

## The University of Illinois at Chicago Economics 346: Econometrics

### ICARUS Basics of SAS to Enter and Process Data

1. The CMS users will start with interactive SAS, but Icarus runs batch SAS. Interactive SAS is giving SAS a command, and having it execute that command. Batch SAS is writing a program in a file and sending the whole program to SAS to execute. If you run SAS batch, you use Pico or Xedit to create a SAS program file (call it *myfile.sas*) and save it on your account. You run the whole program at once with the command *sas myfile*, and the output is sent to your account in 2 pieces. SAS's comments on how everything is going are in *myfile.log*, and your commands' output is in *myfile.lst*. We will call our file *probl.sas*. The following commands open the file and enter the first 2 lines of SAS commands, which tell SAS where the data are.

```
pico probl.sas  
options linesize=80;  
filename dat '/usr/local/lib/b34slm/classdata/penrub21.data';  
data probl;  
infile dat;
```

Note that ALL SAS commands end with semicolons. In our program, the dataset will be called *PROBL*, and it consists of 6 data series, listed in the *INPUT* statement: *RENT*, *NO*, etc. To check that you are linking to the right data, I will give you series means in your homework handouts to compare with the data. *PROC MEANS* is the SAS command to compute the means.

```
input rent no rm sex dist rpp;  
proc means;
```

The 6 means for the series should appear in your *PROBL.LST* file, along with the standard deviations (*STD*), and minimum (*MIN*) and maximum (*MAX*) values for each variable. For example, the mean for *RENT* is 318.156.

2. The next step is to print the data. If we want to restrict the operation to a subset of the data set, we can use the *VAR* statement after the *PROC PRINT* command. A *VAR* statement is

```
VAR list;
```

where *list* is a list of variable names. The spelling of each name must match exactly the *INPUT* statement.

```
proc print;  
var rent no rm dist;
```

3. To plot the data, use the *PROC PLOT* procedure. These

instructions create a scatter plot of RENT and RM, with RENT on the Y axis. Does this look like a straight line? Are there obvious outliers?

```
proc plot;  
plot rent*rm;
```

4. Now we will perform a regression. We will use RENT (rent on Ann Arbor student apartments, in \$) as a dependent variable RM (number of rooms) as the independent variable. PROC REG is the SAS procedure for linear regression. Since no dataset is indicated, the most recent one is used. The MODEL statement tells SAS which is the dependent variable (RENT) and which is the independent variable (RM). Unless otherwise specified, SAS includes an intercept in the estimated equation. If options are requested, a slash follows the last explanatory variable and the desired options listed after the slash and before the semicolon. In the second MODEL command we request the options that the regression print the fitted values (P), print the residuals (R) and calculate the Durbin-Watson statistic (DW).

```
proc reg;  
model rent=rm;  
model rent=rm / p r dw;
```

5. To save the SAS program on your account and then run it, type CTRL-X (see hints at the bottom of the screen), remember that we called it *probl.sas*, and when you are out of the editing program, type *sas probl*.

```
^x  
sas probl
```

Your results will be in *probl.lst* (your LISTING file) and comments by SAS on how each step went (useful for identifying errors if the program did not run) will be in *probl.log* (your LOG file). You can print the listing file or download it for editing in a word processing program, edit it on Icarus, etc. The UNIX command *ls* lists the files on your account.

6. We also will want to read data from files on your account. The first step is creating a data set. Data set names should be less than 8 characters long, and certain symbols (such as %) are prohibited.

```
pico ps5.data
```

This command opens a file on your account called PS5.DATA and puts you in input mode, so you can enter the numbers. We will use this GDP growth and inflation data in problem set 5. The variables are YEAR, V (share of votes in presidential elections going to Democrats), G3 (GDP growth), and P15 (inflation).

1916	0.5168	2.229	4.252
1920	0.3612	-11.463	16.535
1924	0.4176	-3.872	5.161
1928	0.4118	4.623	0.183
1932	0.5916	-15.574	6.657
1936	0.6246	12.625	3.387

When you are done with the table, press CTRL-X to file your data on your account in *ps5.data*. Note, don't use tabs to separate your columns, use at least one space between each data point.

- The dataset is now on your account. Now we will access the data. You use the FILENAME command in your SAS program to tell SAS where the data are. Note that the FILENAME command can come before or after the DATA command naming the dataset within SAS.

```
pico ps5.sas
options linesize=80;
filename ps5data 'ps5.data' ;
data prob2;
infile ps5data;
input year v g3 p15;
proc print;
proc means;
endsas;
```

- Save the file and run it.

```
^x
sas ps5
```

Now you can examine the LISTING and LOG files on your account, and print, download or edit them.

- Here are some commands for creating a new variable from one or more existing variables. This most commonly involves an instruction in the DATA command

new variable = expression using an existing variable

In SAS, this command must end with a semicolon. If the variable has not yet been read in, but exists in the SAS INPUT statement, the program will act as if it already has information about this variable and you can use it to create a new variable. Here are some examples:

```
DATA A;
INPUTY X GDP;
Y = LOG(X); (for the natural logarithm of X)
Z = EXP(X); (for e raised to the power of X)
GDPLAG = LAG(GDP); (for the lag of the variable GDP)
```

```

GDPLAG2 = LAG2(GDP); (for a lag of 2 periods (or use other
numbers)
CHGGDP=DIF(GDP); (or CHGGDP=GDP-LAG(GDP); for the 1-period
change )
CHGGDP=DIF4(GDP); (for the 4-period difference--change from 1
year ago for quarterly data)
GRGDP=(CHGGDP/GDPLAG)*100; (for the 1-period growth rate)

```

10. In SAS, the processing of data generally is separate from the creation of the data set. Data sets can be created in a DATA step (as we are doing) or as a joint output generated when data are processed in what are called PROC's. Data manipulation only takes place in DATA steps, however. The command CARDS; separates the set of SAS commands from the data values that follow. Note that all commands must end with a semicolon, but not the entered data. To try this part, create *prob3.sas*, input the commands and data, then run it.

```

pico prob3.sas
options linesize=80;
data consexpd;
input year pce ipdpce;
rpce=(pce/ipdpce)*100;
grpce=dif(rpce)/lag(rpce)*100;
ipdpce80=(ipdpce/71.4)*100;
rpce80=(pce/ipdpce80)*100;
grpce80=(dif(rpce80)/lag(rpce80))*100;
cards;
1980 1748.1 71.4
1981 1926.1 77.8
1982 2059.2 82.8
1983 2257.5 86.2
1984 2460.3 89.6
1985 2667.4 93.1
1986 2850.6 96.0
1987 3052.2 100.0
1988 3296.1 104.2

```

11. Now that we have successfully completed a data set, let's get summary statistics for each of the variables. To check for data entry errors, it is useful to view the minimum and maximum values, as well as the mean and standard deviation. To do this we use a PROC statement, which has the general form

```
PROC name DATA=datasetname options;
```

We will use PROC MEANS;; the datasetname can be omitted if the most recently created one is to be used, and options means the details of the PROC command that we want to use. We want the mean (MEAN), standard deviation (STD), minimum (MIN), and maximum (MAX) values, as well as the SKEWNESS and KURTOSIS for each variable.

```
proc means mean std min max skewness kurtosis;
```

12. It can be useful to combine datasets to add data or additional variables. This is done in a DATA step. Let's enter some more data and combine the 2 datasets. Similarly, you can break datasets into parts in a DATA step.

```
data unemp;
input ur cu;
cards;
5.8 84.6
7.1 79.3
7.6 78.2
9.7 70.3
9.6 73.9
7.5 80.5
7.2 80.1
7.0 79.7
6.2 81.1
5.5 83.5
```

We merge the two data sets in a DATA statement. We can print our merged data set. This works if they cover the same time period or if they have a variable in common. Here we have annual data from 1980-1988. If year were in both data sets, one went from 1970-1990, and the other 1960-1988, the command BY YEAR; after the MERGE command would have SAS match the years in common and assign missing values (.) to years not covered by each variable.

```
data all;
merge consexpd unemp;
```

We can print the data, choosing a variable, usually a date variable, as the leftmost through the ID command.

```
proc print; id year;
```

13. Now let's examine the errors, fitted values and confidence intervals from a regression. So first, we direct the regression to save these values. Then we create an output data set from the regression procedure named UNEMP-OUT. Naming the fitted values (P), the residuals (R), and the lower and upper confidence bounds (L95 and U95) means we can PRINT them or perform calculations with them. The name of the fitted values is PRED, and so on. PROC PLOT graphs data. Here, we look at the residuals on the y-axis and year on the x-axis, with a line identifying zero (the VREF=0 option).

```
proc reg;
id year;
model ur=cu / p r cli clm;
output out=unempout p=pred L95=L95 u95=u95 r=resid;
proc means;
proc plot data=unempout;
```

```
plot resid*year /vref=0;
```