this file as a Stata file by clicking on “file” (upper left corner) and then “save as” (make sure that the “c” drive is highlighted and you have a put the file name – without an extension – in the file name box (such as: deterp) (5) now bring in the data set on which the coefficient values are to be estimated (i.e., not the set you just created and saved in Stata) (6) run the equation that you want to use to estimate the coefficients (7) bring in the set you created by typing: use “C:/deterp.dta” (8) type: predict yhat (9) click on “window” and then click on “data editor” and the “yhat” column will show the predicted values.

# Multinomial Logit/Probit: Stata automatically selects the most frequent

# occurring outcome as the base outcome (i.e., the outcome all other outcomes are compared to). If you want to designate category 2 as the base type: mlogit grhcum ccus86, base (2)

**Probit/Logit with Panel Data**: First, xtset your data:

xtset stnum year, yearly You should use xtprobit and xtlogit rather than probit or logit. Can’t estimate fixed effects with xtprobit (but can with xtlogit). If you don’t specify random effects, fixed effects or population average, random effects are automatically estimated. Put fe, re or pa after the comma at the end of the command line. Your best method to obtain fixed effects is to use the “fe” extension with xtlogit rather than using either xtlogit with al-wi or xtprobit with al-wi. For example,

xtlogit prevail3 l.repcont l.demcont l.aveinci i.year, fe nolog

However, the “fe” extension in xtlogit *will* work with either bootstrapped or jackknife standard errors but *not* with robust or clustered standard errors.

For example, either of the following could be estimated:

xtlogit prevail3 l.repcont l.demcont i.year, vce (bootstrap) fe nolog

xtlogit prevail3 l.repcont l.demcont i.year, vce (jack) fe nolog

Such an equation may take a very long time to run and the number of observations (e.g., number of “groups”) may be greatly reduced. Since you can, in effect, use fixed effects and robust standard errors with “conditional logit.” For example:

clogit prevail3 l.demcont l.repcont, group(stnum) vce(robust) nolog

The “group” would be analogus to fixed effects. Since clogit requires you to specify a “group” if can’t generate random effects. I received identical results for both of the following:

xtlogit prevail3 l.demcont l.repcont, fe

clogit prevail3 l.demcont l.repcont if year<1967, group(stnum) nolog

For robust st. errors:

xtprobit stminwl1 l.repcont l.demcont l.aveinci, vce (r) nolog

xtlogit stminwl1 l.repcont l.demcont l.aveinci, vce (r) nolog

xtprobit stminwl1 l.repcont l.demcont l.aveinci, vce (robust) nolog

xtlogit stminwl1 l.repcont l.demcont l.aveinci, vce (robust) nolog

Note: you can’t obtain robust standard errors with “fe” on the end of the

command line. However, you can obtain robust standard errors with

both fixed effects and year dummies as follows:

xtlogit stminwl1 l.repcont l.demcont al-wi i.year, vce (robust) nolog

However, with fixed effects and year dummies you may loose many observations.

For clustered standard errors by variable stnum:

xtprobit stminwl1 l.repcont l.demcont l.aveinci, vce (cl stnum) nolog

xtlogit stminwl1 l.repcont l.demcont l.aveinci, vce (cl stnum) nolog

xtprobit stminwl1 l.repcont l.demcont l.aveinci, vce (cluster stnum) nolog

xtlogit stminwl1 l.repcont l.demcont l.aveinci, vce (cluster stnum) nolog

Note: you can’t obtain clustered standard errors with fixed effects

For bootstrap standard errors (takes a long time run – also for bootstrap

and jackknife you need to xtset the data before *each* equation):

xtprobit stminwl1 l.repcont l.demcont l.aveinci, vce (boot) nolog

xtlogit stminwl1 l.repcont l.demcont l.aveinci, vce (boot) nolog

xtprobit stminwl1 l.repcont l.demcont l.aveinci, vce (bootstrap) nolog

xtlogit stminwl1 l.repcont l.demcont l.aveinci, vce (bootstrap) nolog

For jackknife standard errors (takes a long time to run):

xtprobit stminwl1 l.repcont l.demcont l.aveinci, vce (jack) nolog

xtlogit stminwl1 l.repcont l.demcont l.aveinci, vce (jack) nolog

xtprobit stminwl1 l.repcont l.demcont l.aveinci, vce (jackknife) nolog

xtlogit stminwl1 l.repcont l.demcont l.aveinci, vce (jackknife) nolog

# Cox Regression/Event History/Hazard Model/Survival Analysis:

# Note for State Politics Research: In analyses of state’s over time (i.e., consecutive yearly observations on a state) use probit or “xtprobit” (I believe you’ll get the same results) instead of a continuous time procedure (e.g., cox regression) because time is not continuous (i.e., many state legislatures only meet for several months of the year – William Berry suggest this). A good robustness check is to use the “cluster” option discussed below by state [i.e., on the end of the command line put a comma and then cluster(state)]

# From Tom Hayes on Cox Model and testing to see if it’s appropriate

# \*\*\*\*sets the data for duration

# stset durat, failure(adoptinc)

# \*\*\*runs the stcox model, do not include DV, just do stcox and then IVs

# stcox incomerel urban fiscal elect1 elect2 previousa ideology demcont repcont top1 south

# \*\*\*\*\*the following test let's us konw if appropriate model is cox regression. If any variables come out signficant, it may not be best model\*\*\*\*\*

# stphtest, detail

# \*\*\*\*test equality of survivor functions

# sts test adoptinc

**Cox Regression in Stata reports hazard ratios unless you specify you want coefficients by adding , nohr (for no hazard ratio) to the end of the command line**. For example, the following commands produced a hazard ratio:

gen durat1 = year-1891

stset durat1, failure(prevail1)

eststo clear

stcox repcont demcont

eststo

The following commands produced coefficients (since durat1 was already specified above it does not appear in what follows:

eststo clear

stcox repcont demcont, nohr

eststo

It may be the convention to report the coefficients rather than the hazard ratios in papers. If you use the esttab command to have Stata produce a table for Cox Regression results it will report the coefficients, not the hazard ratios.

# stcox (Cox Proportional Hazard Model), and exponential and weibull regressions estimate a hazard rate. The models do not contain the dependent variable in the command line (i.e., only the independent variables are listed for estimation procedure). The dependent variable is listed after the time ordering term in the “stset” (survival set) command. Thus, in the commands that follow the data were stacked by state per year (i.e., a series of consecutive years on one state and then a series of consecutive years on the next state, etc.) and dependent variable in all the analyses was “eitc” (whether the state had an eitc program in that year). The time line for a state ended the first year they adopted the program (i.e., no scores on the dependent variable after that year). A “failure” was coded “1” (i.e., the state adopted an eitc in that year) or “0” (the state did not adopt an eitc in that year).

# The “cluster” option is used on the end of one of the Cox Proportional Hazard models because it was suggested that the accuracy of the error could be improved by clustering by state. The cluster command produces robust standard errors. The “do” file for these commands appears below.

 version 9

 use c:/statepoliticaldata1880to2008, clear

 stset year, failure(eitc)

 stcox lhdem uhdem demgov top1

 stcox lhdem uhdem demgov top1, cluster(state)

 streg lhdem uhdem demgov top1, dist(exponential)

 streg lhdem uhdem demgov top1, dist(weibull)

# Note: in the exponential and weibull regressions the “dist” stands for “distribution.” Also, you can use the cluster command with either the exponential or weibull regressions. Thus, the following would be “okay”:

 streg lhdem uhdem demgov top1, dist(exponential) cluster(state)

# You can avoid much unnecessary convergence output by putting “nolog” and “no show” after the commands. Thus,

 stcox lhdem uhdem demgov top1, nolog noshow

 or: stcox lhdem uhdem demgov top1, cluster(state) nolog noshow

# Unit Root Tests for Panel Data: search typing: findit unit root try xtfisher

# Time-Series Cross-Sectional Data (TSCS) or Panel Data

**NOTE: Stata do file APRDiagnostics.do has the commands for many of the tests discussed in this section.**

 According to Grant and Lebo (Political Analysis 2015) each variable should be *i.i.d.* (independent and identically distributed – i.e., each variable has the same probability distribution and all are mutually independent). If not, analysis can produce spurious results. The equation must be “balanced.” Thus, the analyst should not mix stationary, non-stationary, explosive, fractionally integrated and near-fractionally integrated variables together in the same equation. Therefore, after deciding what model you want to test you then check the stationarity of each variable in the analysis even if the variable is bounded. If you don’t have panel data you can obtain both acf (autocorrelation function) and pacf (partial autocorrelation function) by first using tsset (to name the time variable) and then by using command: corrgram To get useful diagrams of acf and pacf use commands ac or pac followed by the variable name. These commands do not work with panel data. If you are using panel data first “tsset” your data and use the xtpedroni command with your model. This command is a cointegration test. You can’t have any missing data in the equation you estimate. Also if you have missing years you need reconstruct the year variable so there aren’t any missing years. Thus, if years 1982 to 1985 are missing you can’t simply drop each of those years. You would also then need to recode year so that “old” 1986 become 1983, etc. To use the xtpedroni command type (for example):

xtpedroni nt1 demcont repcont demcontn repcontn aveinc

For single variable stationarity tests with panel data for variable nt1:

Levin-Liu-Chu test: xtunitroot llc nt1, trend demean lags(1)

Levin, Lin and Chu (2002). The test assumes that each individual unit in the panel shares the same AR(1) coefficient, but allows for individual effects, time effects and possibly a time trend. Lags of the dependent variable may be introduced to allow for serial correlation in the errors. The test may be viewed as a pooled Dickey-Fuller test, or an Augmented Dickey-Fuller (ADF) test when lags are included, with the null hypothesis that of nonstationarity (I(1) behavior). After transformation, the t-star statistic is distributed standard normal under the null hypothesis of nonstationarity.

Harris-Tzavalis test: xtunitroot ht nt1, trend demean [no lags allowed]

Breitung test: xtunitroot breitung nt1, trend demean lags(1)

Hadri test: xtunitroot hadri nt1, trend demean [no lags allowed]

 Im-Pesaran-Shin: xtunitroot ips nt1, trend demean lags(1)

Fisher test options:

 Augmented Dickey-Fuller: xtunitroot fisher nt1, dfuller trend demean lags(1)

 Phillips-Perron: xtunitroot fisher nt1, pperron trend demean lags(1)

If it is only the Pm statistic which gives a different result you can discard this one because it is for large T, large N (you only have large T). Also p-values below 0.1 are fine, i.e. rejection of the null at the 10% significance level. Also try the "pperron" option instead of dfuller because the Phillips-Perron test is robust to serial correlation (if you have serial correlation in your data). –  [Andy](http://stats.stackexchange.com/users/26338/andy) [Jul 27 '13 at 9:02](http://stats.stackexchange.com/questions/65637/fisher-type-unit-root-test-for-panel-data-results-interpretation-in-stata#comment126971_65725)

**NOTE**: One popular model is the Error Correction Model. Kelly and Witko, AJPS, April, 2012, estimate this model by using xtpcse with the dependent variable being the change (i.e., first difference), using a lagged value of the dependent variable as an independent variable and then using both first differences and lagged values of each of the other independent variables. You can create first differences by mearly putting d. in front of the variable (i.e., d.elderi would be the first difference – the present time period minus the previous time period). You can create lagged variables by putting l. in front of the variable (i.e., l.elderi would be the previous time period score for elderi). Kelly and Witko did not use an adjustment for either autocorrelation or heteroscedasticity. Just a standard xtpcse model as discussed above.

In the following TSCS analysis the data are stacked by state and year (e.g., observations 1-40 are 40 consecutive years for state number #1, observations 41-80 are the same 40 annual observations for state #2, etc.). The dependent variable is top1 and the independent variables are demcont, repcont and top1lag. In some of the models state dummy variables al-wi appear. If the inclusion of the state dummy variables causes perfect collinearity with the unit variable (i.e., stnum), then just run what is described ahead as the “fixed effects” model. The estimators examined are fixed effects (fe), random effects (re) and between effects (be). Before estimating any of these models type: xtset stnum year, yearly (i.e., data are stacked by “stnum” – the state’s number and year with the data being annual). If the resulting output says “strongly balanced” it means that all states had data for all years. If there is not data for some states in some years then it will say “unbalanced.”

Before doing any analysis, download the following tests:

ssc install panelunit

ssc install xtcsd

ssc install xtscc

ssc install xttest2

ssc install xttest3

ssc install xtoverid

ssc install ivreg2

ssc install ivreg28

ssc install ivreg29

ssc install xtfisher

ssc install roblpr (Robinson test for partial integration)

findit lomodrs (Note: roblpr, gphudak, lomodrs, modlpr and kpss are for testing

findit gphudak for fractional integration but won’t work with multiple panels)

findit modlpr

findit kpss

findit xtserial (this test use to be able to be installed through the ssc option but no longer is – if you go through “findit xtserial” you’ll get an option to install it)

findit xttest1 (this test use to be able to be installed through the ssc option but no longer is – if you go through “findit xttest1” you’ll get an option to install it)

findit xtabond2

findit abar (Arellano-Bond test for autocorrelation)

findit xtpedroni (cointegration with panel data)

findit [xtqptest](http://www.stata-journal.com/sjmatches.html?keyword=xtqptest&origkeyword=xtqptest) (autocorrelation test)

findit [xthrtest](http://www.stata-journal.com/sjmatches.html?keyword=xthrtest&origkeyword=xthrtest) (autocorrelation test)

findit [xtistest](http://www.stata-journal.com/sjmatches.html?keyword=xtistest&origkeyword=xtistest) (autocorrelation test)

The following commands immediately ahead explain how to estimate various models and decide between them. The explanation of and rationale for the various models appears after the commands.

xtreg top1 demcont repcont top1lag, fe

estimates store fixed

xtreg top1 demcont repcont top1lag, re

estimates store random

hausman fixed random (if the number to the right of “Prob>chi2” is less than

.05, reject the null hypothesis that the random effects model is preferable – i.e., that the errors are not correlated with the regressors in favor of the alternative hypothesis that the errors are correlated with the regressors and, hence, we should use the fixed effects model)

NOTE: The hausman test can be used with xtregar. Thus, you could estimate the results with xtreg in fe, then run xtserial test (xtserial top1 demcont repcont top1lag) to see if you had first-order autocorrelation. If so, re-estimate the model using xtregar. Save these results (estimates store fixed). Then do the same thing for the random effects model and then run a hausman test on the two xtregar models. However, you can’t use the hausman test with heteroscedasticity adjustments [e.g., xtreg ending in vce(robust) or vce(cluster id)].

Further Note: You could create state dummy variables and estimate fixed effects model all in one command (assuming you have a variable named stnum):

regress top1 demcont repcont top1lag i.stnum (Stata versions prior to 11 require

xi: at the beginning of the command line

xi: regress top1 demcont repcont top1lag i.stnum

Cameron and Trivedi (pp. 267-268) say, “A serious shortcoming of the standard Hausman test is that it requires the RE estimator to be efficient. This in turn require that the (alpha and e – don’t have greek letters) are i.i.d. (independent and identically distributed), an invalid assumption if cluster-robust standard errors for the RE estimator differ substantially from default standard errors.” A user written version of the robust Hausman test can be executed as follows:

xtreg top1 demcont repcont top1lag, re vce(cluster id)

xtoverid (when I ran this test I received the following error message: saved RE estimates are degenerate (sigma\_u=0) and equivalent to pooled OLS)

Other Diagnostic Tests:

Regardless of whether you use fixed effects or random effects (just replace “fe” with “re” in the commands ahead), the following might be useful additional models to estimate:

1.If the Hausman test results suggest that you should use a random effects

model, the LM test helps you decide if you should use OLS instead of random effects. The null hypothesis is that there is no variation among units (states in this example – i.e., no panel effect).

xtreg top1 demcont repcont top1lag, re

xttest0

If the number to the right of “Prob > chi2” is .05, or lower, reject the null hypothesis of no variation between entities in favor of the alternative hypothesis of variation between entities. If the null hypothesis is rejected run OLS (e.g., “reg” command).

xttest1 (you can run this immediately after xttest0)

 xttest1 is an extension of xttest0. It offers several specification tests for

error-component models. It includes the Breusch and Pagan (1980) Lagrange multiplier test for random effects; the Baltagi-Li (1995) test for first-order serial correlation; the Baltagi-Li (1991) joint test for serial correlation and random effects; and the family of robust tests in Bera, Sosa-Escudero, and Yoon (2001). The procedure handles unbalanced panels as long as there are no "gaps" in the series; that is, individual time series may differ in their start and end period but cannot have missing values in intermediate periods. Consider the standard-error component model allowing for possible first-order serial correlation:

 y[i,t] = a + B\*x[i,t] + u[i] + e[i,t]

 e[i,t] = rho e[i,t-1] + v[i,t]

 Typically, researchers are interested in the hypothesis of no random

 effects (Var(u[i])=0), no serial correlation (rho=0), or both. After

 fitting a balanced random-effects model using xtreg, re, xttest0 produces

 seven specification tests:

 1) LM test for random effects, assuming no serial correlation

 2) Adjusted LM test for random effects, which works even under serial

 correlation

 3) One-sided version of the LM test for random effects

 4) One-sided version of the adjusted LM test for random effects

 5) LM joint test for random effects and serial correlation

 6) LM test for first-order serial correlation, assuming no random

 effects

 7) Adjusted test for first-order serial correlation, which works even

 under random effects

 Tests 1, 2, 6, and 7 have asymptotic chi-squared distribution with one

 degree of freedom under the null hypothesis. Test 5 has asymptotic

 chi-squared distribution with two degrees of freedom under the null

 hypothesis, and tests 3 and 4 have standard normal distribution under the

 null.

2. If the results of the Hausman test indicate you should use a fixed effects model

it is important to see if time fixed effects are needed when running a fixed effects model. In order to do this, you first need to create a series of time dummy variables and then execute the test. Assuming you have a variable named “year” and the data are stacked by state (e.g., observation #1 is Alabama in year #1, observation #2 is Alabama in year #2, etc.) the following command will generate year dummy variables:

xi i.year (this created \_Iyear\_1913- \_Iyear\_2003)

xtreg top1 demcont repcont top1lag \_Iyear\_1913- \_Iyear\_2003, fe

testparm \_Iyear\_1913- \_Iyear\_2003

If Prob > F = <.05 then you reject the null hypothesis that all year coefficients are jointly equal to zero. If so, the time fixed-effects are needed.

3.Autocorrelation:

xtreg top1 demcont repcont top1lag, fe

xtserial top1 demcont repcont top1lag

If the number to the right of “Prob > F is .05 or lower reject the null hypothesis of no first-order autocorrelation in favor of the alternative hypothesis the residuals show first-order autocorrelation. If you have first-order autocorrelation replace xtreg with xtregar. Since xtserial does not permit lagged variables you would need to create a lagged variable (not in way Stata would read such as l.demcont but rather some other name). Also, “abar” (another test for autocorrelation - see above) will allow obviously lagged variables.

4. If the results of the Hausman test indicate you should use a fixed effects model

the following test is useful. According to Baltagi, cross-sectional dependence is a problem in macro panels with long time series (over 20-30 years). This is not much of a problem in micro panels (few years and large number of cases). The null hypothesis in the B-P/LM test of independence is that residuals across entities are not correlated. The command to run this test is xttest2 (run it after xtreg, fe): According to Baltagi, cross-sectional dependence is a problem in macro panels with long time series (over 20-30 years). This is not much of a problem in micro panels (few years and large number of cases). The null hypothesis in the B-P/LM test of independence is that residuals across entities are not correlated. The command to run this test is xttest2 (run it after xtreg, fe):

 xtreg top1 demcont repcont top1lag, fe

xttest2

If the number to the right of “Pr” is less than .05 reject the null hypothesis that residuals across entities are independent (i.e., uncorrelated). When I ran the test above, I received an error message that read: “too few common observations across panel. no observations”. Rejection of the null hypothesis could lead to using the “robust” standard errors model shown below. Also, see next test discussed.

5. As mentioned above, cross-sectional dependence is more of an issue in

macro panels with long time series (over 20-30 years) than in micro panels. Pasaran CD (cross-sectional dependence) test is used to test whether the residuals are correlated across entities. Cross-sectional dependence can lead to bias in tests results (also called contemporaneous correlation). The null hypothesis is that residuals are not correlated.

 xtreg top1 demcont repcont top1lag, fe

xtcsd, pesaran abs

If the number to the right of “Pr” is less than .05 reject the null hypothesis that the residuals are not correlated. When I ran this test I received the following error message: “The panel is highly unbalanced. Not enough common observations across panel to perform Pesaran's test. insufficient observations”. Had cross-sectional dependence be present Hoechle suggests to use Driscoll and Kraay standard errors using the command xtscc.

 xtscc top1 demcont repcont top1lag, fe

Note: even though I received the error message above when I ran the xtcsd test, I did not receive an error message when I executed the xtscc command.

Possible Models:

1.The model with heteroscedasticity:

xtreg top1 demcont repcont top1lag, fe vce(robust)

Note: The vce(robust) option should be used with caution. It is robust in the sense that, unlike default standard errors, no assumption is made about the functional form (A. Colin Cameron and Pravin K. Trivedi, Microeconometrics Using Stata, revised ed., p. 334). From their discussion, this can lead to problems. This is a reason for the cluster option discussed ahead.

2.The model with first-order autocorrelation:

xtregar top1 demcont repcont top1lag, fe

3.The model with both first-order autocorrelation and heteroscedasticity:

xtreg top1 demcont repcont top1lag, fe vce(cluster id)

If you have reason to below the disturbances are related to one of the variables use that variable after “cluster.” For example, there may be correlation within states but not across states. Thus, observations in different clusters (e.g., states) are independent but observations within the same cluster (e.g., states) are not independent.

xtreg top1 demcont repcont top1lag, fe vce(cluster stnum)

Note: if you replace “stnum” with “id” (cluster id) you get very different standard errors. The “id” is suppose to represent an individual whereas i.i.d. means(independent and identically distributed. I would’ve thought it would be an individual state and, thus, would be the same as (cluster stnum) but this is not the case.

4.The model with heteroscedastic, contemporaneously correlated cross-

sectionally correlated, and autocorrelated of type AR(1) [Beck & Katz’s panel corrected standard errors]. Assumptions: (1) if I assume no heteroscedasticity - panels(uncorrelated); (2) if I assume the variances differ for each unit – panels(hetero); (3) if I assume that the error terms of panels are correlated – panels(correlated)]; (4) if I assume no autocorrelation – corr(independent); (5) if I assume each panel has first-order serial correlation and the correlation parameter is the same for all groups – corr(ar1); (6) if I assume first-order serial correlation where the correlation parameter is unique for each panel – corr(psar1).

xtpcse top1 demcont repcont top1lag

Note: You can run the xtserial test immediately after running the xtpcse model above to see if you should make an adjustment for first-order autocorrelation.

xtserial top1 demcont repcont top1lag

If the number to the right of “Prob > F is .05 or lower reject the null hypothesis of no first-order autocorrelation in favor of the alternative hypothesis the residuals show first-order autocorrelation.

If you do have first-order autocorrelation try:

xtpcse top1 demcont repcont top1lag, corr(ar1)

xtpcse top1 demcont repcont top1lag, pairwise corr(ar1)

Pairwise includes all available observations with nonmissing pairs. The alternative appears to be casewise (I’m assuming that’s what you get with “corr(ar1)” Pairwise corr(ar1) seems to make little difference vs. corr(ar1) If you can reasonably assume that all units (i.e., states in this case) have the same first order autocorrelation (i.e., constant autocorrelation across panels, i.e., states) then add a comma and correlation (ar1) after the last independent variable. In something I read Beck and Katz argue in favor of this autocorrelation structure and against panel specific autocorrelation (i.e., where each unit, state in this case, could have a different value for rho). For a panel specific (not the same for each unit – e.g., state) adjustment for first-order autocorrelation is:

xtpcse top1 demcont repcont top1lag, corr(psar1)

 xtpcse top1 demcont repcont top1lag, correlation (ar1) rhotype (tscorr)

Note: may also incorporate state dummy variables

xtpcse top1 demcont repcont top1lag al-wi (or do this by using the “i”

variable – thus, xtpcse top1 demcont repcont top1lag i.stnum (where stnum is the state number)

. You can also estimate a heteroscedastic panel corrected standard errors model by putting “het” after the comma:

xtpcse top1 demcont repcont top1lag, het

You can estimate a heteroscedasitic first-order autocorrelation panel corrected standard errors model:

xtpcse top1 demcont repcont top1lag, het corr(ar1)

You may receive the following error message: no time periods are common to all panels, cannot estimate disturbance covariance matrix using casewise inclusion. Sometimes adding an independent variable will overcome this problem.

5.Alternative model to xtpcse with heteroscedastic, contemporaneously

correlated cross-sectionally correlated, and autocorrelated of type AR(1) (N - number of units – states in this example, T – time - for feasibility – this model tends to produce optimistic estimates of standard errors – also, may not work with unbalanced panels – e.g., different years available for different states - xtpcse appears “safer”]

xtgls top1 demcont repcont top1lag, panels(heteroscedastic) corr(ar1)

 -you could also have put , panels(hetero) corr(ar1) or corr(psar1)

6. If you have moving average autocorrelation:

newey top1 demcont repcont top1lag q1cus, lag(1) [might only work if

panels are balanced – e.g., same years available for all states]

7.If you have moving average autocorrelation and cross-sectional dependence:

xtscc top1 demcont repcont top1lag q1cus, lag(1)

8. Bootstrap standard errors:

xtreg top1 demcont repcont top1lag, fe vce(boot)

The examples ahead use “long panels” (i.e., where the number of years is much greater than the number of states).

Fixed effects, autocorrelation but no heteroscedasticity:

 xtgls top1 demcont repcont top1lag al-wi, corr(ar1)

Fixed effects, autocorrelation and heteroscedasticity:

xtgls top1 demcont repcont top1lag al-wi, panels(heteroskedastic) corr(ar1)

If you want flexible autocorrelation across states and a distinct ar(1) process for the error in each state replace corr(ar1) with corr(psar1) I believe the “p” is panel specific. It the number of time periods is not much larger than the number of states use the more restrictive corr(ar1) option.

If you want random effects omit the dummy variables (al-wi) in the above command lines. Much of the above discussion was taken from Microeconometrics Using Stata, revised edition, by A. Colin Cameron and Pravin K. Trivedi, Stata Press, 2010 and “Panel Data Analysis Fixed and Random Effects (using Stata 10)” by Oscar Torres-Reyna (available at www.princeton.edu/~otorres).

Stationarity in Panel Models/Error Correction Models: You need the “xtfisher” command which can be downloaded (findit xtfisher). This is a test to see if a variable is stationary. The null hypothesis is that it has a “unit root.” Rejecting the null hypothesis of a unit root means the variable is stationary. With error correction models a useful test is to see is the residuals are stationary. If so, you don’t need to worry about some important problems. After estimating the model type: predict yhat (to obtain the residuals). Then type: xtfisher yhat if yhat==1 This will yield results for no lags. To specify lag length change the command to read: xtfisher yhat if yhat==1, lag(2) (for two time periods) If you reject this with confidence then the residuals are stationary. The results should say:

Ho: unit root

 chi2(0) = 0.000 (i.e., less than .001 of a unit root)

 Prob > chi2 =

Models/Commands that “run” but not sure why they should be used:

xtreg top1 demcont repcont top1lag, i(id) fe

reg top1 demcont repcont top1lag i.year i.stnum

xtreg top1 demcont repcont top1lag i.year, fe

xtreg top1 demcont repcont top1lag i.year i.stnum, fe

 xtgls top1 demcont repcont top1lag, i (id)

Discussion

Long panels are where time is much greater than the number of units (e.g., states). Short panels are the opposite. Use fixed-effects (FE) whenever you are only interested in analyzing the impact of variables that vary over time. FE explore the relationship between predictor and outcome variables within an entity (country, person, company, etc.). Each entity has its own individual characteristics that may or may not influence the predictor variables (for

example being a male or female could influence the opinion toward certain

issue or the political system of a particular country could have some effect on

trade or GDP or the business practices of a company may influence its stock

price). When using FE we assume that something within the individual may impact or bias the predictor or outcome variables and we need to control for this. This is the rationale behind the assumption of the correlation between entity’s error term and predictor variables. FE remove the effect of those time-invariant

characteristics from the predictor variables so we can assess the predictors’ net

effect. Another important assumption of the FE model is that those time-invariant

characteristics are unique to the individual and should not be correlated with

other individual characteristics. Each entity is different therefore the entity’s

error term and the constant (which captures individual characteristics) should

not be correlated with the others. If the error terms are correlated then FE is no

suitable since inferences may not be correct and you need to model that

relationship (probably using random-effects), this is the main rationale for the

Hausman test (presented later on in this document). Control for time effects (in our example year dummy variables) whenever unexpected variation or special events my affect the dependent variable. The fixed-effects model controls for all time-invariant differences between the individuals, so the estimated coefficients of the fixed-effects models cannot be biased because of omitted time-invariant characteristics…[like culture,religion, gender, race, etc] One side effect of the features of fixed-effects models is that they cannot be used to investigate time-invariant causes of the dependent variables. Technically, time-invariant characteristics of the individuals are perfectly collinear with the person [or entity] dummies. Substantively, fixed-effects models are designed to study the causes of changes within a person [or entity]. A time-invariant characteristic cannot cause such a change, because it is constant for each person.”

One alternative to a fixed effects model is a random effects model. The rationale behind random effects model is that, unlike the fixed effects model, the variation across entities is assumed to be random and uncorrelated with the predictor or independent variables included in the model: “…the crucial distinction between fixed and random effects is whether the unobserved individual effect embodies elements that are correlated with the regressors in the model, not whether these effects are stochastic or not” [Green, 2008, p.183] If you have reason to believe that differences across entities have some influence on your dependent variable then you should use random effects. An advantage of random effects is that you can include time invariant variables (i.e. gender). In the fixed effects model these variables are absorbed by the intercept. The random effects model is:

Yit = *βX*it + α + *u*it + *ε*it (“u” is the between entity error and “e” is the within entity error)

Random effects assume that the entity’s error term is not correlated with the predictors which allows for time-invariant variables to play a role as explanatory variables. In random-effects you need to specify those individual characteristics that may or may not influence the predictor variables. The problem with this is that some variables may not be available therefore leading to omitted variable bias in the model.

The between estimator uses only between or cross-section variation in the data. Because only cross-section variation in the dta is used, the coefficients of any individual-invariant regressors, such as time dummies, cannot be identified. It is seldom used.

To decide between fixed or random effects you can run a Hausman test where the null hypothesis is that the preferred model is random effects vs. the alternative the fixed effects (see Green, 2008, chapter 9). It basically tests whether the unique errors (*ui*) are correlated with the regressors, the null hypothesis is they are not. Run a fixed effects model and save the estimates, then run a random model and save the estimates, then perform the test. (Hausman test is near the beginning of TSCS material)

Older Discussion/Comments from Neal Beck:

I think it is desirable to use the panel corrected standard errors that Beck and Katz (APSR, 1995) developed. STATA has the necessary commands. The impression I got from the STATA manual was that you should only use panel corrected standard errors if the number of time periods is equal to, or greater, than the number of panels. In a study of the 50 states over 23 years, I would seem to violate this assumption. I asked Neal Beck about it and here is his reply, “PCSE’s work in this case. The only issue is T (i.e., time periods) being large enough (say over 15). Clearly you have that. FGLS (parks) does not work here, but you do not care.” In a subsequent message in which I asked about using PCSE’s with both random effects and fixed effects models, Beck replied as follows: “PCSE’S are totally orthogonal to the question of effects. For bigger T TSCS, re and fe are pretty similar. All the issues where re wins are for small T. But whether or not you need effects has literally nothing to with PCSES.” In a later e-mail Beck mentioned that panel corrected standard errors cannot really be done with random effects models.

### Time Series Models

 Dickey-Fuller Unit Root Test: dfuller loginc2, regress

Note: adding “regress” in the above command means you will receive the regression result in addition to the Dickey-Fuller test results Correlogram of ACF and PACF: corrgram loginc2

Note: to obtain pointwise confidence intervals try

the following: ac loginc2, needle (for ACF)

or: pac loginc2, needle (for PACF)

 Also: for a correlogram on first differences try:

corrgram D.loginc2 (Note: there is no space between D. and loginc2)

**ARIMA Models**:

Structural (i.e., substantive independent variables with an arima error process with both one autoregressive and one moving average term) arima loginc2 dempreslag, ar(1) ma(1)

Non-Structural Model (i.e., current value of the dependent variable is entirely in terms of an autoregressive and moving average process) arima D.loginc2, ar(1) ma(1) (Note: no space between D. and loginc2) (for non-differenced dependent variable omit “D.”)

**Bootstrapping** (Jeffrey Harden, SPPQ,“A Bootstrap Method for Conducting Statistical Inference with Clustered Data.” State Politics & Policy Quarterly 11(2):223–246. The main drawback to using the BCSE method is that calculating the standard errors for a model multiple times will produce different estimates each time because different bootstrap samples are drawn each time. This leads to the question of how to report results in journal articles and presentations. This is an issue with any simulation-based technique, such as conventional bootstrapping

or Markov Chain Monte Carlo (MCMC) methods. The important points to note are that the analyst controls the number of simulations and adding more simulations brings the estimate closer to the true value. In the case of BCSE, increasing the number of bootstrap replications (B) shrinks the variation between calculations.1 From this, the analyst could choose the number of digits to round off from each standard error estimate when reporting results, and increase B until multiple calculations return the same rounded value.2 In addition, the seed of the random number generator can be set to duplicate the same bootstrap samples in later iterations. In the article, I identify some differences that make BCSE preferable to the CLRT method: (1) CLRT does not calculate a full covariance matrix of the parameter estimates, (2) CLRT is driven by a much different philosophy than the other methods I examine, and (3) CLRT is more difficult to implement than BCSE.

Example Stata Code

1 \* Method 1

2 regress y x1 x2 x3, vce(bootstrap, rep(1000) cluster(cluster))

3 \* Method 2

4 bootstrap, rep(1000) cluster(cluster): regress y x1 x2 x3

5 predict residuals, residuals

6 loneway residuals cluster

**STATA “DO” Files**

**The “do file” below was for an annual data set (1976-2003) stacked by state (i.e., observation #1 was Alabama for 1976, observation #2 was Alabama for 1977, etc.) where I wanted the average score on id1 for the**

**1980-89 period – the average score on id3 for the same period for all states. Also, the same procedure for the 1980-89 period.**

use "C:\EriksonWrightMcIverUpdate.dta", clear

keep stateid year id1 id3

sort state year

list in 1/10

foreach val in 1 3 {

egen eighties\_id`val' = total(id`val') ///

if year > 1979 & year < 1990, by(state)

egen nineties\_id`val' = total(id`val') ///

if year > 1989 & year < 2000, by(state)

sort stateid eighties\_id`val'

by stateid : replace eighties\_id`val' = ///

eighties\_id`val'[\_n-1] if eighties\_id`val' == .

sort stateid nineties\_id`val'

by stateid : replace nineties\_id`val' = ///

nineties\_id`val'[\_n-1] if nineties\_id`val' == .

replace eighties\_id`val' = eighties\_id`val'/10

replace nineties\_id`val' = nineties\_id`val'/10

}

gen ideology8089 = eighties\_id1 - eighties\_id3

gen ideology9099 = nineties\_id1 - nineties\_id3

drop eight\* nine\*

**The following commands were used to find the number of years between 1989 and 2002 that the Democrats were in “control” (i.e., had a majority of both houses of the state legislature and the governorship) in each state. Stnum was simply the state number (i.e., Alabama was state #1)**

use "C:\StatePoliticalData19492006.dta", clear

gen flag = uhdem>uhrep & lhdem>lhrep & demgov ==1 & year>1988 &

 year<2003

egen demcontrol=total(flag), by(stnum)

tabulate stnum, summ (demcontrol)

**The following commands were used to obtain the average Democratic**

**strength in each state’s government over the 1989-2002 period (lower house worth 25%, upper house worth 25% and the governorship worth 50% -**

**i.e., the maximum value would be 1.0)**

use "C:\StatePoliticalData19492006.dta", clear

gen dem\_strength = .

levelsof stnum

 foreach st in `r(levels)' {

 local total\_percent=0 //

 foreach val in lh uh {

 qui sum `val'rep if stnum == `st' & year>1988 &

 year<2003

 local reps = r(sum)

 qui sum `val'dem if stnum == `st' & year>1988 &

 year<2003

 local dems = r(sum)

 local total = `reps' + `dems'

 local demp = (`dems'/`total') \* .25

 local total\_percent = `total\_percent' + `demp'

 }

 qui count if demgov == 1 & stnum == `st' & year>1988 &

 year<2003

 local demgs = r(N)

 qui count if stnum == `st' & year>1988 & year<2003

 local totalg = r(N)

 local gov\_per = (`demgs'/`totalg') \* .5

 local total\_percent = `total\_percent' + `gov\_per'

 replace dem\_strength = `total\_percent' if stnum == `st' &

 year>1988 & year<2003

 }

**The following commands were helpful in generating Democratic Control of a state’s government:**

To recode a variable that ranged from 0 to 1 (and assumed any value in between) into just 0 and 1, I typed the following:

recode demcont (.0001/.5 = 0) (.5001/1.0 = 1) Note: the recode command does not recognize “<” and “>.” To create and recode a variable that comprises several variables (e.g., Democratic control includes both houses of the legislature plus the governorship):

gen lhdempc= lhdempro

recode lhdempc (.0000/.5 = 0) (.5001/1 = 1)

gen uhdempc= uhdempro

recode uhdempc (.0000/.5 = 0) (.5001/1 = 1)

gen demcont = lhdempc + uhdempc + demgov

recode demcont (.0001/2.999 = 0) (3 = 1)

**The following commands were to convert sptaxrat from fiscal to calendar years.**

bysort state year: assert \_n==1

by state: gen sptaxratcal = .75\*sptaxrat + .25\*sptaxrat[\_n+1]if \_n < \_N

**The do file below is for creating a margin of victory variable**

**for margins of 3% or less**

gen margin=.

replace margin= abs(diff)

gen margin3less=.

replace margin3less=1 if margin<.03001

replace margin3less=0 if margin>.03

**The do file below is for probit and marginal effects in probit**

version 9

use c:/taxreceiveratiodata

probit ballbddole poole95 stcons4 defcon95p nonwhite rat1995 margin95 appro95 ///

 fin95 bud95 elder95 medinc95

dprobit ballbddole poole95 stcons4 defcon95p nonwhite rat1995 margin95 appro95 ///

 fin95 bud95 elder95 medinc95

probit ballbddole poole95 stcons4 defcon95p nonwhite rat1995 margin95 appro95int ///

 fin95int bud95int elder95 medinc95

dprobit ballbddole poole95 stcons4 defcon95p nonwhite rat1995 margin95 appro95int ///

 fin95int bud95int elder95 medinc95

**The do file below is panel corrected standard errors as well as random**

 **effects and fixed effects models**

version 9

use c:taxreceiveratiotime

tsset state year, yearly

xtpcse sptaxrat xsptaxrat pover elderi whitei nameri defconp dhousecc dsenc ///

 dhouseccc dsenccc dhouseccp dsenccp pres reppvr

xtregar sptaxrat pover elderi whitei nameri defconp dhousecc dsenc ///

 dhouseccc dsenccc dhouseccp dsenccp pres reppvr, re

xtregar sptaxrat xsptaxrat pover elderi whitei nameri defconp dhousecc dsenc ///

 dhouseccc dsenccc dhouseccp dsenccp pres reppvr, fe

**The do file below is for a fixed effects model with panel corrected**

**standard errors (no “automatic command” so 49 state dummy variables) and autocorrelation adjustment**

version 9

use c:/taxreceiveratiotime

tsset state year, yearly

xtpcse sptaxrat xsptaxrat pover elderi whitei nameri defconp dhousecc dsenc ///

dhouseccc dsenccc dhouseccp dsenccp pres repv child5t17 margin3less ///

 zala zalaska zariz zark zcal zcolo zconn zdel zflor ///

zgeor zhaw zidaho zillin zindiana ziowa zkent zkan zmaine zlouis zmary ///

 zmass zmich zminn zmiss zmissouri zmont zneb znev znhamp zjersey ///

 zmexico znyork zncar zndak zohio zok zore zpenn zrhode zscar ///

 zsdak ztenn ztex zutah zvmont zvirg zwash zwvirg zwis

xtpcse sptaxrat xsptaxrat pover elderi whitei nameri defconp dhousecc

 dsenc ///

dhouseccc dsenccc dhouseccp dsenccp pres repv child5t17 margin3less ///

 zala zalaska zariz zark zcal zcolo zconn zdel zflor ///

zgeor zhaw zidaho zillin zindiana ziowa zkent zkan zmaine zlouis zmary ///

 zmass zmich zminn zmiss zmissouri zmont zneb znev znhamp zjersey ///

 zmexico znyork zncar zndak zohio zok zore zpenn zrhode zscar ///

 zsdak ztenn ztex zutah zvmont zvirg zwash zwvirg zwis,

 correlation(ar1)

xtpcse sptaxrat xsptaxrat pover elderi whitei nameri defconp dhousecc

 dsenc ///

dhouseccc dsenccc dhouseccp dsenccp pres repv child5t17 margin3less ///

 zala zalaska zariz zark zcal zcolo zconn zdel zflor ///

 zgeor zhaw zidaho zillin zindiana ziowa zkent zkan zmaine zlouis

 zmary ///

 zmass zmich zminn zmiss zmissouri zmont zneb znev znhamp zjersey ///

 zmexico znyork zncar zndak zohio zok zore zpenn zrhode zscar ///

zsdak ztenn ztex zutah zvmont zvirg zwash zwvirg zwis, correlation(ar1) rhotype(tscorr)

**The following commands were both to go from calendar to fiscal years**

**and to collapse annual data into four year periods (i.e., presidential administrations) from 1981 through 2004**

set more off

version 9

use c:/TaxReceiveRatioTime, clear

tsset state year, yearly

bysort state year: assert \_n==1

by state: gen sptaxratcal= .75\*sptaxrat+.25\*(sptaxrat[\_n+1]) if \_n <\_N

by state: gen lagyear=year if \_n==1

assert lagyear==1981|lagyear==.

assert year>=1981

by state: replace lagyear= year if mod(year-1981,4)==0

by state: replace lagyear= (lagyear[\_n-1]) if lagyear==.

tab lagyear

bysort lagyear: sum year

bysort state lagyear year: assert \_n==1

local vlist ""

//convert string variables to numeric

foreach var of varlist medinc1 medinc thou {

replace `var'=subinstr(`var',",","",.)

destring `var', replace

}

foreach var of varlist state-lagyear{

local vlist "`vlist' `var'"

}

local vlist: subinstr local vlist "state" "", all word

local vlist: subinstr local vlist "lagyear" "", all word

di "`vlist'"

collapse `vlist', by(state lagyear)

bysort state lagyear: assert \_n==1

 **Converting the variable “stideoan” (state ideology) from annual data to average scores for presidential administrations over the 1973-2008 period and having the average score read for each year (i.e., the average of 1973, 1974, 1975 and 1976 read for each of those same four years) type:**

use c:/StataTechnicalSupportVersion, clear

set more off

gen term=0

forvalues i = 1973(4)2008 {

replace term=1 if year==`i'

}

bysort stnum: replace term=sum(term)

bysort stnum term: egen term\_avg=mean(stideoan)

set more on

exit

The variable “term\_avg” was what you were after.

**“DO” File: Partisan Strength Variables:**

**Creating Alt/Lowry’s Political Control Variables (Demcont = 1 = Democratic governor plus Democratic control of both houses of the legislature, 0 = other; Repcont = 1 = Republican governor plus Republican control of both houses of the legislature, other = 0; DS = 1 = Democratic governor and Split Legislature – one house majority Democratic and the other majority Republican, 0 = other; DR = 1= Democratic Governor and Republican control of both houses of the legislature, other = 0; RS = 1 = Republican governor and one house majority Republican and the other house majority Democratic, other = 0; and RD = 1 = Republican governor and Democratic majorities in both houses of the legislature, other = 0).**

use "F:\statedata8.dta", clear

tsset stnum year, yearly

//The following commands generate Alt/Lowry Political Variables and Unified //Partisan Control Variables at the State Level

drop demcont repcont ds dr rs rd

drop lhdemp uhdemp lhdemc uhdemc lhrepp uhrepp lhrepc uhrepc

gen lhtot = lhdem + lhrep

gen lhdemp = lhdem/lhtot

gen lhrepp= lhrep/lhtot

gen uhtot = uhdem + uhrep

gen uhdemp = uhdem/uhtot

gen uhrepp= uhrep/uhtot

gen lhdemc= lhdemp

recode lhdemc (.0000/.5 = 0) (.5001/1 = 1)

gen uhdemc= uhdemp

recode uhdemc (.0000/.5 = 0) (.5001/1 = 1)

gen demcont = lhdemc + uhdemc + demgov

recode demcont (.0001/2.999 = 0) (3 = 1)

gen lhrepc= lhrepp

recode lhrepc (.0000/.5 = 0) (.5001/1 = 1)

gen uhrepc= uhrepp

recode uhrepc (.0000/.5 = 0) (.5001/1 = 1)

gen repcont = lhrepc + uhrepc + repgov

recode repcont (.0001/2.999 = 0) (3 = 1)

gen demgov1=demgov

recode demgov1 (1=3)

gen lhdemc1=lhdemc

recode lhdemc1 (1=2)

gen ds=demgov1 + lhdemc1 + uhdemc

recode ds (5=1) (4=1) (0/3=0) (6=0)

gen dr=demgov1 + lhrepc + uhrepc

recode dr (5=1) (else=0)

gen repgov1= repgov

recode repgov1 (1=3)

gen lhrepc1=lhrepc

recode lhrepc1 (1=2)

gen rs=repgov1 + lhrepc1 + uhrepc

recode rs (5=1) (4=1) (0/3=0) (6=0)

gen rd=repgov1 + lhdemc + uhdemc

recode rd (5=1) (else=0)

drop demgov1 repgov1 lhdemc1 lhrepc1 lhtot uhtot

//The following commands generate Alt/Lowry Political Variables and Unified //Partisan Control Variables at the Federal Level

//Variable Names are same as at state level except the first letter if "f" (federal - //fdr is the federal equivalent of dr)

//In the case of unified partisan control the federal variables have an "n" on the //end (national - demcontn is the

//federal equivalent of demcont)

drop demcontn repcontn

drop fds fdr frs frd

drop flhdemp fuhdemp flhdemc fuhdemc pres1

drop flhrepp fuhrepp flhrepc fuhrepc

gen flhtot = demh + reph

gen flhdemp = demh/flhtot

gen flhrepp = reph/flhtot

gen fuhtot = dems + reps

gen fuhdemp = dems/fuhtot

gen fuhrepp= reps/fuhtot

gen flhdemc = flhdemp

recode flhdemc (.0000/.5 = 0) (.5001/1 = 1)

gen fuhdemc= fuhdemp

recode fuhdemc (.0000/.5 = 0) (.5001/1 = 1)

gen demcontn = flhdemc + fuhdemc + pres

recode demcontn (.0001/2.999 = 0) (3 = 1)

gen flhrepc = flhrepp

recode flhrepc (.0000/.5 = 0) (.5001/1 = 1)

gen fuhrepc= fuhrepp

recode fuhrepc (.0000/.5 = 0) (.5001/1 = 1)

gen pres1 = pres

recode pres1 (0 = 1) (1 = 0)

gen repcontn = flhrepc + fuhrepc + pres1

recode repcontn (.0001/2.999 = 0) (3 = 1)

gen pres2 = pres

recode pres2 (1=3)

gen flhdemc1= flhdemc

recode flhdemc1 (1=2)

gen fds= pres2 + flhdemc1 + fuhdemc

recode fds (5=1) (4=1) (0/3=0) (6=0)

gen fdr= pres2 + flhrepc + fuhrepc

recode fdr (5=1) (else=0)

gen pres3 = pres1

recode pres3 (1=3)

gen flhrepc1=flhrepc

recode flhrepc1 (1=2)

gen frs=pres3 + flhrepc1 + fuhrepc

recode frs (5=1) (4=1) (0/3=0) (6=0)

gen frd=pres3 + flhdemc + fuhdemc

recode frd (5=1) (else=0)

drop pres2 pres3 flhtot fuhtot flhdemc1 flhrepc1

**The following were useful obtaining the proportion of income going to particular income groups**

use "G:/statedata.dta", clear

tsset stnum year, yearly

drop q1cper q2cper q3cper q4cper q5cper ratq5q1c

gen q1cper1 = q1clow/5

gen q1cper = q1cper1/meanic

gen q2cper1 = q2c/5

gen q2cper = q2cper1/meanic

gen q3cper1 = q3c/5

gen q3cper = q3cper1/meanic

gen q4cper1 = q4c/5

gen q4cper = q4cper1/meanic

gen q5cper1 = q5chigh/5

gen q5cper = q5cper1/meanic

drop q1cper1 q2cper1 q3cper1 q4cper1 q5cper1

gen ratq5q1c = q5cper/q1cper

**The following commands were used to generate a neighboring states variable.**

\*\*In this file, NeigboringStatesUsable2, the data are stacked by state and year and year (i.e., obs. #2 is state #1 in year 2)

\*\* where as in NeighboringStatesUsable3 the data are stacked by year and state (i.e., obs. #2 is state #2 in year 1)

\*Code altered by Carl Klarner on June 6, 2016, but code originally from Berry and Berry (1990) and Frederick Boehmke.

\*For STATA 10.

\*NOTE: The setup below is to generate 1 neighboring state variable. In this case the created variable is rtwpolneighave2. You can create 1 new

\*neighboring states variable and save the dataset. To create a second neighboring states variable simply substitute the variable you want to

\*work with for rtwpol (I believe you only need to change line 111 below) in the commands below save the new do file. The new neighboring states variable will be what ever your variable name is

\*now ending with neighave2. The neighboring states variable is each states policy score times it's proportion of the total population of the

\*neighboring states. The score must be no lower than 0 and no higher than 1.

\*\*FURTHER NOTE: \*The list below of bordering states is the state's number plus the number in the brackets. For example, for state #1 (Alabama)

\*the first border state is listed as \_n + 8. Since Alabama is state #1 \_n becomes 1, thus, 1 + 8 = 9 and means that state #9 (Florida) is one of the states bordering Alabama. Pay attention

\*to the sign after "n". For example, notice for state #5 (California) the first bordering state is \_n-02. So, 5 - 2 = 3 and means that

\*state #3 (Arizona) is one of the states bordering California. I've checked and the alphabetical order of states for Klarner's "stateno"

\*variable is the same order we use. The list of bordering states below is very similar, but not identical, to our original list. For example,

\*Klarner's list doesn't have Alaska or Hawaii bordering any states where we, originally, had Hawaii bordering California. Klarner's list had

\*Kentucky bordering Arkansas (state #4) which neither us, or sources I could find online, had. In the following cases we deleted one state

\*from Boehmke's list: state #17 Kentucky (deleted Arkansas), state #19 Maine (deleted Massachusetts), state #20 Maryland (deleted New Jersey),

\*state 21 Massachusetts (deleted Maine) and state #30 New Jersey (deleted Maryland).

\*Original information about file.

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* File: nbrs-code.do \*/

/\* Author: Frederick J. Boehmke \*/

/\* Date: August 25, 2001 \*/

/\* Purpose: STATA do-file to be called by some other do-file (or manually). \*/

/\* \*/

/\* --IMPORTANT-- \*/

/\* \*/

/\* Must be sorted by stnum (which is very specific - based on sort by full \*/

/\* state name, NOT abbreviation). All 50 states MUST be included! \*/

/\* Coding taking from Berry and Berry's 1990 AJPS article. \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

\*This code creates four groups of variables.

\*Group #1

\*This code constructs a set of variables \*neighsum, \*neighsum, etc., each of which sums the value of a variable

\*"\*" you specify, in the states next to the state a row in your dataset represents. The "\*" represents the name of the variable

\*in your dataset that you will specify.

\*Group #2

\*Second, it constructs a set of variables \*neighave1, \*neighave1, etc., each of which average the value of a variable

\*"\*" you specify in the states next to the state a row in your dataset represents.

\*Group #3

\*Third, it constructs a set of variables \*neighave2, \*neighave2, etc., each of which take the weighted average of the value of a variable

\*"\*" you specify in the states next to the state a row in your dataset represents. The weight is determined by state population.

\*An accompanying file provides state population.

\*Group #4

\*Fourth, you may consider an interaction between the ratio of the population of the state in question, with the population of its neighbors.

\*For example, Delaware is much smaller than all its neighbors combined (MD, NJ and PA)

\*while California has around three times the combined population of its neighbors (AZ, NV and OR).

\*This is a distinct issue from the issue of weighting the average of neighbor state values by neighbors' populations (i.e., group #2 v group #3).

\*The variable "popratio" below computes the ratio of the population of an observation's state, to its combined neighbors' populations, and a fourth set

\*of variables computed here interact the variable in question with "popratio" and are called \*neighave2int, \*neighave2int, etc.

\*Keep in mind that STATA limits the length of variable names to 32 characters, so the variables you're feeding in can't have more than 20 characters

\*in their names.

\*IMPORTANT

\*WHEREEVER "yyy" APPEARS IN THE CODE BELOW, YOU HAVE TO INPUT SOME CHARACTER OR NUMBER, DESCRIBED ON THE LINE IMMEDIATELY ABOVE.

\*If "yyy" is left in the code anywhere, the code will not run all the way through.

\*There are ten places below you have to input values, including opening your dataset and saving it.

\*The instructions are precise and require information that is easy to provide, so you should be able to complete

\*you work with this file in under ten minutes.

\*You must have a state identifier that represents the number of a state in an alphabetical list.

version 10

use "F:\statedata8.dta", clear

gen stateno=stnum

\*Open your dataset here.

\*Appearance 1 of 10 of yyy.

\*First, rename your state identifier and year variables to the following names.

\*As indicated above, your state identifier should represent the number of a state in an alphabetical list.

\*Put the name of your state identifier where "yyy" is below.

\*If your state identifier is already called "stateno" and represents the number of the state in an alphabetical list,

\*then don't run the following line of code.

\*Appearance 2 of 10 of yyy.

\*Put the name of your year variable where "yyy" is below.

\*If your year identifier is already called "year" then don't run the following line of code.

\*Appearance 3 of 10 of yyy.

\*gen year=yyy

\*The following specifies how many neighbor states each state has.

recode stateno (1=4) (2=0) (3=5) (4=6) (5=3) (6=7) (7=3) (8=3) (9=2) (10=5) ///

(11=0) (12=6) (13=6) (14=4) (15=6) (16=4) (17=7) (18=3) (19=1) (20=4) ///

(21=5) (22=4) (23=5) (24=4) (25=8) (26=4) (27=6) (28=5) (29=3) (30=3) ///

(31=5) (32=5) (33=4) (34=3) (35=5) (36=6) (37=4) (38=6) (39=2) (40=2) ///

(41=6) (42=8) (43=4) (44=6) (45=3) (46=5) (47=2) (48=5) (49=4) (50=6) ///

, gen(numneighs)

sort year stateno

\*The following specifies the list of variables you wish to do the computation on.

\*Put the variable names in place of "yyy" below that you wish to do the computations on, one after the other.

\*Delete the third row below if you're only going to do this on two variables, or the second and third rows

\*below if you're only going to do this on one variable.

\*Appearance 5 of 10 of yyy.

local aaa1 rtwpol

\*Just copy and paste the above to create more lines with aaa4, aaa5, etc., if you want to do

\*the computations on four or more variables.

\*Group #1

\*The following sums the values of neighbor states' values.

\*Change "yyy" in the following line to reflect the number of variables you are going to produce the computations for.

\*If you are only going to do this for one variable, put in the number "1."

\*Appearance 6 of 10 of yyy.

forvalues bbb=1/1 {

gen `aaa`bbb''neighsum=.

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+8]+`aaa`bbb''[\_n+9]+`aaa`bbb''[\_n+23]+`aaa`bbb''[\_n+41] if stateno==1

replace `aaa`bbb''neighsum = 0 if stateno==2

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+03]+`aaa`bbb''[\_n+02]+`aaa`bbb''[\_n+28]+`aaa`bbb''[\_n+25]+`aaa`bbb''[\_n+41] if stateno==3

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+14]+`aaa`bbb''[\_n+39]+`aaa`bbb''[\_n+32]+`aaa`bbb''[\_n+21]+`aaa`bbb''[\_n+38]+`aaa`bbb''[\_n+20] if stateno==4

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-02]+`aaa`bbb''[\_n+23]+`aaa`bbb''[\_n+32] if stateno==5

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-03]+`aaa`bbb''[\_n+25]+`aaa`bbb''[\_n+38]+`aaa`bbb''[\_n+44]+`aaa`bbb''[\_n+10]+`aaa`bbb''[\_n+21]+`aaa`bbb''[\_n+30] if stateno==6

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+14]+`aaa`bbb''[\_n+25]+`aaa`bbb''[\_n+32] if stateno==7

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+12]+`aaa`bbb''[\_n+30]+`aaa`bbb''[\_n+22] if stateno==8

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-08]+`aaa`bbb''[\_n+01] if stateno==9

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-01]+`aaa`bbb''[\_n-09]+`aaa`bbb''[\_n+32]+`aaa`bbb''[\_n+30]+`aaa`bbb''[\_n+23] if stateno==10

replace `aaa`bbb''neighsum = 0 if stateno==11

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+35]+`aaa`bbb''[\_n+25]+`aaa`bbb''[\_n+16]+`aaa`bbb''[\_n+32]+`aaa`bbb''[\_n+38]+`aaa`bbb''[\_n+14] if stateno==12

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+36]+`aaa`bbb''[\_n+02]+`aaa`bbb''[\_n+12]+`aaa`bbb''[\_n+04]+`aaa`bbb''[\_n+01]+`aaa`bbb''[\_n+09] if stateno==13

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+03]+`aaa`bbb''[\_n-01]+`aaa`bbb''[\_n+08]+`aaa`bbb''[\_n+21] if stateno==14

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+10]+`aaa`bbb''[\_n+12]+`aaa`bbb''[\_n+26]+`aaa`bbb''[\_n+08]+`aaa`bbb''[\_n+34]+`aaa`bbb''[\_n-02] if stateno==15

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+20]+`aaa`bbb''[\_n-10]+`aaa`bbb''[\_n+11]+`aaa`bbb''[\_n+09] if stateno==16

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+25]+`aaa`bbb''[\_n-04]+`aaa`bbb''[\_n-03]+`aaa`bbb''[\_n+08]+`aaa`bbb''[\_n+18]+`aaa`bbb''[\_n+29]+`aaa`bbb''[\_n+31] if stateno==17

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+25]+`aaa`bbb''[\_n-14]+`aaa`bbb''[\_n+06] if stateno==18

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+10] if stateno==19

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+26]+`aaa`bbb''[\_n+28]+`aaa`bbb''[\_n+18]+`aaa`bbb''[\_n-12] if stateno==20

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+18]+`aaa`bbb''[\_n-14]+`aaa`bbb''[\_n+11]+`aaa`bbb''[\_n+08]+`aaa`bbb''[\_n+24] if stateno==21

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+27]+`aaa`bbb''[\_n+01]+`aaa`bbb''[\_n-08]+`aaa`bbb''[\_n-13] if stateno==22

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+11]+`aaa`bbb''[\_n+18]+`aaa`bbb''[\_n-08]+`aaa`bbb''[\_n+26]+`aaa`bbb''[\_n-01] if stateno==23

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-06]+`aaa`bbb''[\_n-23]+`aaa`bbb''[\_n+18]+`aaa`bbb''[\_n-20] if stateno==24

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-21]+`aaa`bbb''[\_n+11]+`aaa`bbb''[\_n-09]+`aaa`bbb''[\_n+02]+`aaa`bbb''[\_n-10]+`aaa`bbb''[\_n-12]+`aaa`bbb''[\_n-08]+`aaa`bbb''[\_n+17] if stateno==25

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-14]+`aaa`bbb''[\_n+24]+`aaa`bbb''[\_n+15]+`aaa`bbb''[\_n+08] if stateno==26

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-11]+`aaa`bbb''[\_n-21]+`aaa`bbb''[\_n+23]+`aaa`bbb''[\_n+14]+`aaa`bbb''[\_n-12]+`aaa`bbb''[\_n-02] if stateno==27

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-23]+`aaa`bbb''[\_n+09]+`aaa`bbb''[\_n-16]+`aaa`bbb''[\_n+16]+`aaa`bbb''[\_n-25] if stateno==28

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-08]+`aaa`bbb''[\_n+16]+`aaa`bbb''[\_n-10] if stateno==29

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-22]+`aaa`bbb''[\_n+08]+`aaa`bbb''[\_n+02] if stateno==30

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-28]+`aaa`bbb''[\_n+13]+`aaa`bbb''[\_n-25]+`aaa`bbb''[\_n+05]+`aaa`bbb''[\_n+12] if stateno==31

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+06]+`aaa`bbb''[\_n-02]+`aaa`bbb''[\_n-25]+`aaa`bbb''[\_n-11]+`aaa`bbb''[\_n+13] if stateno==32

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+07]+`aaa`bbb''[\_n-23]+`aaa`bbb''[\_n+09]+`aaa`bbb''[\_n+13] if stateno==33

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+07]+`aaa`bbb''[\_n-08]+`aaa`bbb''[\_n-11] if stateno==34

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-18]+`aaa`bbb''[\_n-21]+`aaa`bbb''[\_n-13]+`aaa`bbb''[\_n+03]+`aaa`bbb''[\_n+13] if stateno==35

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n+07]+`aaa`bbb''[\_n-05]+`aaa`bbb''[\_n-30]+`aaa`bbb''[\_n-20]+`aaa`bbb''[\_n-11]+`aaa`bbb''[\_n-32] if stateno==36

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-32]+`aaa`bbb''[\_n-09]+`aaa`bbb''[\_n-25]+`aaa`bbb''[\_n+10] if stateno==37

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-30]+`aaa`bbb''[\_n-18]+`aaa`bbb''[\_n+10]+`aaa`bbb''[\_n-03]+`aaa`bbb''[\_n-06]+`aaa`bbb''[\_n-08] if stateno==38

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-32]+`aaa`bbb''[\_n-18] if stateno==39

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-30]+`aaa`bbb''[\_n-07] if stateno==40

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-07]+`aaa`bbb''[\_n-14]+`aaa`bbb''[\_n+09]+`aaa`bbb''[\_n-15]+`aaa`bbb''[\_n-18]+`aaa`bbb''[\_n-26] if stateno==41

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-09]+`aaa`bbb''[\_n-32]+`aaa`bbb''[\_n-41]+`aaa`bbb''[\_n-18]+`aaa`bbb''[\_n-38]+`aaa`bbb''[\_n-17]+`aaa`bbb''[\_n-25]+`aaa`bbb''[\_n+04] if stateno==42

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-12]+`aaa`bbb''[\_n-07]+`aaa`bbb''[\_n-39]+`aaa`bbb''[\_n-25] if stateno==43

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-41]+`aaa`bbb''[\_n-16]+`aaa`bbb''[\_n-32]+`aaa`bbb''[\_n+06]+`aaa`bbb''[\_n-38]+`aaa`bbb''[\_n-13] if stateno==44

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-16]+`aaa`bbb''[\_n-24]+`aaa`bbb''[\_n-13] if stateno==45

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-13]+`aaa`bbb''[\_n-04]+`aaa`bbb''[\_n-29]+`aaa`bbb''[\_n+02]+`aaa`bbb''[\_n-26] if stateno==46

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-10]+`aaa`bbb''[\_n-35] if stateno==47

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-02]+`aaa`bbb''[\_n-31]+`aaa`bbb''[\_n-13]+`aaa`bbb''[\_n-10]+`aaa`bbb''[\_n-28] if stateno==48

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-26]+`aaa`bbb''[\_n-34]+`aaa`bbb''[\_n-36]+`aaa`bbb''[\_n-27] if stateno==49

replace `aaa`bbb''neighsum = `aaa`bbb''[\_n-44]+`aaa`bbb''[\_n-06]+`aaa`bbb''[\_n-38]+`aaa`bbb''[\_n-24]+`aaa`bbb''[\_n-09]+`aaa`bbb''[\_n-23] if stateno==50

}

\*

\*Group #2

\*The following averages the values of neighbor states' values.

\*Change "yyy" in the following line to reflect the number of variables you are going to produce the computations for.

\*This is the same number you put in above for group #1.

\*If you are only going to do this for one variable, put in the number "1."

\*Appearance 7 of 10 of yyy.

forvalues bbb=1/1 {

gen `aaa`bbb''neighave1=.

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+8]+`aaa`bbb''[\_n+9]+`aaa`bbb''[\_n+23]+`aaa`bbb''[\_n+41] ) / numneighs if stateno==1

replace `aaa`bbb''neighave1 = (0 ) / numneighs if stateno==2

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+03]+`aaa`bbb''[\_n+02]+`aaa`bbb''[\_n+28]+`aaa`bbb''[\_n+25]+`aaa`bbb''[\_n+41] ) / numneighs if stateno==3

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+14]+`aaa`bbb''[\_n+39]+`aaa`bbb''[\_n+32]+`aaa`bbb''[\_n+21]+`aaa`bbb''[\_n+38]+`aaa`bbb''[\_n+20] ) / numneighs if stateno==4

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-02]+`aaa`bbb''[\_n+23]+`aaa`bbb''[\_n+32] ) / numneighs if stateno==5

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-03]+`aaa`bbb''[\_n+25]+`aaa`bbb''[\_n+38]+`aaa`bbb''[\_n+44]+`aaa`bbb''[\_n+10]+`aaa`bbb''[\_n+21]+`aaa`bbb''[\_n+30] ) / numneighs if stateno==6

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+14]+`aaa`bbb''[\_n+25]+`aaa`bbb''[\_n+32] ) / numneighs if stateno==7

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+12]+`aaa`bbb''[\_n+30]+`aaa`bbb''[\_n+22] ) / numneighs if stateno==8

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-08]+`aaa`bbb''[\_n+01] ) / numneighs if stateno==9

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-01]+`aaa`bbb''[\_n-09]+`aaa`bbb''[\_n+32]+`aaa`bbb''[\_n+30]+`aaa`bbb''[\_n+23] ) / numneighs if stateno==10

replace `aaa`bbb''neighave1 = (0 ) / numneighs if stateno==11

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+35]+`aaa`bbb''[\_n+25]+`aaa`bbb''[\_n+16]+`aaa`bbb''[\_n+32]+`aaa`bbb''[\_n+38]+`aaa`bbb''[\_n+14] ) / numneighs if stateno==12

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+36]+`aaa`bbb''[\_n+02]+`aaa`bbb''[\_n+12]+`aaa`bbb''[\_n+04]+`aaa`bbb''[\_n+01]+`aaa`bbb''[\_n+09] ) / numneighs if stateno==13

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+03]+`aaa`bbb''[\_n-01]+`aaa`bbb''[\_n+08]+`aaa`bbb''[\_n+21] ) / numneighs if stateno==14

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+10]+`aaa`bbb''[\_n+12]+`aaa`bbb''[\_n+26]+`aaa`bbb''[\_n+08]+`aaa`bbb''[\_n+34]+`aaa`bbb''[\_n-02] ) / numneighs if stateno==15

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+20]+`aaa`bbb''[\_n-10]+`aaa`bbb''[\_n+11]+`aaa`bbb''[\_n+09] ) / numneighs if stateno==16

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+25]+`aaa`bbb''[\_n-04]+`aaa`bbb''[\_n-03]+`aaa`bbb''[\_n+08]+`aaa`bbb''[\_n+18]+`aaa`bbb''[\_n+29]+`aaa`bbb''[\_n+31] ) / numneighs if stateno==17

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+25]+`aaa`bbb''[\_n-14]+`aaa`bbb''[\_n+06] ) / numneighs if stateno==18

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+10]) / numneighs if stateno==19

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+26]+`aaa`bbb''[\_n+28]+`aaa`bbb''[\_n+18]+`aaa`bbb''[\_n-12] ) / numneighs if stateno==20

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+18]+`aaa`bbb''[\_n-14]+`aaa`bbb''[\_n+11]+`aaa`bbb''[\_n+08]+`aaa`bbb''[\_n+24] ) / numneighs if stateno==21

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+27]+`aaa`bbb''[\_n+01]+`aaa`bbb''[\_n-08]+`aaa`bbb''[\_n-13] ) / numneighs if stateno==22

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+11]+`aaa`bbb''[\_n+18]+`aaa`bbb''[\_n-08]+`aaa`bbb''[\_n+26]+`aaa`bbb''[\_n-01] ) / numneighs if stateno==23

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-06]+`aaa`bbb''[\_n-23]+`aaa`bbb''[\_n+18]+`aaa`bbb''[\_n-20] ) / numneighs if stateno==24

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-21]+`aaa`bbb''[\_n+11]+`aaa`bbb''[\_n-09]+`aaa`bbb''[\_n+02]+`aaa`bbb''[\_n-10]+`aaa`bbb''[\_n-12]+`aaa`bbb''[\_n-08]+`aaa`bbb''[\_n+17] ) / numneighs if stateno==25

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-14]+`aaa`bbb''[\_n+24]+`aaa`bbb''[\_n+15]+`aaa`bbb''[\_n+08] ) / numneighs if stateno==26

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-11]+`aaa`bbb''[\_n-21]+`aaa`bbb''[\_n+23]+`aaa`bbb''[\_n+14]+`aaa`bbb''[\_n-12]+`aaa`bbb''[\_n-02] ) / numneighs if stateno==27

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-23]+`aaa`bbb''[\_n+09]+`aaa`bbb''[\_n-16]+`aaa`bbb''[\_n+16]+`aaa`bbb''[\_n-25] ) / numneighs if stateno==28

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-08]+`aaa`bbb''[\_n+16]+`aaa`bbb''[\_n-10] ) / numneighs if stateno==29

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-22]+`aaa`bbb''[\_n+08]+`aaa`bbb''[\_n+02] ) / numneighs if stateno==30

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-28]+`aaa`bbb''[\_n+13]+`aaa`bbb''[\_n-25]+`aaa`bbb''[\_n+05]+`aaa`bbb''[\_n+12] ) / numneighs if stateno==31

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+06]+`aaa`bbb''[\_n-02]+`aaa`bbb''[\_n-25]+`aaa`bbb''[\_n-11]+`aaa`bbb''[\_n+13] ) / numneighs if stateno==32

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+07]+`aaa`bbb''[\_n-23]+`aaa`bbb''[\_n+09]+`aaa`bbb''[\_n+13] ) / numneighs if stateno==33

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+07]+`aaa`bbb''[\_n-08]+`aaa`bbb''[\_n-11] ) / numneighs if stateno==34

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-18]+`aaa`bbb''[\_n-21]+`aaa`bbb''[\_n-13]+`aaa`bbb''[\_n+03]+`aaa`bbb''[\_n+13] ) / numneighs if stateno==35

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n+07]+`aaa`bbb''[\_n-05]+`aaa`bbb''[\_n-30]+`aaa`bbb''[\_n-20]+`aaa`bbb''[\_n-11]+`aaa`bbb''[\_n-32] ) / numneighs if stateno==36

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-32]+`aaa`bbb''[\_n-09]+`aaa`bbb''[\_n-25]+`aaa`bbb''[\_n+10] ) / numneighs if stateno==37

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-30]+`aaa`bbb''[\_n-18]+`aaa`bbb''[\_n+10]+`aaa`bbb''[\_n-03]+`aaa`bbb''[\_n-06]+`aaa`bbb''[\_n-08] ) / numneighs if stateno==38

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-32]+`aaa`bbb''[\_n-18] ) / numneighs if stateno==39

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-30]+`aaa`bbb''[\_n-07] ) / numneighs if stateno==40

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-07]+`aaa`bbb''[\_n-14]+`aaa`bbb''[\_n+09]+`aaa`bbb''[\_n-15]+`aaa`bbb''[\_n-18]+`aaa`bbb''[\_n-26] ) / numneighs if stateno==41

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-09]+`aaa`bbb''[\_n-32]+`aaa`bbb''[\_n-41]+`aaa`bbb''[\_n-18]+`aaa`bbb''[\_n-38]+`aaa`bbb''[\_n-17]+`aaa`bbb''[\_n-25]+`aaa`bbb''[\_n+04] ) / numneighs if stateno==42

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-12]+`aaa`bbb''[\_n-07]+`aaa`bbb''[\_n-39]+`aaa`bbb''[\_n-25] ) / numneighs if stateno==43

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-41]+`aaa`bbb''[\_n-16]+`aaa`bbb''[\_n-32]+`aaa`bbb''[\_n+06]+`aaa`bbb''[\_n-38]+`aaa`bbb''[\_n-13] ) / numneighs if stateno==44

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-16]+`aaa`bbb''[\_n-24]+`aaa`bbb''[\_n-13] ) / numneighs if stateno==45

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-13]+`aaa`bbb''[\_n-04]+`aaa`bbb''[\_n-29]+`aaa`bbb''[\_n+02]+`aaa`bbb''[\_n-26] ) / numneighs if stateno==46

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-10]+`aaa`bbb''[\_n-35] ) / numneighs if stateno==47

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-02]+`aaa`bbb''[\_n-31]+`aaa`bbb''[\_n-13]+`aaa`bbb''[\_n-10]+`aaa`bbb''[\_n-28] ) / numneighs if stateno==48

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-26]+`aaa`bbb''[\_n-34]+`aaa`bbb''[\_n-36]+`aaa`bbb''[\_n-27] ) / numneighs if stateno==49

replace `aaa`bbb''neighave1 = (`aaa`bbb''[\_n-44]+`aaa`bbb''[\_n-06]+`aaa`bbb''[\_n-38]+`aaa`bbb''[\_n-24]+`aaa`bbb''[\_n-09]+`aaa`bbb''[\_n-23] ) / numneighs if stateno==50

}

\*

\*Group #3

\*The following takes the weighted average of neighbor states' values, weighting by population.

\*First, the total population of all neighbor states is computed.

gen popneighsum=.

replace popneighsum = pop[\_n+8]+pop[\_n+9]+pop[\_n+23]+pop[\_n+41] if stateno==1

replace popneighsum = 0 if stateno==2

replace popneighsum = pop[\_n+03]+pop[\_n+02]+pop[\_n+28]+pop[\_n+25]+pop[\_n+41] if stateno==3

replace popneighsum = pop[\_n+14]+pop[\_n+39]+pop[\_n+32]+pop[\_n+21]+pop[\_n+38]+pop[\_n+20] if stateno==4

replace popneighsum = pop[\_n-02]+pop[\_n+23]+pop[\_n+32] if stateno==5

replace popneighsum = pop[\_n-03]+pop[\_n+25]+pop[\_n+38]+pop[\_n+44]+pop[\_n+10]+pop[\_n+21]+pop[\_n+30] if stateno==6

replace popneighsum = pop[\_n+14]+pop[\_n+25]+pop[\_n+32] if stateno==7

replace popneighsum = pop[\_n+12]+pop[\_n+30]+pop[\_n+22] if stateno==8

replace popneighsum = pop[\_n-08]+pop[\_n+01] if stateno==9

replace popneighsum = pop[\_n-01]+pop[\_n-09]+pop[\_n+32]+pop[\_n+30]+pop[\_n+23] if stateno==10

replace popneighsum = 0 if stateno==11

replace popneighsum = pop[\_n+35]+pop[\_n+25]+pop[\_n+16]+pop[\_n+32]+pop[\_n+38]+pop[\_n+14] if stateno==12

replace popneighsum = pop[\_n+36]+pop[\_n+02]+pop[\_n+12]+pop[\_n+04]+pop[\_n+01]+pop[\_n+09] if stateno==13

replace popneighsum = pop[\_n+03]+pop[\_n-01]+pop[\_n+08]+pop[\_n+21] if stateno==14

replace popneighsum = pop[\_n+10]+pop[\_n+12]+pop[\_n+26]+pop[\_n+08]+pop[\_n+34]+pop[\_n-02] if stateno==15

replace popneighsum = pop[\_n+20]+pop[\_n-10]+pop[\_n+11]+pop[\_n+09] if stateno==16

replace popneighsum = pop[\_n+25]+pop[\_n-04]+pop[\_n-03]+pop[\_n+08]+pop[\_n+18]+pop[\_n+29]+pop[\_n+31] if stateno==17

replace popneighsum = pop[\_n+25]+pop[\_n-14]+pop[\_n+06] if stateno==18

replace popneighsum = pop[\_n+10] if stateno==19

replace popneighsum = pop[\_n+26]+pop[\_n+28]+pop[\_n+18]+pop[\_n-12] if stateno==20

replace popneighsum = pop[\_n+18]+pop[\_n-14]+pop[\_n+11]+pop[\_n+08]+pop[\_n+24] if stateno==21

replace popneighsum = pop[\_n+27]+pop[\_n+01]+pop[\_n-08]+pop[\_n-13] if stateno==22

replace popneighsum = pop[\_n+11]+pop[\_n+18]+pop[\_n-08]+pop[\_n+26]+pop[\_n-01] if stateno==23

replace popneighsum = pop[\_n-06]+pop[\_n-23]+pop[\_n+18]+pop[\_n-20] if stateno==24

replace popneighsum = pop[\_n-21]+pop[\_n+11]+pop[\_n-09]+pop[\_n+02]+pop[\_n-10]+pop[\_n-12]+pop[\_n-08]+pop[\_n+17] if stateno==25

replace popneighsum = pop[\_n-14]+pop[\_n+24]+pop[\_n+15]+pop[\_n+08] if stateno==26

replace popneighsum = pop[\_n-11]+pop[\_n-21]+pop[\_n+23]+pop[\_n+14]+pop[\_n-12]+pop[\_n-02] if stateno==27

replace popneighsum = pop[\_n-23]+pop[\_n+09]+pop[\_n-16]+pop[\_n+16]+pop[\_n-25] if stateno==28

replace popneighsum = pop[\_n-08]+pop[\_n+16]+pop[\_n-10] if stateno==29

replace popneighsum = pop[\_n-22]+pop[\_n+08]+pop[\_n+02] if stateno==30

replace popneighsum = pop[\_n-28]+pop[\_n+13]+pop[\_n-25]+pop[\_n+05]+pop[\_n+12] if stateno==31

replace popneighsum = pop[\_n+06]+pop[\_n-02]+pop[\_n-25]+pop[\_n-11]+pop[\_n+13] if stateno==32

replace popneighsum = pop[\_n+07]+pop[\_n-23]+pop[\_n+09]+pop[\_n+13] if stateno==33

replace popneighsum = pop[\_n+07]+pop[\_n-08]+pop[\_n-11] if stateno==34

replace popneighsum = pop[\_n-18]+pop[\_n-21]+pop[\_n-13]+pop[\_n+03]+pop[\_n+13] if stateno==35

replace popneighsum = pop[\_n+07]+pop[\_n-05]+pop[\_n-30]+pop[\_n-20]+pop[\_n-11]+pop[\_n-32] if stateno==36

replace popneighsum = pop[\_n-32]+pop[\_n-09]+pop[\_n-25]+pop[\_n+10] if stateno==37

replace popneighsum = pop[\_n-30]+pop[\_n-18]+pop[\_n+10]+pop[\_n-03]+pop[\_n-06]+pop[\_n-08] if stateno==38

replace popneighsum = pop[\_n-32]+pop[\_n-18] if stateno==39

replace popneighsum = pop[\_n-30]+pop[\_n-07] if stateno==40

replace popneighsum = pop[\_n-07]+pop[\_n-14]+pop[\_n+09]+pop[\_n-15]+pop[\_n-18]+pop[\_n-26] if stateno==41

replace popneighsum = pop[\_n-09]+pop[\_n-32]+pop[\_n-41]+pop[\_n-18]+pop[\_n-38]+pop[\_n-17]+pop[\_n-25]+pop[\_n+04] if stateno==42

replace popneighsum = pop[\_n-12]+pop[\_n-07]+pop[\_n-39]+pop[\_n-25] if stateno==43

replace popneighsum = pop[\_n-41]+pop[\_n-16]+pop[\_n-32]+pop[\_n+06]+pop[\_n-38]+pop[\_n-13] if stateno==44

replace popneighsum = pop[\_n-16]+pop[\_n-24]+pop[\_n-13] if stateno==45

replace popneighsum = pop[\_n-13]+pop[\_n-04]+pop[\_n-29]+pop[\_n+02]+pop[\_n-26] if stateno==46

replace popneighsum = pop[\_n-10]+pop[\_n-35] if stateno==47

replace popneighsum = pop[\_n-02]+pop[\_n-31]+pop[\_n-13]+pop[\_n-10]+pop[\_n-28] if stateno==48

replace popneighsum = pop[\_n-26]+pop[\_n-34]+pop[\_n-36]+pop[\_n-27] if stateno==49

replace popneighsum = pop[\_n-44]+pop[\_n-06]+pop[\_n-38]+pop[\_n-24]+pop[\_n-09]+pop[\_n-23] if stateno==50

\*Now the variable of interest for group #3 is computed.

\*Change "yyy" in the following line to reflect the number of variables you are going to produce the computations for.

\*This is the same number you put in above for group #1.

\*If you are only going to do this for one variable, put in the number "1."

\*Appearance 8 of 10 of yyy.

forvalues bbb=1/1 {

gen `aaa`bbb''z=`aaa`bbb''\*pop

gen `aaa`bbb''neighave2=.

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+8]+`aaa`bbb''z[\_n+9]+`aaa`bbb''z[\_n+23]+`aaa`bbb''z[\_n+41] ) / popneighsum if stateno==1

replace `aaa`bbb''neighave2 = ( 0 ) / popneighsum if stateno==2

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+03]+`aaa`bbb''z[\_n+02]+`aaa`bbb''z[\_n+28]+`aaa`bbb''z[\_n+25]+`aaa`bbb''z[\_n+41] ) / popneighsum if stateno==3

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+14]+`aaa`bbb''z[\_n+39]+`aaa`bbb''z[\_n+32]+`aaa`bbb''z[\_n+21]+`aaa`bbb''z[\_n+38]+`aaa`bbb''z[\_n+20] ) / popneighsum if stateno==4

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-02]+`aaa`bbb''z[\_n+23]+`aaa`bbb''z[\_n+32] ) / popneighsum if stateno==5

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-03]+`aaa`bbb''z[\_n+25]+`aaa`bbb''z[\_n+38]+`aaa`bbb''z[\_n+44]+`aaa`bbb''z[\_n+10]+`aaa`bbb''z[\_n+21]+`aaa`bbb''z[\_n+30] ) / popneighsum if stateno==6

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+14]+`aaa`bbb''z[\_n+25]+`aaa`bbb''z[\_n+32] ) / popneighsum if stateno==7

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+12]+`aaa`bbb''z[\_n+30]+`aaa`bbb''z[\_n+22] ) / popneighsum if stateno==8

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-08]+`aaa`bbb''z[\_n+01] ) / popneighsum if stateno==9

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-01]+`aaa`bbb''z[\_n-09]+`aaa`bbb''z[\_n+32]+`aaa`bbb''z[\_n+30]+`aaa`bbb''z[\_n+23] ) / popneighsum if stateno==10

replace `aaa`bbb''neighave2 = ( 0 ) / popneighsum if stateno==11

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+35]+`aaa`bbb''z[\_n+25]+`aaa`bbb''z[\_n+16]+`aaa`bbb''z[\_n+32]+`aaa`bbb''z[\_n+38]+`aaa`bbb''z[\_n+14] ) / popneighsum if stateno==12

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+36]+`aaa`bbb''z[\_n+02]+`aaa`bbb''z[\_n+12]+`aaa`bbb''z[\_n+04]+`aaa`bbb''z[\_n+01]+`aaa`bbb''z[\_n+09] ) / popneighsum if stateno==13

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+03]+`aaa`bbb''z[\_n-01]+`aaa`bbb''z[\_n+08]+`aaa`bbb''z[\_n+21] ) / popneighsum if stateno==14

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+10]+`aaa`bbb''z[\_n+12]+`aaa`bbb''z[\_n+26]+`aaa`bbb''z[\_n+08]+`aaa`bbb''z[\_n+34]+`aaa`bbb''z[\_n-02] ) / popneighsum if stateno==15

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+20]+`aaa`bbb''z[\_n-10]+`aaa`bbb''z[\_n+11]+`aaa`bbb''z[\_n+09] ) / popneighsum if stateno==16

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-13]+`aaa`bbb''z[\_n+25]+`aaa`bbb''z[\_n-04]+`aaa`bbb''z[\_n-03]+`aaa`bbb''z[\_n+08]+`aaa`bbb''z[\_n+18]+`aaa`bbb''z[\_n+29]+`aaa`bbb''z[\_n+31] ) / popneighsum if stateno==17

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+25]+`aaa`bbb''z[\_n-14]+`aaa`bbb''z[\_n+06] ) / popneighsum if stateno==18

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+10]+`aaa`bbb''z[\_n+02] ) / popneighsum if stateno==19

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+26]+`aaa`bbb''z[\_n+28]+`aaa`bbb''z[\_n+18]+`aaa`bbb''z[\_n-12] ) / popneighsum if stateno==20

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+18]+`aaa`bbb''z[\_n-14]+`aaa`bbb''z[\_n+11]+`aaa`bbb''z[\_n+08]+`aaa`bbb''z[\_n+24] ) / popneighsum if stateno==21

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+27]+`aaa`bbb''z[\_n+01]+`aaa`bbb''z[\_n-08]+`aaa`bbb''z[\_n-13] ) / popneighsum if stateno==22

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+11]+`aaa`bbb''z[\_n+18]+`aaa`bbb''z[\_n-08]+`aaa`bbb''z[\_n+26]+`aaa`bbb''z[\_n-01] ) / popneighsum if stateno==23

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-06]+`aaa`bbb''z[\_n-23]+`aaa`bbb''z[\_n+18]+`aaa`bbb''z[\_n-20] ) / popneighsum if stateno==24

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-21]+`aaa`bbb''z[\_n+11]+`aaa`bbb''z[\_n-09]+`aaa`bbb''z[\_n+02]+`aaa`bbb''z[\_n-10]+`aaa`bbb''z[\_n-12]+`aaa`bbb''z[\_n-08]+`aaa`bbb''z[\_n+17] ) / popneighsum if stateno==25

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-14]+`aaa`bbb''z[\_n+24]+`aaa`bbb''z[\_n+15]+`aaa`bbb''z[\_n+08] ) / popneighsum if stateno==26

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-11]+`aaa`bbb''z[\_n-21]+`aaa`bbb''z[\_n+23]+`aaa`bbb''z[\_n+14]+`aaa`bbb''z[\_n-12]+`aaa`bbb''z[\_n-02] ) / popneighsum if stateno==27

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-23]+`aaa`bbb''z[\_n+09]+`aaa`bbb''z[\_n-16]+`aaa`bbb''z[\_n+16]+`aaa`bbb''z[\_n-25] ) / popneighsum if stateno==28

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-08]+`aaa`bbb''z[\_n+16]+`aaa`bbb''z[\_n-10] ) / popneighsum if stateno==29

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-22]+`aaa`bbb''z[\_n+08]+`aaa`bbb''z[\_n+02] ) / popneighsum if stateno==30

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-28]+`aaa`bbb''z[\_n+13]+`aaa`bbb''z[\_n-25]+`aaa`bbb''z[\_n+05]+`aaa`bbb''z[\_n+12] ) / popneighsum if stateno==31

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+06]+`aaa`bbb''z[\_n-02]+`aaa`bbb''z[\_n-25]+`aaa`bbb''z[\_n-11]+`aaa`bbb''z[\_n+13] ) / popneighsum if stateno==32

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+07]+`aaa`bbb''z[\_n-23]+`aaa`bbb''z[\_n+09]+`aaa`bbb''z[\_n+13] ) / popneighsum if stateno==33

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+07]+`aaa`bbb''z[\_n-08]+`aaa`bbb''z[\_n-11]

) / popneighsum if stateno==34

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-18]+`aaa`bbb''z[\_n-21]+`aaa`bbb''z[\_n-13]+`aaa`bbb''z[\_n+03]+`aaa`bbb''z[\_n+13] ) / popneighsum if stateno==35

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n+07]+`aaa`bbb''z[\_n-05]+`aaa`bbb''z[\_n-30]+`aaa`bbb''z[\_n-20]+`aaa`bbb''z[\_n-11]+`aaa`bbb''z[\_n-32] ) / popneighsum if stateno==36

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-32]+`aaa`bbb''z[\_n-09]+`aaa`bbb''z[\_n-25]+`aaa`bbb''z[\_n+10] ) / popneighsum if stateno==37

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-30]+`aaa`bbb''z[\_n-18]+`aaa`bbb''z[\_n+10]+`aaa`bbb''z[\_n-03]+`aaa`bbb''z[\_n-06]+`aaa`bbb''z[\_n-08] ) / popneighsum if stateno==38

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-32]+`aaa`bbb''z[\_n-18] ) / popneighsum if stateno==39

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-30]+`aaa`bbb''z[\_n-07] ) / popneighsum if stateno==40

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-07]+`aaa`bbb''z[\_n-14]+`aaa`bbb''z[\_n+09]+`aaa`bbb''z[\_n-15]+`aaa`bbb''z[\_n-18]+`aaa`bbb''z[\_n-26] ) / popneighsum if stateno==41

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-09]+`aaa`bbb''z[\_n-32]+`aaa`bbb''z[\_n-41]+`aaa`bbb''z[\_n-18]+`aaa`bbb''z[\_n-38]+`aaa`bbb''z[\_n-17]+`aaa`bbb''z[\_n-25]+`aaa`bbb''z[\_n+04] ) / popneighsum if stateno==42

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-12]+`aaa`bbb''z[\_n-07]+`aaa`bbb''z[\_n-39]+`aaa`bbb''z[\_n-25] ) / popneighsum if stateno==43

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-41]+`aaa`bbb''z[\_n-16]+`aaa`bbb''z[\_n-32]+`aaa`bbb''z[\_n+06]+`aaa`bbb''z[\_n-38]+`aaa`bbb''z[\_n-13] ) / popneighsum if stateno==44

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-16]+`aaa`bbb''z[\_n-24]+`aaa`bbb''z[\_n-13] ) / popneighsum if stateno==45

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-13]+`aaa`bbb''z[\_n-04]+`aaa`bbb''z[\_n-29]+`aaa`bbb''z[\_n+02]+`aaa`bbb''z[\_n-26] ) / popneighsum if stateno==46

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-10]+`aaa`bbb''z[\_n-35] ) / popneighsum if stateno==47

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-02]+`aaa`bbb''z[\_n-31]+`aaa`bbb''z[\_n-13]+`aaa`bbb''z[\_n-10]+`aaa`bbb''z[\_n-28] ) / popneighsum if stateno==48

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-26]+`aaa`bbb''z[\_n-34]+`aaa`bbb''z[\_n-36]+`aaa`bbb''z[\_n-27] ) / popneighsum if stateno==49

replace `aaa`bbb''neighave2 = ( `aaa`bbb''z[\_n-44]+`aaa`bbb''z[\_n-06]+`aaa`bbb''z[\_n-38]+`aaa`bbb''z[\_n-24]+`aaa`bbb''z[\_n-09]+`aaa`bbb''z[\_n-23] ) / popneighsum if stateno==50

drop `aaa`bbb''z

}

\*

\*Group #4

\*The following creates interactions between \*neighave2 variables and popratio.

\*First, the following creates "popratio", a variable that is the ratio of a state's population to the population of the states around it.

\*Fractions indicate that it is smaller than the combined population of the states around it, numbers greater than one indicate that the

\*states around it have a larger population than the state in question.

gen popratio=pop/popneighsum

\*Change "yyy" in the following line to reflect the number of variables you are going to produce the computations for.

\*This is the same number you put in above for group #1.

\*If you are only going to do this for one variable, put in the number "1."

\*Appearance 9 of 10 of yyy.

forvalues bbb=1/1 {

gen `aaa`bbb''neighave2int=`aaa`bbb''neighave2\*popratio

}

\*

sort stnum year

\*\*\*save F:/statedata8ns.dta

**1. Alabama**

**2. Alaska**

**3. Arizona**

**4. Arkansas**

**5. California**

**6. Colorado**

**7. Connecticut**

**8. Delaware**

**9. Florida**

**10. Georgia**

**11. Hawaii**

**12. Idaho**

**13. Illinois**

**14. Indiana**

**15. Iowa**

**16. Kansas**

**17. Kentucky**

**18. Louisiana**

**19. Maine**

**20. Maryland**

**21. Massachusetts**

**22. Michigan**

**23. Minnesota**

**24. Mississippi**

**25. Missouri**

**26. Montana**

**27. Nebraska**

**28. Nevada**

**29. New Hampshire**

**30. New Jersey**

**31. New Mexico**

**32. New York**

**33. North Carolina**

**34. North Dakota**

**35. Ohio**

**36. Oklahoma**

**37. Oregon**

**38. Pennsylvania**

**39. Rhode Island**

**40. South Carolina**

**41. South Dakota**

**42. Tennessee**

**43. Texas**

**44. Utah**

**45. Vermont**

**46. Virginia**

**47. Washington**

**48. West Virginia**

**49. Wisconsin**

**50. Wyoming**

**EXCEL**

#  When preparing an Excel file for STATA, make sure that only the *top* *row* has letters. STATA will read the top row as the variable names, but a second row will be read as the first scores on the variables. Additionally, if you are going to read an Excel file into STATA, save the Excel file as both an Excel file and a “tab delimited text file.” The “tab delimited text file” is the file STATA will read.

**To remove black bars on cells**: (1) right click on mouse; (2) click on “Format

Cells”; (3) click on “Border”; (4) click on “None” (as opposed to “Outline” or “Inside”) and (5) click on “OK.”

**To remove commas or percent signs**: change the heading in the upper middle

of the page (may appear as “Number Format” if you move the cursor to this part of the page) from “Number”, “Percentage”, etc. (there are approximately 11 different options) to “General.”

**To remove a green triangle from a column of cells**: (1) highlight the column;

(2) click on “Data” near the top of the screen; (3) click on “Text to Columns”; (4) click on “Next”; and (5) click on “Finish”. You can also do this one cell at a time by left clicking on the cell, left clicking on the “!” like icon that appears to the left of the cell and then clicking on “ignore error.”

**Sort: If sorting for a class, deleted the “names” column. Then highlight**

**the rest of the spreadsheet, click on “Data,” click on “Sort,” click on “Sort by” (choose the column you want the data sorted on the basis of – e.g., “ID”) and it should go from lowest to highest in the column you selected. If this doesn’t work, try what is written below.**

**To sort from, for example, lowest to highest scores in a column do the following: (1) put the column you wish to sort on the basis of in column “A” (i.e., furthest column to the left); (2) highlight column “A”; (3) click on heading entitled “Data”; (4) click on “Sort”; (5) choose sorting criteria (usually “lowest to highest” is the default option – i.e., to sort from lowest to highest you should be able to click on “OK” – YOU MAY NEED TO CHOOSE “EXPAND THE CURRENT SELECTION” – if it doesn’t work, check the option OTHER than “Expand the Current Selection”)**

**Reading an Excel file into STATA:** Make sure that any “missing data” is

denoted with a period (“.”), which is Stata’s missing data code (to do this read the “Recoding and Missing Data” section later in this document). Save the Excel file as a “tab delimited text” file to the “c” drive (or whichever drive your saving the data on - the tab delimited option is in the lower part of the gray box that will appear when you save the file (clicking on “Save as” in the upper left corner). You do not need a file extension. Note: a gray box in the middle of the screen will say that the file is not saved as an Excel file. Click on “yes.” Go into Stata. In the command line type: set mem 5000k and press “enter.” In the command line type: insheet using “C:/senate1.txt” (you need both the quotation marks and the txt on the end). Save as a Stata file by clicking in the upper left corner on “file” and then “save as” and make sure you remove the “\*” and have .dta as a file extension. Thus to save as a file called senate you would need the file name box to read: senate.dta (no asterik).

# Converting a STATA file into Excel: read the data file into Stata; click on

# “Window” (top of the screen); click on “Data Editor”; click on upper left corner to highlight and keep highlighting as you move right; click on “Edit” (top of screen); click on “copy”(or control “c” – i.e., “Ctrl c”); then go into Excel and “paste” (or control “v” – i.e., “Ctrl v”). Check to see that dots (i.e., “.”) representing missing data are in the new Excel file. If not, follow

 “Recoding and Missing Data” section ahead.

**Undo Last Command:** control z (i.e., “Ctrl z”)

**Leave a Cell Blank in which You Had Previously Typed a Score:** using the

mouse, highlight the cell and then push “backspace” key

**Select Particular Non-Consecutive Items from a Row or Column: First, look**

**under Stata (i.e., do it in Stata, NOT Excel) discussion under a heading: Deleting Observations from a Dataset.** If you want to select say i.tems 5, 10 and 15 from a row but not the intervening items depress the “Control” key while selecting items. If you want to select a block (e.g., not items 1-4 but 5-15 consecutively) hold down the “Shift” key. Now right click on mouse and select “copy.” The use paste or “control v” to transfer the data to the other file. Read section below entitled: **Copy and Paste Non-Consecutive Rows or Columns (e.g., transforming data organized by year to data organized by state in which case you might want to copy and paste lines 2, 52, 102, etc.)**

**Repeating a Score Down a Column:** click once on the cell you want to

duplicate and “drag” down the column to highlight. Then use: ctrl d (i.e., “control d”) to duplicate the cell. Move the mouse outside the highlighted area and click.

**Moving Data Up or Down a Cell:** To move data “up” a cell you need to use the

 “cut and paste” command by highlighting and then clicking on “edit”; To

 move data “down” a cell you could “cut and paste” or: (1) Highlight

the cell you want to omit (i.e., to bring every score up a cell); Click on

“Insert” (near top of screen); (2) Click on “cell” and it will give you options (the option with the dot already in it is the one to move the data down.

# For Manipulating a Column, Multiplying (e.g., Creating an Interaction Term)

# or Dividing an Entire Column by Another Column or Scalar:

#  (1) Highlight the letter above the column to the left of where you want the new column to go; (2) Click on “Insert” (top of screen); (3) Click on “column” (if you want to get rid of a column – click on “edit” and then click on “delete”); (4) Click on anywhere not highlighted to save the change; (5) Click on the second row (the first row under the column title or header – i.e., where the first data goes); (6) Type an “equal sign” (=) and enter the necessary formula [for example, to multiply column “D” by 2 (starting with row 2 in column D – the first column with data) type: 2\*D2 (- the last “2” in the formula is for row 2 in column D: for division replace \* with / - to multiply column D times column E type: =D2\*E2]; (7) Press “enter”; (8) Highlight row 2 of the new column you are creating (which in this example is probably column C - i.e., the first cell in the new column that contains data – which should be the score in column D row 2, doubled) and every cell in the new column that you want the new data to appear in; (9) ctrl d

**Recoding and Missing Data (Search and Replace in Excel)**: (1) highlight the

Data (probably, column) you want to change (if you want to change all the columns just keep highlighting as you move from left to right); (2) Click on “Edit” (upper left side of screen); (3) Click on “Find”; (4) In the “Find What” box tell it what number you want to change [if you want to convert a blank space into a “dot” (i.e., “.”) then leave the “Find What” box blank – i.e., don’t type the word “blank” in the box]; (5) Click on “Replace” (right side of gray box); (6) Put the number you want to change it to in the “replace with” box (if a dot –“.” – then put a “.” in the box); (7) Click on “Replace All.” If You need to move the gray box so it does not cover the column you are working with click in the blue area of the find/replace box and “drag” it to uncover what you need to uncover; (8) make sure to click to remove the black screen (if not, I don’t think the change occurs)

**To Highlight an Entire Dataset:** Click on the blank gray box which is to the left

 of column A and above row 1.

**Adding a Row to an Existing Excel File:** Click on the row number just below

where you want the new row inserted. You will see an entire row turn black. Then go to the top and pull down the menu that says “Home.” Toward the right side of the top row you will see “Insert.” Click on “Insert” and you will then have an option for “rows, columns or cells” listed below one another. Highlight “rows” and click. A new row should be created. If you want to add another row directly below the newly created row simply hit “control Y.”

**Transposing a Row to a Column (or vice versa):** See discussion in Stata

under “Transpose Rows and Columns.” In Excel: (1) highlight the row you

want to transpose; (2) under “edit,” click on “copy”; (3) open the document you want the new data in (it may be preferable to put it in a separate spreadsheet and work from there); (4) under “edit,” click on “paste special”; (5) check “transpose”; (6) click on “OK.” Given that Excel limits you to 256 variables, you might need to convert an Excel file in Stata. Save the Excel file as .csv and, using the “insheet” command in Stata, read the file into Stata. Then use the “xpose” command to transpose your data (see Stata manual on using the “xpose” command).NOTE: In “Vista” version of Excel you: (1) open document you want to transfer the transposed data to; (2) highlight want you want to transpose; (3) right click on the mouse; (4) click on “copy”; (5) right click again on mouse and click on “paste special”; (6) check “transpose” You might also try Stata.

In Stata type: xpose, clear (You’ll lose the variable names however)

**Delete a Row or Column:** (1) highlight the row or column you wish to delete; (2)

under “edit,” click on “Delete.”

**Downloading a Non-Excel Data File into Excel:** (1) avoid Adobe Acrobat files

(i.e., “pdf” files – choose html); (2) click and highlight the data you wish to download; (3) in the upper left corner click on “Save As”; (4) while you can use any “File Name” you want, be sure that across from “Save as Type” you choose (i.e., highlight) “Text File” and for “Encoding” select “Western European (ISO)”; (5) choose the drive you want to save to and click on “Save”; (6) Open Excel; (7) under “Files of Type” choose “Text Files”; (8) double click on the file name for the file you downloaded; (9) choose “Fixed Width”; (10) click on “next”; (11) click on “next” again; (12) click on “Finish”; (13) save the data as a “Microsoft Excel Workbook” file.

**Data in Excel Do Not Match Columns in Excel:** if the data in the resulting

Excel file do not “lineup” with the Excel cells (e.g., the downloaded data are wider than a typical Excel cell), go to a space halfway between two columns (e.g., half way between the letters “B” and “C”), wait to have a “plus sign” (or cross like figure) with left/right arrows appear, left click with the mouse and you should see dashed lines running down the columns which you can then manipulate.

**Reducing the Width of Columns (e.g., the columns are too wide)** –

1. highlight the columns that are too wide; (2) click on “Format”; (3) click

on “Column”; (4) click on “AutoFit Selection”

**Data in Excel Can Not be Manipulated (e.g., can not transfer rows and**

**columns or perform other functions):** follow the commands in “Data in Excel Do Not Match Columns in Excel” except; (1) go half way between columns “A” and “B” and drag the dashed line on the right side as far to the right as you can (i.e., put all the data as well as names that might appear on the left side of page in column “A” – it will be very wide – only the data will end up being transferred); (2) highlight the data you are interested in; (3) click on “Data” at the top of the screen; (4) click on “Text to Columns”; (5) click on “Next” (“Fixed Width” should be the default which will have a “dot” in it); (6) click on “Finish.” You may need to repeat this procedure because of how the data are blocked. For example, you may have data on all 50 states with left/right lines separating one group of years from another. In such a circumstance you need to go below each set of left/right lines (i.e., you don’t want to highlight the left/right lines or other such writing or figures, just the data you want to use) highlight the 50 state scores and then repeat the process for another year or set of years.

 **Note:** If the above does not work on any part of your document try the

following: (1) repeat steps 1-5 (i.e., through “click on Next”); (2) In the “Data Preview” box click between the numbers on what appears to be a ruler at points that place vertical lines isolating the columns you want to use (i.e., say at 15, then 30, etc.) – go as far to the right as your data allow; (3) click on “Next”; (4) click on “Finish” – If this doesn’t work then repeat initial steps 1-4 (i.e., through “Text to Columns”) and then change from “Fixed Width” to “Delimited” and follow as immediately above. Also, you might need to copy (i.e., highlight) what you want to move, click on “copy,” then “paste special” and click on “values.” I’ve been able to move stuff from a downloaded Excel file into another Excel file by this method.

**Copy and Paste Non-Consecutive Rows or Columns (e.g., transforming**

**data organized by year to data organized by state in which case you might want to copy and paste lines 2, 52, 102, etc.):** If you keep the “control” key down, you can highlight non-consecutive rows or columns. Unfortunately, when you then try the copy or cut option, it won’t work. So, you need an alternative strategy. Try the following: (1) duplicate the entire file so you won’t harm the original; (2) take the unit you trying to “stack” by (e.g., state or year) and copy the names of the various units so they are on both the left and right side of the spreadsheet (i.e., it is much easier to know what unit you’re on if the information is on both sides of the spreadsheet); (3) highlight all the information between consecutive observations on the unit (e.g., if you have data for the 50 state by year – observation 1 is Alabama in year 1920 and observation 2 is Alaska in 1920 and want to covert this so that observation 2 is Alabama in 1921 then highlight all rows between Alabama in 1921 and Alabama in 1922) and then right click on “copy” and then paste (e.g., using “control v”) into the new document.

**Removing a Symbol (e.g., “$”) from a Column of Data –** (1) highlight the row

or column the symbol you want to remove is in; (2) click on “Format”; (3) click on “Cells”; (4) highlight “number” (to covert to just numbers); (5) click on “Okay”

**Reducing the Size of a Cell (e.g., the data take two rows for each cell and**

**one of the rows is blank)**: (1) select all the cells that have information; (2) go to

the “Edit” menu; (3) within this menu go to “Clear”; (4) from the list select “Formats” (the cells should now be broken into two – now we will delete the blank cells); (5) go to the “Edit” menu; (6) within this menu select “Go To”; (7) on the “Go To” menu click “Special”; (8) on the list select “Blanks”; (9) press “OK”; (10) go to the “Edit” menu; (11) within this menu select “Delete”; (12) in the delete box select “Shift cells up”; (13) press “OK”

**Interpolation**: Let us say that you are working with annual time series data and

have scores at decade wide intervals (e.g., census figures every 10 years). If you want to average the change over the intervening years, precede as follows: (1) find the amount (or rate) of change over the time interval you are interested in (e.g., if there is a 2 point change over 10 years, this would be .2 of a point per year); (2) In the first cell in which you lack data, type a formula such as: =A2-.2 (this indicates a decrease of 2 tenths of a unit per year from the score in cell A2 – assuming cell A1 contains the name of the variable, cell A2 would have the first score – you type the preceding formula in cell A3 – i.e., the first cell without data - other mathematical signs are \* for multiply, + for add and / for division) and press “enter”; (3) Highlight the cell you just typed the formula in (e.g., A3) and as many cells as you want this same amount of change to extend; (4) Click on “Edit”; (5) Click on “Fill”; (6) Click on “Down” – or whatever direction you are going; (7) Click somewhere else to remove the highlight.

Interpolating at decade intervals over multiple decades. Put the first value in cell A1 (e.g., 1940) then, assuming by year, put the next decade you have data for in cell A11 (e.g, 1950) and proceed as follows:

1.Put the first value in cell A1 and the last value in the appropriate cell – in the

example below that is cell A11

2.Click on cell "B1" and type in the formula box at the top type:

=(A11-A1)/(ROW(A11)-ROW(A1)) and press “enter”

3.Highlight cell A2 and in the formula box at the top type: =A1+B$1 and press

“enter”

4.Highlight from cell A2 to A10 (or to the cell just prior to the end of the time point

for which you have actual data)

5.Under “Editing” (toward the upper right side of the screen) click on “Fill” and

then “Down”

=(A21-A11)/(ROW(A21)-ROW(A11))

=A11+B$11

=(A31-A21)/(ROW(A31)-ROW(A21))

=A21+B$21

=(A41-A31)/(ROW(A41)-ROW(A31))

=A31+B$31

=(A51-A41)/(ROW(A51)-ROW(A41))

=A41+B$41

=(A61-A51)/(ROW(A61)-ROW(A51))

=A51+B$51

=(A71-A61)/(ROW(A71)-ROW(A61))

=A61+B$61

=(A81-A71)/(ROW(A81)-ROW(A71))

=A71+B$71

=(A91-A81)/(ROW(A91)-ROW(A81))

=A81+B$81

=(A101-A91)/(ROW(A101)-ROW(A91))

=A91+B$91

=(A111-A101)/(ROW(A111)-ROW(A101))

=A101+B$101

=(A121-A111)/(ROW(A121)-ROW(A111))

=A111+B$111

=(A131-A121)/(ROW(A131)-ROW(A121))

=A121+B$121

**Statistics in Excel:** 1 – open an Excel file; 2 – click on “Tools”; 3 – click on “Data

Analysis”; 4 – click on “Regression”; 5 – make sure that the dependent variable is either in Column A or the last column on the right; 6 - It is important to remember that Excel does not read letter variable names. Also, the variables need to be by column (i.e., one column per variable). For example, suppose your first column (i.e., column “A” in Excel) contains 100 observations on a variable named “CONS.” Excel can’t read the variable name “CONS.” Since the variable (i.e., column) contains 100 scores beginning with the second row and ending with the 101st row, enter the following in the box: $A$2:$A$101 (i.e., Excel will read cells A2 through A101). If “CONS” were the dependent variable (i.e., “y”) in a regression, then $A$2:$A$101 would go in the “y” box. In my case, the independent variables were in columns A through C and the dependent variable was in column D. It would appear handy to have the independent variables next to each other. Thus, if you have 4 variables (i.e., the dependent variable and three independent variables), make your dependent variable either the first, or last, column. In the “Input Y Range” box I entered: $D$2:$D$101 and in the “Input X Range” box I entered: $A$2:$C$101 and check “Residuals” below.

**Fill Command:** To quickly fill a cell with the contents of an adjacent cell, you can press CTRL+D to fill

from the cell above or CTRL+R to fill from the cell to the left. To fill a cell with the contents of a cell below it (that is, to fill up), on the **Home** tab, in the **Editing** group, click **Fill**, and then click **Up**. To fill a cell with the contents of a cell to the right (fill left), on the **Home** tab, in the **Editing** group, click **Fill**, and then click **Left**.

Omitting Commas: To get rid of commas (i.e., thousands separator): (1)

highlight everything you want converted; (1) look for “General” in the middle of the heading on top; (2) change it to “Number” (this should “do it” – if not, click on the “,” icon – you’ll figure it out).

Getting “#NA” Rather than a Number when Cutting and Pasting: Highlight

what you want to cut and paste, click on “Paste Special” and look for a numbers option (e.g., 1 2 3 in a diagram that appears when you select “Paste Special”. Use the options with the right click on the mouse rather than “ctrl c” and “ctrl v” to copy and paste.