ECON 20: Lecture #3
Stata Introduction
Companion for R Users

This document shows how to do the steps in the STATA introduction for students who want to use or learn R instead of STATA. If you are going to use R, I highly recommend downloading and using Rstudio, a free program that provides a much better user interface.

1 Getting started

Before even getting started, I recommend installing a few packages in R that we will need for this exercise. Type the following commands into R:

```
install.packages('rio')
install.packages('dplyr')
```

The `rio` package allows R to quickly and easily read many different types of datasets (including STATA datasets). The `dplyr` package is part of a family of packages that greatly simplify basic operations on datasets, such as generating new variables.

Each time you use R, you need to tell it to load installed packages into memory before they can be used. So after installing, you should type:

```
library('rio')
library('dplyr')
```

so that you can actually use the packages during your R session. When writing R programs (see below), the first couple lines of your program should also load the necessary packages using a set of `library()` commands.

To change the working directory in R, use the `setwd()` command:

```
setwd('~/Dropbox/Econ20/stata-intro')
```

2 Loading the data

Now that we are working from the correct directory, we can load our dataset into R by typing the following into the command line:

```
cps <- import('cpsmar08_10pct.dta')
```

This first command highlights the single most important difference between working with STATA and R, which is how things are stored. STATA essentially operates on a spreadsheet – we load the spreadsheet into memort and that is *the dataset*. In R, you can simultaneously have many different "datasets" in memory and they can be stored in many different formats.

The command above tells R to load our CPS dataset and name it `cps`. It will be stored as a *data frame*, which is the closest thing to a STATA dataset in R.

3 Exploring the data

The simplest equivalent to the STATA `browse` command in R is the `head()` command. Typing `head(cps)` will display the first five rows of the `cps` data frame including variable names. You can use `head(cps,N)`, where N is a number, to display the first N rows. The `tail(cps,N)` command works the same way except that it prints the last N rows.

In R, missing values will be displayed as `NA`, rather than as a period like in STATA. Many of STATA's basic commands ignore missing values by default, but many of R's basic commands do not. Suppose we want to find the mean of the wage variable in our dataset. To do this, we type `mean(cps$wage_hr)` into R. The `$` operator is the standard way to tell R to look for a column within a data frame (remember that our dataset is called `cps`, so the command is telling R to compute the mean of the variable `wage_hr` which is a column in `cps`). The output from issuing this command is:

```
> mean(cps$wage_hr)
[1] NA
```

Because `wage_hr` has missing values, we get a missing value for the mean. To fix this, use:

```
> mean(cps$wage,na.rm=TRUE)
[1] 16.98556
```

4 Basic operations

Creating variables

I recommend using the `dplyr` syntax for this type of operation. To create the new variable `lnwage`, use:

```
cps <- mutate(cps,lnwage=log(wage_hr))
```

The mutate command works by first telling R what dataset to operate on and then providing a set of operations to perform. You could acheive the same goal with

```
cps$lnwage <- log(cps$wage_hr)
```

but I recommend using the `mutate()` framework because it is much more flexible and can handle many operations at once.

To get the summary statistics for our new variable, use:

```
> summary(cps$lnwage)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   -Inf   1.907   2.619    -Inf   3.139   5.441   13415
```

Uh oh! This is one issue that comes up in R when working with logs. In STATA, taking the natural log of zero results in a missing value, but in R it generates `-Inf`. So we need to replace those values to missing. There are a lot of ways to do this, but my preference is using the `dplyr` and `case_when` framework:

```
> cps <- mutate(cps,
+               lnwage = case_when(wage_hr>0 ~ lnwage))
> summary(cps$lnwage)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  0.277   2.398   2.823   2.849   3.261   5.441   13867
```

The `case_when` line replaces `lnwage` to `NA` when `wage_hr` is missing or 0.

Regressions

To estimate a regression, we use the `lm()` command (for linear model). For our wages and education regression, we use:

```
reg <- lm(lnwage ~ educ,data=cps)
```

Again, we need to remember the difference in how things are stored between STATA and R. The above command tells R to store the regression from `lm()` as an object called `reg`. To get R's equivalent to STATA's regression output, we can apply the `summary()` operator to the regression object:

```
> summary(reg)

Call:
lm(formula = lnwage ~ yrsed, data = cps)

Residuals:
     Min       1Q   Median       3Q      Max
-2.57830 -0.34745 -0.01198  0.35347  3.04942

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.21416    0.06994   17.36   <2e-16 ***
yrsed        0.11778    0.00494   23.84   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.5586 on 1659 degrees of freedom
  (13867 observations deleted due to missingness)
Multiple R-squared:  0.2552,  Adjusted R-squared:  0.2548
F-statistic: 568.6 on 1 and 1659 DF,  p-value: < 2.2e-16
```

R does not provide the MSE automatically, but we can calculate it easily using:

```
> var(reg$residuals)
[1] 0.3118519
```

The regression object stores the residuals from the regression as `reg$residuals`, and to get the MSE we can just take the variance of the residuals.

Plotting relationships

To construct our scatter plot of wages and education, we can use:

```
plot(cps$yrsed,cps$lnwage)
```

Again, we are using the `$` operator to tell R to look for variables within our dataset which R knows as `cps`.

Saving plots is somewhat more complicated in R. The idea here is to first initialize a blank piece of paper, write on that blank piece of paper, and then tell R we are done writing. In practice, the commands look like this:

```
pdf('wagescatter.pdf')
plot(cps$yrsed,cps$lnwage)
dev.off()
```

This will store the graph as `wagescatter.pdf`. I think this will also work to create an image file, i.e. a `.png` file, but I am not sure.

Creating more complicated plots also uses a different "idea" than in STATA. Basically, we will create our initial scatter plot and then add to it with additional lines of code. To create a scatter plot with the linear regression line, you can use

```
plot(cps$yrsed,cps$lnwage)
abline(lm(lnwage ~ yrsed,data=cps))
```

There is a well-known plotting package for R called `ggplot2` with a cult following that has its own syntax for plot creation. I am not very skilled with `ggplot2`, but you are of course free to explore this on your own.

5 R programming

The equivalent of a `.do` file in STATA is a `.R` file. You execute an R program by typing `source('PROGRAM.R')` in the command line. To my knowledge, there is no direct equivalent of a log file in R. You can store the output of a program using the `sink()` command by insering a line like this at the start of your program:

```
sink('PROGRAM.txt')
```

which will capture the output of your program into that `.txt` file. To make sure everything you want gets captured into this file, there are two options:

- When telling R to run your program, add `verbose=TRUE` to the command, i.e. `source('PROGRAM.R',verbose=TRUE)`.
- Tell R to `print()` output you are especially interested, i.e. you could insert `print(summary(reg))` after estimating the regression to *print* the output of the `summary(reg)` command into your log file.

Below is a sample R program to accomplish the tasks from the STATA exercise:

```
### CLEAR MEMORY (SAME AS CLEAR ALL IN STATA) ###
rm(list=ls(all=TRUE))

### INSTALL PACKAGES WE NEED ###
#install.packages(rio)
#install.packages(dplyr)
### LOAD PACKAGES WE NEED ###
library(rio)
library(dplyr)


## CHANGE DIRECTORY IF NEEDED ## ##
##setwd('~/Dropbox (Dartmouth College)/econ20-steve/lectures/3-regression/activity')

## START LOG FILE ##
sink('stata_intro.txt')

## READ IN DATASET ##
data <- import('cpsmar08_10pct.dta')

## CONSTRUCT LOG WAGE VARIABLE ###
cps <- mutate(cps,lnwage=log(wage_hr))
cps <- mutate(cps,
              lnwage = case_when(wage_hr>0 ~ lnwage))
summary(cps$lnwage)

## ESTIMATE REGRESSION ##
reg <- lm(lnwage ~ educ, data=cps)
print(suummary(reg))

### WHAT IS THE MSE? ###
var(reg$residuals)

## CONSTRUCT SCATTER PLOT ##
pdf('wagescatterR.pdf')
plot(cps$yrsed,cps$lnwage)
```

```
abline(lm(lnwage ~ educ,data=cps))
dev.off()

### FOR EXERCISES ###
### ESTIMATE REGRESSION OF RESIDUALS ON YRSED ###
### THIS IS EASIER IF WE JUST USE THE SUBSET OF DATA WITH NONMISSING LOG WAGE ####

## SUBSET OF CPS WITH VALID LN WAGE ###
cpssub <- filter(cps,!is.na(lnwage))

## REDO THE REGRESSION ON THESE DATA ###
reg <- lm(lnwage ~ yrsed,data=cpssub)

### REGRESS THE RESIDUALS ON YRSED ###
residreg <- lm(reg$residuals ~ cpssub$yrsed)
print(summary(residreg))
```