Basic Econometrics with R and STATA: A Cookbook Approach

By Stephen Matteo Miller[1]

[1] Comments welcome! I would like to thank my dissertation advisor, Joe Reid, for his many thoughts about how research gets done, and my teachers in econometrics and statistics: David Levy, Dan Carr and Jim Gentle. I also thank Rob Hyndman for helpful discussions about the inner workings of R, the Monash University Economics Graduate Students for helping me to sharpen the presentation of ideas, Nektarios Aslanidis, Matt Dobra, Pushkar Maitra, and Gaobo Pang for helpful comments and discussions. Finally, I would like to thank Grant Farnsworth for creating his Rosetta Stone-like "Econometrics with R", available from http://cran.r-project.org/doc/contrib/Farnsworth-EconometricsInR.pdf, which inspired me to write this.

Table of Contents

Introduction

Competition tends to lower prices. If you only work with one econometrics package, there is no competition. Since all packages are different, they each do some things better than others, i.e., there are comparative advantages. If two packages compete for your attention, the price you pay, in terms of your time spent doing research in the future, might be lowered if you invest some time now to learn more than one. This is the principle that motivates the presentation of the ideas in "The Basics of R and STATA: A Cookbook Approach." If time permits, I might add in other programs, to knock the price down further, but the market price for this "duopoly" is already considerably lower than for a "monopoly".[2]

While it is true that specialization makes you rich (which, would tend to nullify what I just said), unless you intend to acquire expert programming skills (in which case, it does make sense to specialize), there may be times when you have no time to figure out how to do one particular operation with one particular statistical package, and it looks good on a resume to have multiple statistical software packages listed anyway. If your knowledge is limited to one program, then you might have to do some things the hard way. That can mean lots of time lost. During my first year working at the World Bank, we routinely used to switch between STATA, SPSS, Excel and E-Views, with occasional forays in S-Plus. We were analysing household surveys, and the data sets were so large, and we had so many time constraints that switching was essential. So, if you can use more than one program, you can choose the one with the lower cost in terms of time lost. Hence, you face an "inframarginal" trade-off as you can specialize in becoming an expert in R and hence computational statistics, or you could use that time to pursue other interests. The choice is yours.

I will begin by referring you to a working paper I wrote long ago, which is sort of a comment on Mankiw, Romer and Weil's (MRW's) (1992) Augmented Solow Model, available from

http://www.gmu.edu/departments/economics/working/WPE_99/99_09.pdf

Although I once submitted it for publication, I have no intention of ever publishing this paper, because it's much better suited as a teaching device about statistical visualization. So it is useful here because it will make it easy to relate the ideas underlying my intent in writing that paper, and the approach and objectives that I hope to meet in teaching how to use R and STATA.

The Augmented Solow Model is simple to understand. If you know the Solow model, then just add in education, that's it. Of course, the model is not the production function/phase-diagram version, but what's known as the "log-linearization around the steady-state" version of the model. In other words, you begin in the steady-state, and then pull back on your sling-shot to observe how the model lands back in the steady-state. I think there are real data problems with MRW's paper, which I identify. The data I used in that paper were actually from the Journal of Applied Econometrics data archive homepage, specifically from the entry for Jonathan Temple's (1998)

[2] I have added in the output from an exercise in E-Views for presenting co-integration analysis, but since I do not have the authority to distribute the data, I'll only report the results, just to show you the process of doing co-integration analysis.

paper in which he too also criticizes the Augmented Solow Model. If you did not know, the Journal of Applied Econometrics requires all data used in papers they publish to be posted at http://qed.econ.queensu.ca/jae/. Hence, it is a great source of data, if you can't seem to find any relevant data elsewhere.

So, why does Temple write his paper, and why did I write the above-mentioned? What MRW do is apply a classical statistics methodology to test the model they propose. Cleveland (1993) attributes the classical statistical paradigm to Fisher's (1958) book first published in 1925. The classical statistics paradigm is: theorize first, and then test with data. This is the approach taken by many economists. Steve Levitt at Chicago (of Freak-o-nomics fame) seems to advocate a slightly different, less formalized approach, in which you put forth a theory, but don't look to test it directly, but instead do thorough statistical analysis of reduced form specifications to make sure the signs of the coefficients go the right way. I like this approach more than the pure classical statistics approach, but you might also consider other paradigms of data analysis.

John Tukey (1977) offers an alternative, which turns the classical statistical paradigm upside-down. Instead of theorize first, and then test, you look at, or explore, the data and then see what kind of theory this might suggest. This works well when you are doing research in a new field in which there is little theory, such as the human genome project. In economics, there is a long tradition of economic theory, so rather than toss out the baby with the bathwater, in my personal work, I sometimes work with a hybrid. I usually start with a really simple econometric model, and then explore it over time with data, sort of along the lines of Levitt's approach. I'm not suggesting you follow this approach. In fact it may not appeal to many, and there may be no room to do this in many applications, but by exploring a simple statistical model, I admit my own limitations, so that I don't oversell my results. Other closely related methods fall under non-parametric and robust statistics, and Tukey was a key contributor here as well.

The reason for thinking about alternatives to the classical statistical paradigm is that it assumes you know lots. If you read Mankiw, Romer, Weil (1992), you will get this impression. In Temple's paper, among his criticisms is the fact that a few influential observations, including Greece, Portugal and Turkey, are driving the results. In the paper I wrote, one thing I show is that most of Africa seems to have a different model than everyone else [and by the way, I think that's because of the institutions, moreseo than geography, and definitely not the culture]. Also, there is multi-collinearity between school enrollment and investment rates. If your aim is to get fitted values for the dependent variable, then multi-collinearity is not a problem. However, if you intend to explore the marginal impacts of the right-hand-side variables, good luck, as you won't be able to attribute how much individual explanatory power the collinear variables have. I'll borrow from a table I once saw my econometrics professor, David Levy, use to distinguish between the three approaches

Assume You Know Lots About Data	Assume You Know Little About Data
Classical Statistics/Econometrics: MRW	Robust Statistics: Temple
	EDA: the working paper

If you keep this in the back of your mind always, you will understand why there are many reasons to be skeptical about modern applied econometrics. There is another fascinating tid-bit of information about the origins of the classical statistics paradigm. Many of the founders (like Galton, Pearson) were unashamedly members of the Eugenics movement, in which highly educated, but perhaps not very street-wise, people sought to measure just how superior Europeans were to everyone else. For more details to this story, you can see *Vanity of the Philosopher* written by Sandra Peart and my dissertation co-advisor David Levy, which is a fascinating follow up to Levy's two previous fascinating texts, *How the Dismal Science Got Its Name*, and *The Economic Ideas of Ordinary People*. Even as the credibility of Eugenics died a long time ago, a remnant of that arrogance still remains in the classical statistical paradigm, and it is reminiscent of Adam Smith's (1981, Book I., Ch. II, Par. 24) observation

> The difference of natural talents in different men is, in reality, much less than we are aware of; and the very different genius which appears to distinguish men of different professions, when grown up to maturity, is not upon many occasions so much the cause, as the effect of the division of labour. **The difference between the most dissimilar characters, between a philosopher and a common street porter, for example, seems to arise not so much from nature, as from habit, custom, and education.** When they came into the world, and for the first six or eight years of their existence, they were, perhaps, very much alike, and neither their parents nor play–fellows could perceive any remarkable difference. About that age, or soon after, they come to be employed in very different occupations. **The difference of talents comes then to be taken notice of, and widens by degrees, till at last the vanity of the philosopher is willing to acknowledge scarce any resemblance.** But without the disposition to truck, barter, and exchange, every man must have procured to himself every necessary and conveniency of life which he wanted. All must have had the same duties to perform, and the same work to do, and there could have been no such difference of employment as could alone give occasion to any great difference of talents.

I added the bold type to emphasize the point. Put another way, everyone is important. So, if you keep that in mind as you do your research, you'll do well.

Overall, you'll see that in fact R and STATA do many things very well, but I find R has a slight advantage in that it has been heavily influenced by statisticians. I find statisticians have freed themselves from the assumption of normality and the central limit theorem, as well as moments-based estimation, much more-so than econometricians. To do so is liberating.

Why is this important? I remember David Levy once saying in class, "The world [of data] is messy. Infinity lives there." Infinity would not exist in a world where everything followed the normal distribution, but when does the data follow a normal distribution? That's something you should always keep front and center in your mind as you do research, but be warned if you try to go the way of the statistician, you may face resistance when trying to publish your work in economics journals, so be prepared to bridge both worlds.

With this in mind, there are areas where STATA currently does things better (like ordered logit, generalized ordered logit). For GARCH models, while STATA does a little better than R, you'd probably be better off using something like E-Views, *Ox*©, or S-Plus's Finmetrics.

How to Download R: The Program

Since STATA is costly, if you don't have access to this software package, and do not have money to spend on software, do not despair. The R open source statistical package can be downloaded free of charge from http://cran.r-project.org/, or http://cran.au.r-project.org/. After you go to this web-site, you can simply look to the middle of the page that pops up, and you'll see a box, this particular version having been copied from the web-site in April 2006 (keep in mind it can change fairly frequently, as versions 2.3.1 and 2.4.0. came out in the following six months)

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Linux
- MacOS X
- Windows (95 and later)

Source Code for all Platforms

Windows and Mac users most likely want the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- **The latest release** (2006-04-24): R-2.3.0.tar.gz (read what's new in the latest version).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

Click on "Windows (95 and later)" in the **Download and Install R** section. This takes you to another page, where you see base and contrib. Click on base and select "R-2.3.0-win32.exe", or whatever is the latest version, to be saved to your desktop. It's about 27 megs, so make sure you have room. If you double-click on the icon, a language menu pops up [you can even choose Catalan or Sinhalese]. Once you pick the language, click yes to whatever Windows Installer asks. It might be good to click on the boxes for the three help files so that when you do a help search in R, you can actually get a link to the internet. Also, Rob Hyndman solved a mystery for me.

For anyone working from a Monash-based computer, once you download the program and the icon is created, you should "right click" on the icon and a menu pops up. Under Properties, and Target, you see something like

"C:\Program Files\R\R-2.4.0\bin\Rgui.exe"

After the second quote you should copy and paste the following line

http_proxy=http://proxy.monash.edu.au/ http_proxy_user=ask

Click okay. If you choose this, you'll have to tell R your username and password when you update your packages. This way you can update your R packages, which contain additional routines and estimators, periodically using the command `update.packages(checkBuilt = TRUE)`. Alternatively, paste the following, putting your username where it says username and password where it says password

http_proxy=http://username:password@proxy.monash.edu.au:80/

How to Download R: The Packages

Just as you can add to STATA's capabilities by downloading routines contributed by STATA users, which you can find through STATA's help menu, and which is linked to the web, R has an ever increasing list of what are called "packages", contributed by R users. Packages here should not to be confused with the term "statistical software package" that I was using earlier. To download these packages, there is a not-so-straightforward way, and a simple way. The not-so-straightforward way is described in Grant Farnsworth's manuscript, but since I read that I noticed you can actually do this within R. If you look at the top of your R interface, there are several menus, and one of them is called Packages. First, you'll have to choose

Packages
Install package(s)…

You'll either be asked for your username and password or you'll go straight through, depending on what you selected above. Then you'll be asked for your location, and you can choose the one nearest you, like Australia (VIC). Then, you'll see a menu of packages that you can choose from. Once, you select it, it will remain in memory, and can periodically be updated using the `update.packages(checkBuilt = TRUE)` command that I mentioned above. Once you download the package to your computer, you still have to load in the package to make it functional each time you run R, which you can select from the following menu

Packages
Load package…

This will give you a list of packages that you can choose from, so select the one(s) of interest to you.

How to Read in Data in R and STATA

You can actually download the data I used in the working paper from the Journal of Applied Econometrics data archives homepage http://qed.econ.queensu.ca/jae/. That will take you to the entire archive, so to get the right data file

Step 1: Click & save the "temple.zip" file to whichever location you prefer from[3]

http://qed.econ.queensu.ca/jae/1998-v13.4/temple/

Step 2: Open up the zip file, and inside is a comma separated file called "testsol.csv," which is the data file that opens automatically in EXCEL if you double-click on the icon. Notice, there's already a potential data problem, namely that the column heading above the country names is blank, so fill it in with a name "country" or something else, and then save the file as a ".csv" file once again, perhaps with a name like "temple.csv". I've also saved it in the example below to an easy to remember location "C:/temple.csv" [**remember that R likes forward slashes**, so it will **NOT** read the location if it is written as "C:\temple.csv"], but keep it wherever you prefer.[4]

Step 3: Open up R, go to the File menu and select New Script, into which you can save the commands so that you don't have to retype your work. In that script file will go all the commands related to this exercise. Similarly, in STATA, open what's called a "do-file". They work on the same principles: you keep your codes in a file, preferably with descriptions of what the commands do, and then you can automatically execute, and even re-execute later, the same commands.

Step 4: Choose a name for the object you want to create in R, [I call it "`mydata`", but if you called it "`pumpkin`" it would work too] to which you will assign, "`<-`", the data file in comma-separated format, according to its exact location, followed by the command to treat the first row as the variable names, using the following command

R Code:

```
> mydata <- read.table("C:/temple.csv",header=T,sep=",")
```

[hit enter or "Run" icon][5]

---

[3] There is also a "readme" file, which contains a detailed description of what exactly is in that data file http://qed.econ.queensu.ca/jae/1998-v13.4/temple/readme.jt.txt
What's great about Jonathan Temple's paper is that he has also written some S-Plus codes, which are in the "temple.zip" file, but which I've not tried in R. If you're really interested in looking at the Jonathan Temple paper, he gives you many if not all of the ingredients to exactly replicate his paper, which by doing you will acquire a greater ability to do your own research in applied econometrics. Trust me on that!

[4] By the way, if you want the exact address of the file and don't know what it is, first find the file in Windows Explorer, then highlight the file's icon, and right-click on the mouse, select properties and on the menu that pops up, the exact location can be found next to "Location: ...". This you can copy and then paste into the command line I have in Step 4.

[5] The "read.csv" function in R is best if you only have numeric data, but if you have different types of data, try the "read.table" command in R, which gives you more options about reading in data, including whether or not the first row has variable names, using the "header equals True" command, or "`header = T`", and by telling R that the values are separated with a character called comma, or '`sep=","`' where the quotation marks around the comma means R thinks of the thing as a character. Also, notice that all the variable names become capitalized, unlike in STATA, which are always lowercase, unless you yourself specify capital letters.

STATA Translation:

```
clear
set mem 100m
insheet using "C:\temple.csv", comma
```

[hit enter or "do-" icon][6]

The first line clears the memory, the second line sets the memory that STATA will use to 100 megabytes. You probably won't require anything near that amount unless you start working with household surveys, but I did this just to be safe. Now R can read in at least some "foreign" data files, like STATA data files, so before you start, you might remember to type in the Load Package and select **foreign**. If, after you imported the data into STATA, as you'll see shortly, you saved that STATA data file as "C:\temple.dta", you could subsequently read it into R using the commands

```
> mystatadata <- read.table("C:/temple.dta")
```

Step 5: Presuming there are no problems with the file format, R should have read in your dataset. To verify, just type in mydata into the command prompt

R Code:

```
> mydata
```

[hit enter or "Run" icon]

and you should see the entire dataset flash before your eyes, including the variable headings. Alternatively, R also has a data editor that you can select from the Edit menu, as follows

Edit

Data editor…

From this you can see whether the data is numeric or character, and you can accordingly change the format from one to the other, as need be. The STATA equivalent of this is[7]

```
browse
```

[hit enter or "do-" icon] or

```
edit
```

[hit enter or "do-" icon]

Things look okay at first glance with the STATA data, BUT, you will see in the introductory section to regression analysis that there is a minor problem that can be corrected.

[6] Again, notice that R, unlike STATA and everyone else, uses forward slashes. R is funny that way. Also note that in STATA, if you want to give specific variable names that differ from the ones in the dataset, or if the dataset has no variable names, you would list the names exactly as you want them, in between "`insheet`" and "`using`", i.e. "`insheet var1 var2 var3 using …`"

[7] STATA does have a command prompt, so to see your data, you can either do "`browse`" or "`edit`" or hit the respective icons at the top of the screen

Simple Summary Statistics

The first thing you might try is get an idea about sample statistics for variables. In this case, what about the variable income growth (DY)

R Code:

```
> summary(mydata$DY)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
-1.1110  0.1317  0.4545  0.4489  0.7337  1.6590 16.0000
```

To get a summary of all the data, just remove the **$DY** and type **summary(mydata).**

STATA Translation:

```
summarize dy
```

[STATA recognizes underscored letters are sufficient to run the command]

```
Variable      Obs    Mean          Std. Dev.     Min           Max
dy            105    .4489098      .4760209      -1.11122      1.659233
```

If you just type **summarize**, without any variable names, like with R's **summary(mydata)** command, you will get results for each variable in the dataset. Let's compare. Well, STATA gives you more aspects of the distribution in the summary command. STATA summarizes location (i.e., the mean), and scale (i.e., standard deviation), as well as observations, and extreme statistics, while R focuses on the location statistics at various points along the distribution. It also tells you how many missing values there are, in this case 16. You will see they report almost identical means, max's and min's. If you'd like the standard deviation and other summary statistics in R, you can use simple commands. The R Code for standard deviation is

```
> sd(mydata$DY, na.rm=T)
      [1] 0.4760209
```

You see that the standard deviations are the same. You will also note that R does not like missing variables, so you have to tell it to compute the standard deviation in spite of the missing values, using the "not available" option, na.rm

`na.rm = T[rue]`. The default is `na.rm = F[alse]`.

But, R is more powerful than STATA. R will let you compute other location statistics very easily. That is, you can compute any quantile, the mean, or a trimmed mean. Also, R will let you compute different estimates of scale, other than the simple standard deviation/variance, such as median absolute deviations (i.e., the median of all absolute values of the deviations from the median. MAD is computed by first calculating the sample median, then for each value in the sample subtract the median from it, and the median among all those differences gives you an alternative to the standard deviation). Let's try all of these. For the median, simply type

```
> median(mydata$DY, na.rm=T)
      [1] 0.454473
```

The median absolute deviation in this case is very similar to the standard deviation

```
> mad(mydata$DY, na.rm=T)

       [1] 0.4475302
```

What about other points of location in the distribution, the quantiles. If I type in the the following command, I get the minimum (0%), maximum(100%), the 25th, 50th (or median) and 75th quantiles (Note: quantile is another way to say percentile)[8]

```
> quantile(mydata$DY,probs=seq(0,1,0.25),na.rm=T)

           0%       25%       50%       75%      100%
-1.111220  0.131653  0.454473  0.733686  1.659233
```

STATA will allow you to do this last step, simply by entering in the following command on the prompt, or by selecting on the menu of options

Statistics
Summary, tables & reports
Tables
Table of summary statistics (tabstat)

or type

```
tabstat dy, statistics( min p25 p50 p75 p90 max ) columns(variables)

       stats  dy

              min    -1.11122
              p25     .131653
              p50     .454473
              p75     .733686
              p90     .966824
              max    1.659233
```

Okay, so you can have plenty of fun with location statistics, so I won't dwell too much on them. Let's move to regressions.

[8] Note, the **probs=seq(0,1,0.25)**, option says in English: "on a 0 to 1 interval compute every 25th percentile". If you replace 0.25, with 0.2, you'll get the 0, 20th, 40th, 60th, 80th, and 100th percentiles.

Simple Regressions, and a Digression on Data Entry Problems

Okay, the data's in memory, so you can now just run regressions to your heart's content. For instance, let's just say that I want to estimate the following regression model, a simple log-linear version of the Solow Model

$$1) \quad \ln\left(\frac{y_{1985,i}}{y_{1960,i}}\right) = \beta_0 + \beta_1 \ln\left(y_{1960,i}\right) + \beta_2 \ln\left(n_i + g + \delta\right) + \beta_3 \ln\left(inv_i\right) + \varepsilon_i$$

where the left-hand side variable is the log-differential of gross domestic product between 1960 and 1985, which measures how different is GDP in 1985 from 1960. The Solow model would predict that: 1) the cross-sectional average GDP growth rate can be positive $\beta_0 > 0$, or negative $\beta_0 < 0$, 2) there should be convergence, such that higher initial income should mean lower growth rates, i.e., $\beta_1 < 0$, 3) population growth has a negative effect on growth, $\beta_2 < 0$, but let me point out that there is a minor inconsistency here, because usually when you have a log-log specification, your coefficient can be interpreted here as a percentage change, or elasticity. However, the n_i is already a population growth rate, so really, the coefficient should be interpreted as a percentage change in the population growth rate; the specification tells you something about acceleration in population growth, not population growth itself. Finally, higher investment rates (i.e., Investment/GDP) should lead to higher growth, $\beta_3 > 0$. Written this way, the basic Solow model says if you're either initially rich or have accelerating population growth, it's bad news for you (or as they say in the States "it sucks to be you"). The only good news for Solow is if you invest lots. Let's see if, at a first glance, Solow holds up, at least with this formulation.

R code:

```
> lm(DY~LGDP60+LNGD+LINV, data=mydata)

       Call:
       lm(formula = DY ~ LGDP60 + LNGD + LINV, data = mydata)

       Coefficients:
       (Intercept)       LGDP60         LNGD         LINV
            2.0143      -0.1830      -0.4248       0.6932
```

Or you could do the following to get more detail. First create an object called "**myreg**" and then use the **summary** command to get more detail.[9]

```
> myreg <- lm(DY~LGDP60+LNGD+LINV,data=mydata)
> summary(myreg)
```

[9] In R, you don't call a regression a regression, but rather a linear model, hence the acronym "lm". The first thing R asks is "what is the formula you want me to run?" So, your "dependent"/left-hand-side variable is income growth, DY, tilde/squiggle "~" separates that from the "independent"/right-hand side variables, and then since it's a linear model, you just put plusses in between the variables you would like estimated (I guess if you wanted to force a negative coefficient, you would just have a minus sign). Notice that to the right of the equation is the option "data=mydata".

```
Call:
lm(formula = DY ~ LGDP60 + LNGD + LINV, data = mydata)
Residuals:
      Min        1Q    Median        3Q       Max
-1.127428 -0.178513 -0.007347  0.188355  0.968636

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.01428    0.83695   2.407   0.0179 *
LGDP60      -0.18296    0.04164  -4.394 2.75e-05 ***
LNGD        -0.42477    0.26517  -1.602   0.1123
LINV         0.69322    0.08368   8.284 5.23e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3583 on 101 degrees of freedom
Multiple R-Squared: 0.4497,     Adjusted R-squared: 0.4333
F-statistic: 27.51 on 3 and 101 DF,  p-value: 4.368e-13
```

STATA translation:

```
regress dy lgdp60 lngd linv
```

```
Source   SS         df     MS           Number of obs = 105
                                        F(  3,   101) = 27.51
Model    10.5969361 3      3.53231204   Prob > F      = 0.0000
Residual 12.9690372 101    .128406309   R-squared     = 0.4497
                                        Adj R-squared = 0.4333
Total  23.5659733   104    .226595897   Root MSE      = .35834

dy     Coef.        Std. Err.     t      P>t    [95% Conf. Interval]

lgdp60 -.1829637    .0416373     -4.39  0.000  -.265561   -.1003664
lngd   -.4247712    .2651737     -1.60  0.112  -.9508046   .1012621
linv    .6932226    .0836818      8.28  0.000   .5272205   .8592247
_cons  2.014278     .8369527      2.41  0.018   .3539886  3.674567
```

Notice the coefficients and standard errors in R and STATA are the same; that's good to know. So, the intercept, suggests that on average between 1960 and 1985, holding everything else constant, log income doubled, since $\beta_0 = 2.014278$. Next, notice that there seems to be a sort of convergence, since $\beta_1 = \text{-}.1829637$ is negative. Next, there's the doom-and-gloom story of the acceleration of population growth, $\beta_2 = \text{-}.4247712$. Finally, there's the Solow good news, $\beta_3 = .6932226$. In sum

1. on average growth was positive, when other factors equal zero
2. there is convergence, since a percentage increase in initial income implies almost two-tenths of a percent lower growth rate
3. Malthus might have smiled to see the more than fourth-tenths of one percentage decrease in growth due to a percentage increase in population growth, and
4. Solow is happy because he would say, "see, I told you, a one percent increase in investment/GDP implies almost seven-tenths of one percent higher growth

Well, okay. At first glance, it appears almost like a Malthusian story, so with a smile, we can gladly conclude that it's almost completely wrong. It's time to look for something else. Mankiw, Romer and Weil say "wait a minute, there's something missing here: you forgot how much you liked school." They want to add the average years of completed schooling to the regression

2) $$\ln\left(\frac{y_{1985,i}}{y_{1960,i}}\right) = \beta_0 + \beta_1 \ln\left(y_{1960,i}\right) + \beta_2 \ln\left(n_i + g + \delta\right) + \beta_3 \ln\left(inv_i\right) + \beta_4 \ln\left(sch_i\right) + \varepsilon_i$$

As with investment, MRW think that $\beta_4 > 0$. Since the variable is already in the dataset, I create a new object "**myaugreg**" into which I will dump the new regression formula that requires just an additional term in R, **+LSCH**, and then presto-change-o

```
> myaugreg <- lm(DY~LGDP60+LNGD+LINV+LSCH,data=mydata)
> summary(myaugreg)

       Call:
       lm(formula = DY ~ LGDP60 + LNGD + LINV + LSCH, data = mydata)
       Residuals:
            Min       1Q   Median       3Q      Max
       -0.96074 -0.18731  0.00183  0.19687  0.92485

       Coefficients:
                   Estimate Std. Error t value       Pr(>|t|)
       (Intercept)  3.11285    0.85045   3.660       0.000406 ***
       LGDP60      -0.29732    0.04997  -5.950       4.07e-08 ***
       LNGD        -0.50668    0.25232  -2.008       0.047362 *
       LINV         0.55286    0.08770   6.304       8.12e-09 ***
       LSCH         0.21645    0.05928   3.651       0.000419 ***
       ---
       Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

       Residual standard error: 0.3388 on 99 degrees of freedom
       Multiple R-Squared: 0.516,      Adjusted R-squared: 0.4964
       F-statistic: 26.38 on 4 and 99 DF,  p-value: 6.696e-15
```

I'll get to STATA, but let me interpret the changes first, and later confirm if STATA gives me the same thing. Note the sample falls from 105 to 104 because one country (looks like Guinea) has no education data. So now, on average log income tripled during that period, the convergence rate is higher, the elasticity of population growth is stronger, the effect of investment is weaker than before, and average years of schooling do have a positive effect on growth. At first glance, Malthus, Solow, and MRW would be happy (if you read Temple's paper or the working paper I wrote, and you'll see how to make Malthus, Solow, and MRW unhappy). Now let's try to run the regression in STATA. When I run

```
regress dy lgdp60 lngd linv lsch

       no observations
```

unlike before, I don't get table of output. That's funny. So, the first thing I do is type in the "`browse`" or "`edit`" command in STATA to look at the data. What I notice is that when STATA imported the file, which is not a problem with R, that some of the cells were interpreted as blank character cells, when in fact they should have been filled with a period representing a missing value. So at this point R laughs at STATA: "haha, you can't even read in the data correctly". So, we'll have to coax STATA into seeing things properly. Automatically, you can interpret STATA as saying "I'm reading this variable as a character/string variable, so if you don't agree, help me out." As proof, simply type the command[10]

[10] Again the underline on "`des`" implies that to use the command you only need the first three letters.

describe

```
country         str15   %15s
mrwno           byte    %8.0g   MRWNO
nsam            byte    %8.0g   NSAM
isam            byte    %8.0g   ISAM
osam            byte    %8.0g   OSAM
africa          byte    %8.0g   AFRICA
latinca         byte    %8.0g   LATINCA
eastasia        byte    %8.0g   EASTASIA
indust          byte    %8.0g   INDUST
gdp60           long    %12.0g  GDP60
gdp85           int     %8.0g   GDP85
gdpg            float   %9.0g   GDPG
popn            float   %9.0g   POPN
inv             float   %9.0g   INV
school          float   %9.0g   SCHOOL
lngd            float   %9.0g   LNGD
linv            float   %9.0g   LINV
lsch            str8    %9s     LSCH
lgdp60          float   %9.0g   LGDP60
lgdp85          str8    %9s     LGDP85
dy              float   %9.0g   DY
```

Look at the two lines that I have highlighted with bold print, the line for "**lsch**", natural log of schooling, and also the line for "**lgdp85**", natural log of GDP in 1985. The two middle columns suggest that the variables are being read as eight-character string variables (**str8**). So, the first thing you might try is to fill in those missing values in STATA, by hand, by opening up the editor. I do so, and as I'm doing this, STATA is spitting out in it's own language what I just did to the data, hence when I close the data editor, you'll notice the following commands on the screen

```
- replace lsch = "." in 16
- replace lgdp85 = "." in 14
- replace lgdp85 = "." in 36
- replace lgdp85 = "." in 44
- replace lgdp85 = "." in 45
- replace lgdp85 = "." in 66
- replace lsch = "." in 66
- replace lsch = "." in 68
- replace lsch = "." in 72
- replace lgdp85 = "." in 72
- replace lgdp85 = "." in 78
- replace lgdp85 = "." in 81
- replace lgdp85 = "." in 82
- replace lgdp85 = "." in 91
- replace lgdp85 = "." in 111
- replace lgdp85 = "." in 114
- replace lgdp85 = "." in 118
```

You could use these codes again, by copying and pasting them into a do-file, AS LONG AS you do not change the order of the data. However, STATA won't accept these codes as is, so if you are using a do-file, you could copy and paste these codes into the do file, and then type "crtl" + "h", and paste "- " in the "find" command, and leave the "replace" space empty. When you hit enter, it will remove all of the "- " and replace it with nothing, saving you lot's of trouble. This might seem like overkill for a few lines, but wait until you have to do this 200 times, or more. Ouch! Using "crtl + h" in this way will produce the following codes which you can reuse in STATA anytime

```
replace lsch = "." in 16
replace lgdp85 = "." in 14
replace lgdp85 = "." in 36
replace lgdp85 = "." in 44
replace lgdp85 = "." in 45
replace lgdp85 = "." in 66
replace lsch = "." in 66
replace lsch = "." in 68
replace lsch = "." in 72
replace lgdp85 = "." in 72
replace lgdp85 = "." in 78
replace lgdp85 = "." in 81
replace lgdp85 = "." in 82
replace lgdp85 = "." in 91
replace lgdp85 = "." in 111
replace lgdp85 = "." in 114
replace lgdp85 = "." in 118
```

After I did this, I still can't seem to read in the data properly. So, let me try something else. Hmm, maybe I can simply recreate the variables from Temple's file (but I'll temporarily rename the ones I'll drop so that I can double-check.)

```
rename lsch lsch2
rename lgdp85 lgdp852
generate lsch=ln(school)
generate lgdp85=ln(gdp85)
```

I think I have what looks right. So, then I try to run the regression again

```
regress dy lgdp60 lngd linv lsch
```

```
      Source   SS          df     MS            Number of obs = 104
                                                      F(  4,    99) = 26.38
      Model    12.1132343  4      3.02830858    Prob > F = 0.0000
      Residual 11.364106   99     .11478895     R-squared = 0.5160
                                                 Adj R-squared = 0.4964
      Total         23.4773403    103    .227935343    Root MSE = .33881

      dy     Coef.         Std. Err.     t      P>t    [95% Conf. Interval]

      lgdp60 -.2973202     .0499732     -5.95  0.000   -.396478  -.1981625
      lngd   -.5066783     .2523226     -2.01  0.047  -1.007341  -.0060155
      linv    .5528552     .0877002      6.30  0.000    .378839   .7268714
      lsch    .2164498     .0592838      3.65  0.000    .0988179  .3340817
      _cons  2.116063      .7930588      2.67  0.009    .542462  3.689663
```

Now comparing with what we got in R, everything looks right, EXCEPT the intercept. So, then I look back at the data, using "`browse`" or "`edit`", [actually I did this first, and you should do this first too] and you notice that the **`lsch`** and **`lsch2`** are NOT the same. You can verify by typing in the STATA command prompt

```
edit country school lsch lsch2 lgdp85 lgdp852
```

Since STATA is still not reading in **`lsch2`** as numeric, we can't just ask STATA why they're not the same. The first things that come to mind to fix this are: 1) copy the two series into EXCEL, and 2) look at what R has listed. I'll show you the latter, but get used to using EXCEL to check your data because it's usually quicker. Since we know that **`mydata$LSCH`** in R is supposed to be the natural log of some variable, the way to transform the data back to its supposed original form is to compute the

exponential of the natural log. So, to transform back R's version of **mydata$LSCH** use the following command

```
> exp(mydata$LSCH)
```

What you see is that **exp(mydata$LSCH)** looks like **mydata$SCHOOL** but divided by 100. So it looks like Temple does this because it's school enrolment rates. Now we can go back to STATA and first drop the last version of **lsch** and recreate it using the following commands

```
drop lsch
generate lsch=ln(school/100)
```

Having done that we simply rerun the codes again in STATA, and you get

```
reg dy lgdp60 lngd linv lsch
```

```
      Source       SS           df      MS           Number of obs = 104
                                                     F(  4,    99) = 26.38
      Model        12.1132343    4      3.02830857   Prob > F = 0.0000
      Residual     11.3641061    99     .11478895    R-squared = 0.5160
                                                     Adj R-squared = 0.4964
      Total        23.4773403    103    .227935343   Root MSE = .33881

      dy     Coef.         Std. Err.     t      P>t    [95% Conf. Interval]

      lgdp60 -.2973202     .0499732     -5.95  0.000   -.396478  -.1981625
      lngd   -.5066783     .2523226     -2.01  0.047  -1.007341  -.0060155
      linv    .5528552     .0877002      6.30  0.000    .378839   .7268714
      lsch    .2164498     .0592838      3.65  0.000    .0988179  .3340817
      _cons  3.112851      .8504456      3.66  0.000   1.425382  4.800319
```

Now, it looks just like R. So, you can drop the two old variables

```
drop lsch2 lgdp852
```

Replicating Other Peoples' Work

So, in case you weren't aware, there's a real problem in economics research: you often can't replicate other peoples' work. The "irreproducibility" of economics research was first documented in a 1986 article in the American Economic Review, based on a project funded by the Journal of Money, Credit and Banking (see Dewald, et al. (1986)).

In one of the econometrics classes I took as a graduate student, one of the assignments was to replicate a paper, in line with the Dewald, et al. (1986) study. My results were close in some cases, but not in others. By replication, I mean: can you read the paper, follow their steps to estimation, presuming you are able to get the data they used (sometimes people give you the run-around at this stage), and can you get the same results that they get? A perfect example of one the problems in applied research is when you see in the data citations "taken from various sources," or something to that effect. So be diligent in avoiding that and other ambiguities, whether you intended them or not.

In the newspapers, you will sometimes read of one controlled laboratory experiment in the hard sciences being replicated (or not) by people in another laboratory. I even heard one prominent economist whine that another prominent economist's experimental work could not be replicated, and that the data were not freely available. That's why I encourage you to distribute freely any codes you may develop on your own, as well as the data (unless it's confidential, or perhaps proprietary) to anyone who asks. As long as you're not trying to prove that demand curves don't slope downward, welcome the opportunity to discover errors in your own work in your search for truth. Often the only way knowledge advances is because someone was bold enough to try something new and different, and trial and the discovery of error is part of that process. My dissertation advisor, Joe Reid, used to say that the way to do research is to combine technique with a minimum amount of creativity. The creativity was his explanation for the success of the "Chicago School" in having so many Nobel Laureates. I'm convinced.

So, in this case, I'll try to replicate some of the regressions in Mankiw, Romer, and Weil (1992). Specifically, I'll try to replicate the coefficients reported in Table IV on page 426. First, notice that R and STATA are producing basically the same results. Second, notice that the smaller the sample size the greater the difference between the results in the MRW paper and what is reported here.[11] In this case, I would not attribute it to any intended deceit. The most likely explanation is the fact that the paper appears in print in 1992, and computers have come a long way since then (the internet probably wasn't even available to the public when they did the actual estimation!). So this is more-or-less reproducible. Still, reproducible should be taken literally: either it is or it is not.

[11] The three samples as described on page 413 are the 98 countries in which oil is not a dominant sector, the 75 countries whose data quality is judged to be above the unsatisfactory grade of "D" by Summers & Heston (1988), and the sample of OECD countries.

The Non-Oil Sample

R Code:

```
> myregnsam <- lm(DY~LGDP60+LNGD+LINV,data=subset(mydata,mydata$NSAM == 1))
> summary(myregnsam)

       Call:
       lm(formula = DY ~ LGDP60 + LNGD + LINV, data = subset(mydata,
           mydata$NSAM == 1))

       Residuals:
            Min       1Q   Median       3Q      Max
       -1.07648 -0.15215  0.01185  0.19595  0.96056

       Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
       (Intercept)  1.91937    0.83368   2.302  0.02353 *
       LGDP60      -0.14090    0.05202  -2.709  0.00803 **
       LNGD        -0.30235    0.30439  -0.993  0.32311
       LINV         0.64724    0.08670   7.465 4.16e-11 ***
       ---
       Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

       Residual standard error: 0.3507 on 94 degrees of freedom
       Multiple R-Squared: 0.4019,     Adjusted R-squared: 0.3828
       F-statistic: 21.05 on 3 and 94 DF,  p-value: 1.622e-10
```

STATA Translation:

```
regress dy lgdp60 lngd linv if nsam==1

       Source       SS           df     MS           Number of obs = 98
                                                     F(  3,    94) = 21.05
       Model        7.76763568   3      2.58921189   Prob > F = 0.0000
       Residual     11.5611668   94     .122991136   R-squared = 0.4019
                                                     Adj R-squared = 0.3828
       Total        19.3288025   97     .199266005   Root MSE = .3507

       dy     Coef.         Std. Err.     t      P>t    [95% Conf. Interval]

       lgdp60 -.1409009     .0520183     -2.71  0.008  -.2441845  -.0376173
       lngd   -.3023473     .3043856     -0.99  0.323  -.9067121   .3020174
       linv    .6472379     .0866999      7.47  0.000   .4750932   .8193827
       _cons  1.919375      .8336781      2.30  0.024   .2640873  3.574662
```

The Intermediate Sample

R Code:

```
> myregisam <- lm(DY~LGDP60+LNGD+LINV,data=subset(mydata,mydata$ISAM == 1))
> summary(myregisam)

Call:
lm(formula = DY ~ LGDP60 + LNGD + LINV, data = subset(mydata,
    mydata$ISAM == 1))

Residuals:
       Min         1Q     Median         3Q        Max
-1.176e+00 -1.733e-01  6.905e-05  1.467e-01  9.347e-01

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.24967    0.85473   2.632 0.010406 *
LGDP60      -0.22783    0.05725  -3.980 0.000165 ***
LNGD        -0.45746    0.30743  -1.488 0.141177
LINV         0.64587    0.10392   6.215 3.11e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3258 on 71 degrees of freedom
Multiple R-Squared: 0.3788,     Adjusted R-squared: 0.3526
F-statistic: 14.43 on 3 and 71 DF,  p-value: 1.950e-07
```

STATA translation:

```
regress dy lgdp60 lngd linv if isam==1

      Source |       SS           df       MS      Number of obs =      75
-------------+----------------------------------   F(  3,    71) =   14.43
       Model |  4.59631642         3  1.53210547   Prob > F      =  0.0000
    Residual |  7.53685483        71  .106152885   R-squared     =  0.3788
-------------+----------------------------------   Adj R-squared =  0.3526
       Total |  12.1331713        74  .163961774   Root MSE      =  .32581

------------------------------------------------------------------------------
          dy |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      lgdp60 |  -.2278295   .0572506    -3.98   0.000    -.3419838   -.1136751
        lngd |  -.4574575   .3074298    -1.49   0.141    -1.070455    .1555401
        linv |   .6458655   .1039176     6.22   0.000     .4386598    .8530713
       _cons |   2.249667   .854728      2.63   0.010     .5453879    3.953947
------------------------------------------------------------------------------
```

The OECD Sample
R Code:

```
> myregosam <- lm(DY~LGDP60+LNGD+LINV,data=subset(mydata,mydata$OSAM == 1))
> summary(myregosam)

Call:
lm(formula = DY ~ LGDP60 + LNGD + LINV, data = subset(mydata,
    mydata$OSAM == 1))

Residuals:
     Min       1Q   Median       3Q      Max
-0.18289 -0.11598 -0.03232  0.06111  0.36293

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.14032    1.18069   1.813   0.0866 .
LGDP60      -0.34991    0.06574  -5.322 4.65e-05 ***
LNGD        -0.76626    0.34524  -2.220   0.0395 *
LINV         0.39010    0.17612   2.215   0.0399 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1529 on 18 degrees of freedom
Multiple R-Squared: 0.6767,     Adjusted R-squared: 0.6228
F-statistic: 12.56 on 3 and 18 DF,  p-value: 0.0001145
```

STATA translation:

```
regress dy lgdp60 lngd linv if osam==1

      Source       SS           df       MS           Number of obs = 22
                                                      F(  3,    18) = 12.56
       Model       .880514816    3      .293504939    Prob > F = 0.0001
    Residual       .420714816    18     .023373045    R-squared = 0.6767
                                                      Adj R-squared = 0.6228
       Total       1.30122963    21     .061963316    Root MSE = .15288

------------------------------------------------------------------------------
          dy     Coef.         Std. Err.      t      P>t     [95% Conf. Interval]
-------------------------------------------------------------------------------
      lgdp60   -.3499109     .0657429     -5.32   0.000    -.4880315   -.2117903
        lngd   -.7662572     .3452362     -2.22   0.040    -1.491572   -.0409428
        linv    .3900988     .1761202      2.21   0.040     .020084     .7601137
       _cons    2.14032      1.180692      1.81   0.087    -.3402218    4.620861
------------------------------------------------------------------------------
```

Basic Statistical Graphics

I think R is better than STATA for statistical graphics, simply because creators of R based the grammar, and hence the program's capabilities of expression, on S-Plus, and S-Plus is by far the best package for statistical graphics I've seen. Although, statisticians may even have preferences for programs I'm not aware of. However, STATA has come a long way since version 7.0. STATA, in my opinion is now second best for graphics. You'll see below that sometimes I find STATA's graphics seem less accurate than R's, but the story is more or less the same. If you work with STATA version 8 or 9, you will see that everything is now menu driven, which is great, because you can do your first graph simply by fiddling with the menu options, and then if you need to replicate it in the future, you can "copy" + "paste" the commands in a do-file for future reference that STATA spits out on the screen each time you submit a request to create a graph.

Okay, so let's return to estimating equation 2), the Augmented Solow model around the steady-state. One of the first things you should do is to look at the residuals from the Augmented Solow Model OLS regression. You might think of doing a historgram, perhaps overlaid with a normal density plot. This can be done in R, but again, it requires lots of knowledge about how to give R the exact details it wants to do exactly what you want it to do. To compare, look below at the lines of code required to do this plot, versus what is required in STATA. Although R keeps the augmented Solow Model regression output in it's memory, I will reproduce it so that you won't have to flip back to remind yourself of the command, so that you can create the residuals as follows

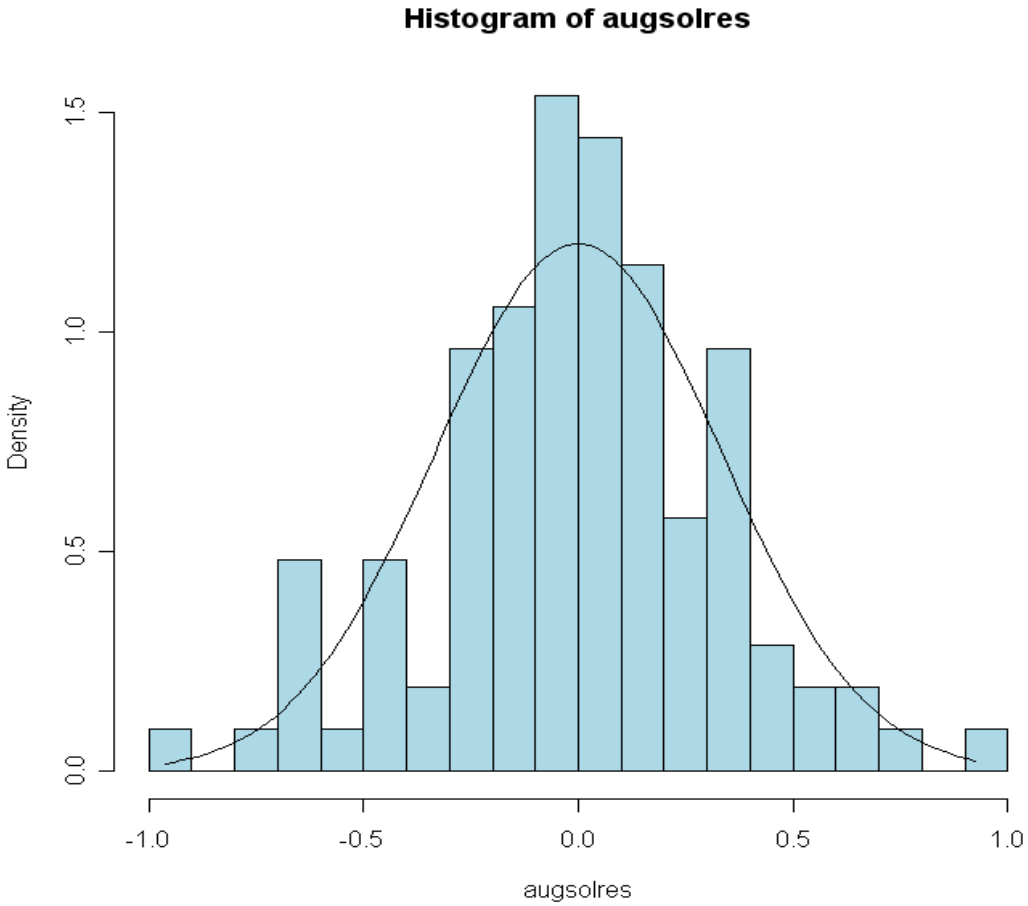
R Code:

```
> myaugreg <- lm(DY~LGDP60+LNGD+LINV+LSCH,data=mydata)
> augsolres <- myaugreg$residuals
```

A historgram for these residuals with a Normal density plot laid over it is produced in Figure 1 below. The first line creates a histogram with 20 bins, the second line creates, the object over which the normal density line will be created, the third line creates the normal density object based on the new.x specified in the second line, with the min , and the final line plots it. I got this from the S-Plus 2000 manual (see p. 473), just to tell you how similar R and S-Plus are.

```
> hist(augsolres,br=20,freq = FALSE, col='light blue')
> new.x <- seq(min(augsolres), max(augsolres), length=length(augsolres))
> new.dens <- dnorm(new.x, mean = mean(augsolres), sd=sd(augsolres))
> lines(new.x, new.dens)
```

Figure 1. Histogram and Normal Density Plot of Augmented Solow Model OLS Residuals in R



You can change the color, but I chose North Carolina Blue, i.e., 'light blue'. In STATA, they do everything for you, so the commands are simpler, but you also have less control over the plots. In STATA, however, its memory is almost as short as the last command you ran. So, unless you just estimated the regression, to create the residuals, you will have to re-estimate the regression, and then create the residuals

STATA Translation:

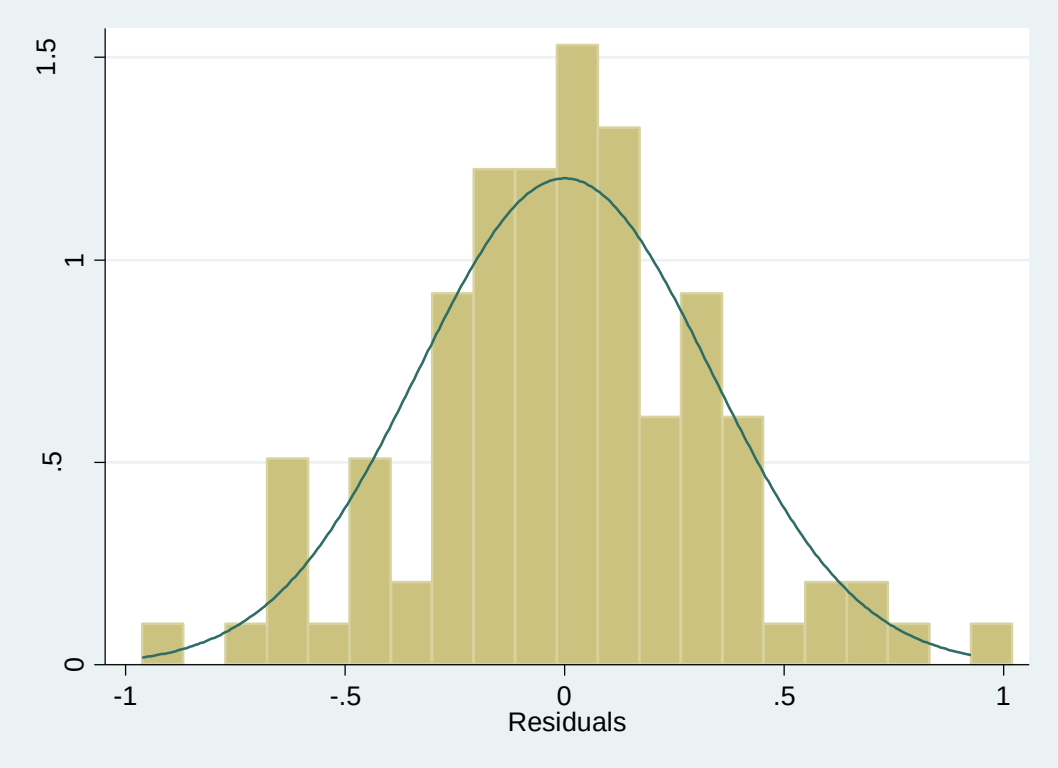
```
reg dy lgdp60 lngd linv lsch
predict augsolres, res
```

This will create a new variable in your dataset called **augsolres**. You create a histogram with 20 bins and a normal density plot using

```
histogram augsolres, bin(20) normal
```

You'll notice that the process of assigning observations to bins for each bar in the histogram seems to differ between R and STATA, hence the histograms look slightly different in the center of the distribution.

Figure 2. Histogram and Normal Density Plot of Augmented Solow Model OLS Residuals in STATA



Having done the plots, now comes the warning. If you want to look at the distribution of something, or compare it with a theoretical distribution, DON'T use the histogram with a normal density plot. I quote from the R help menu for the histogram command[12]

```
## Comparing data with a model distribution should be done with qqplot()!
     qqplot(x, qchisq(ppoints(x), df = 4)); abline(0,1, col = 2, lty = 2)

## if you really insist on using hist() ... :
     x <- rchisq(100, df = 4)
     hist(x, freq = FALSE, ylim = c(0, 0.2))
     curve(dchisq(x, df = 4), col = 2, lty = 2, lwd = 2, add = TRUE)
```

I actually tried that suggestion before I found the S-Plus commands above, but it didn't seem to work when I replaced the Chi-squared with the normal. So, since a histogram with normal density plot is such a BAD idea, forget about that and consider a better way to compare two univariate distributions using quantiles (quantile being another word for percentile). The logic behind this is that the data are first sorted from largest to smallest. You then assign a ranking to that data point, and for each of the *i* points in the sample, a probability point is computed as follows

3) $$p_i = \frac{i-a}{m+1-2a}$$

[12] In fact, I tried this command in R but after replacing rchisq and dchisq with rnorm and dnorm, it didn't work. So, I then went looking to see if I couldn't find the proper way to do this in R in the S-Plus 2000 manual, and I found something that works. So that's a hint that you have an even greater number of references to check beyond just the R documentation.

where *i* is the point, *a* is a parameter that is used in plotting, and *m* is the sample size. The logic of what it does to the sample is not obvious right away, so an example can help. First draw a unit normal random variable (i.e., it has mean zero, standard deviation equal to one). Before that, set the seed so that you can reproduce this result as often as you'd like, and then draw the pseudo-random sample

```
> set.seed(91)
> rs <- rnorm(15)
> cbind(sort(rs))

        [1,] -0.96037627
        [2,] -0.90704826
        [3,] -0.85393422
        [4,] -0.85348024
        [5,] -0.66227141
        [6,] -0.31626267
        [7,] -0.11384293
        [8,] -0.04062524
        [9,]  0.02152212
       [10,]  0.04456292
       [11,]  0.21770639
       [12,]  0.51552953
       [13,]  1.84387598
       [14,]  2.03309294
       [15,]  2.47094468
```

I used **`cbind()`** in front of **`sort()`** to display the output as a column and show you the ranks in brackets. Now, this is my random sample, and the numbers in brackets would be the *i*'s in equation 3). If you have a small sample, less than or equal to 10, then you should use $a = 0.375$, but we have 15, so $a = 0.5$. If you plug these values into equation 3), you get the probability points of the empirical distribution using

4) $$p_i = \frac{i - 0.5}{m + 1 - 2 \cdot (0.5)} = \frac{i - 0.5}{15}$$

If you plug in the ranks from 1 to 15 into equation 3), you get what's reported in the second column below [try it out: `(1-0.5)/15`, `(2-0.5)/15`, ... `(15-0.5)/15`]

```
> cbind("Random Sample"=sort(rs),"Percentiles"=ppoints(rs))
      Random Sample Percentiles
 [1,]   -0.96037627  0.03333333
 [2,]   -0.90704826  0.10000000
 [3,]   -0.85393422  0.16666667
 [4,]   -0.85348024  0.23333333
 [5,]   -0.66227141  0.30000000
 [6,]   -0.31626267  0.36666667
 [7,]   -0.11384293  0.43333333
 [8,]   -0.04062524  0.50000000
 [9,]    0.02152212  0.56666667
[10,]    0.04456292  0.63333333
[11,]    0.21770639  0.70000000
[12,]    0.51552953  0.76666667
[13,]    1.84387598  0.83333333
[14,]    2.03309294  0.90000000
[15,]    2.47094468  0.96666667
```

Verify for yourself, that, minus the labels, the quantile-quantile normal plot with a reference line that has an intercept equal to the mean of the sample, and a slope equal to the standard deviation of the sample can be reproduced with the following codes

```
> plot(qnorm(ppoints(15),mean=mean(rs),sd=sd(rs)),sort(rs))
> abline(0,1)
```

and that this is equivalent to the "canned" command

```
> qqnorm(rs)
> abline(mean(rs),sd(rs))
```

[However, I have to tell you that I'm not sure why it is equivalent to the line with an intercept equal to zero and slope equal to one]. Note that there is usually a **`qqnorm`** reference line called **`qqline`**. I believe S-Plus, R's commercial software older brother, uses a command equivalent to **`abline(mean(rs),sd(rs))`**, but the R command, **`qqline`**, adds a reference line that goes through the first and third quartiles (i.e., the 25^{th} and 75^{th} percentiles). So, in R, just use their "canned" commands, which are good enough

```
> qqnorm(rs)
> qqline(rs)
```

With a basic understanding of how quantiles and qq-plots work, you can apply it to the residuals from the Augmented Solow Model, to produce Figure 3

```
> qqnorm(augsolres)
> qqline(augsolres)
```

On the left, if the dots are below (above) the line, then the data is heavier (lighter) tailed on the left-side of the distribution, and on the right, if the dots are above (below) the line, then the data is heavier (lighter) tailed on the right side of the distribution. Later I'll add in text in the R plot as a reminder of what's heavy-tailed and light-tailed in a qq-plot.

The STATA command "**`qnorm`**", produces Figure 4:[13]

```
qnorm augsolres
```

[13] The "**`qnorm`**" command is confusing, and some statisticians might scream in outrage if they saw this. S-Plus makes a distinction between a quantile-normal plot, called "**`qnorm`**" and a quantile-quantile normal distribution plot, called "**`qqnorm`**" as we just saw. The latter is just what we have just done, while the former, is just a plot of the probability points for the normal distribution. You will see the former in many older applied statistics papers, like Fama (1965). The reason why you no longer see them is that like the histogram, they are harder to read when making a comparison between two distributions (i.e., it is easier to compare something against a straight line than a curved line; quantile-quantile normal plots make our life easy).

Figure 3. R Quantile-Quantile Normal Plot: Augmented Solow Model OLS Residuals

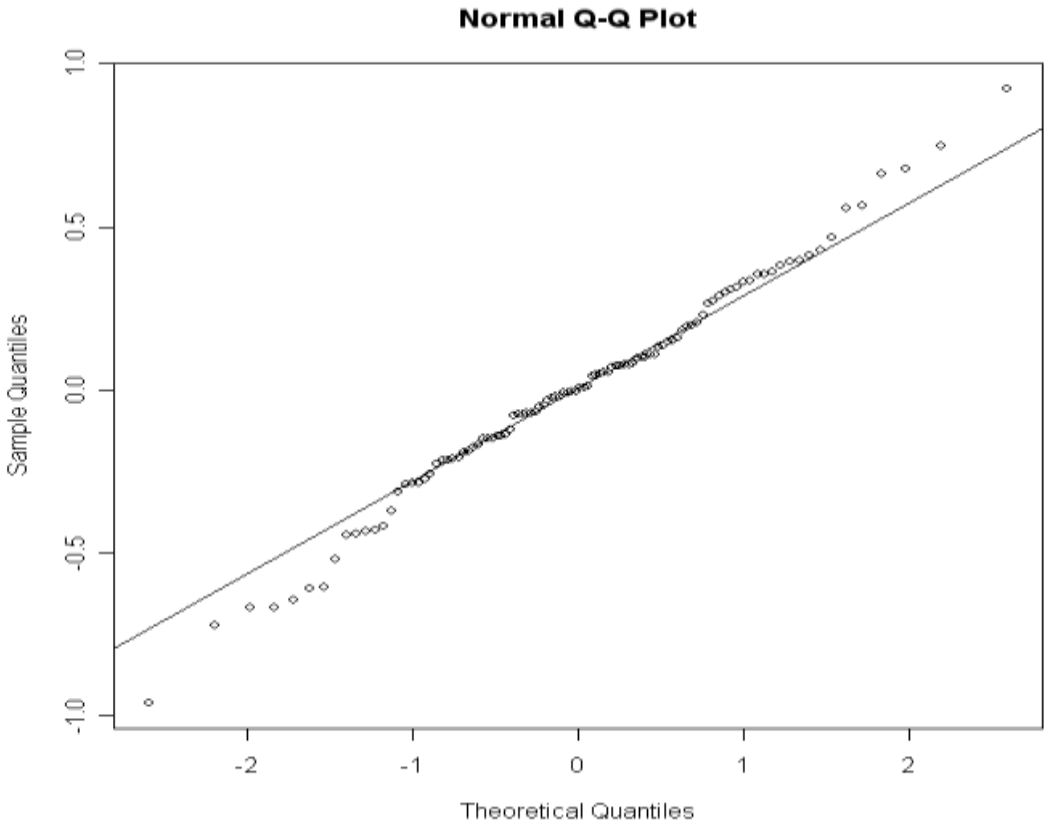
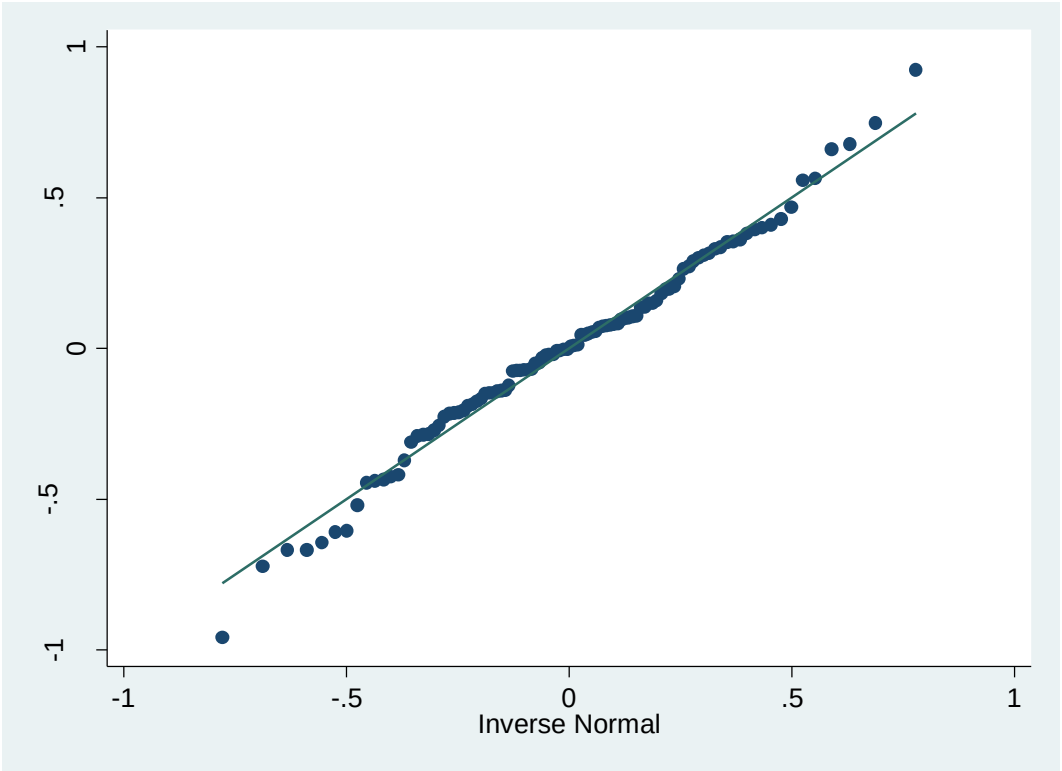


Figure 4. STATA Quantile-Quantile Normal Plot: Augmented Solow Model OLS Residuals



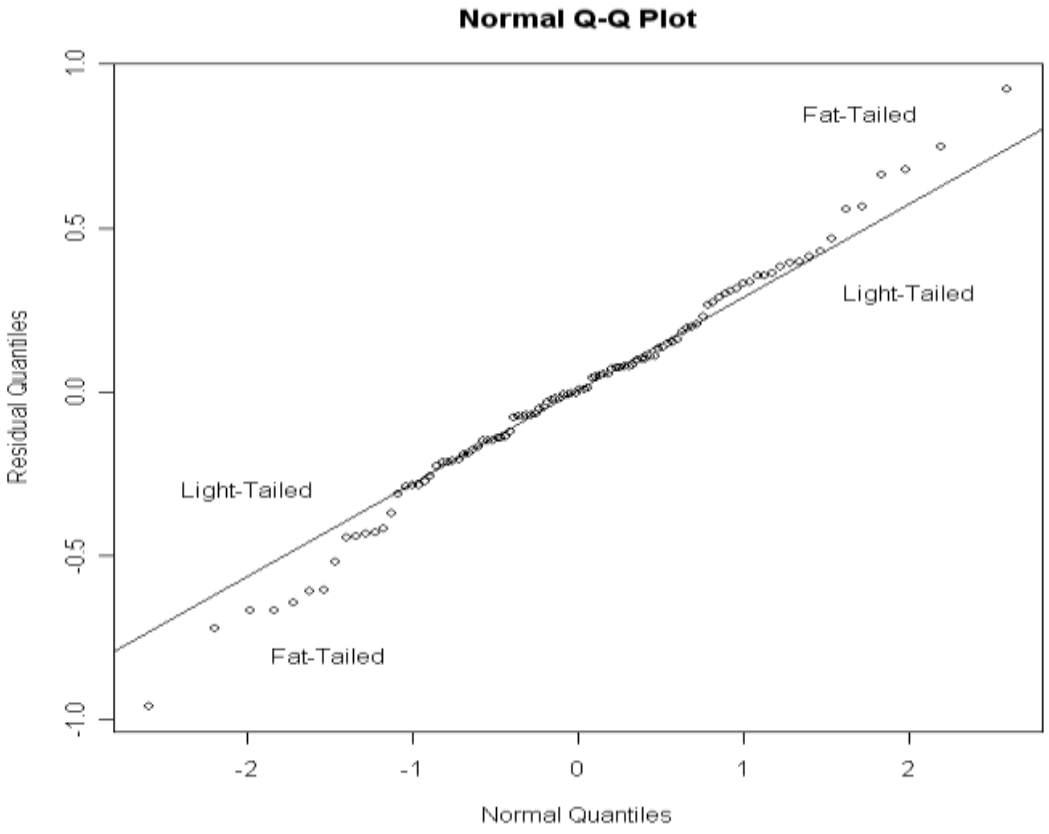
Slightly Fancier Graphs with More Control Over Graph Options

Now I can demonstrate how R trumps STATA. It gives you way more control over your environment (the cost is acquiring the ability to think like a programmer). This extends beyond graphics too, but for now here's an illustration. I'd like to change the default titles, and labels, and also add text to the plot. R will let me add text, STATA will not (although STATA allows you to label points). Running the following gives

R Code:

```
> qqnorm(augsolres,main="Normal Q-Q Plot",xlab="Normal
Quantiles",ylab="Residual Quantiles")
> qqline(augsolres)
> text(-1.5,-0.8,"Fat-Tailed")
> text(-2,-0.3,"Light-Tailed")
> text(2,0.3,"Light-Tailed")
> text(1.7,0.85,"Fat-Tailed")
```

Figure 5. Quantile-Quantile Normal Plot of Augmented Solow Model OLS Residuals in R with Custom Axis Labels, Title and Text Overlaid on Plot

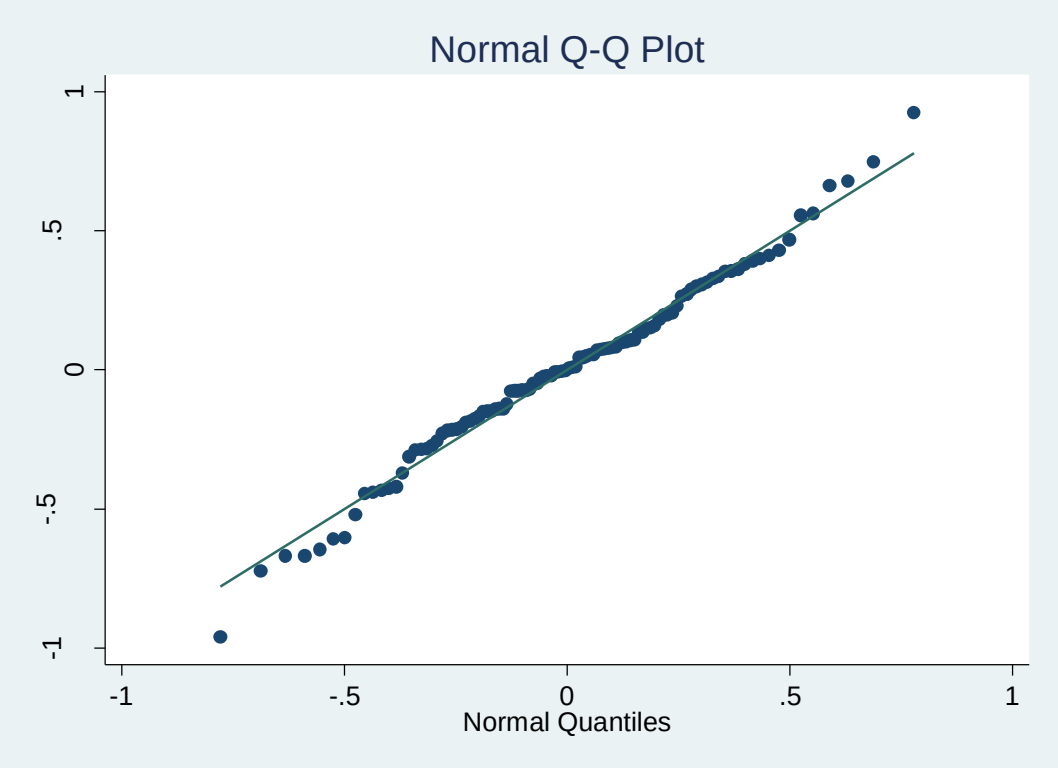


The last four lines I put in to help remind you when the data is heavy-tailed and when it's light-tailed. Through trial and error, I figured out the proper location for the text. In STATA, sometimes you're left guessing if you forget what it does, because the help files don't help here. The STATA command without the reminder about which points are light- and heavy-tailed is

STATA Translation:

```
qnorm augsolres, ytitle("Residual Quantiles") xtitle("Normal Quantiles")
title("Normal Q-Q Plot")
```

Figure 6. Quantile-Quantile Normal Plot of Augmented Solow Model OLS Residuals in STATA, with Custom Axis Labels, Title but without Text Overlaid on Plot



More Advanced Graphics

Okay, so if you look again at the "Visualizing Growth" working paper, you'll notice that I wrote some codes in the appendix for S-Plus, and I can, with minor alterations apply them in R, to generate the graphs. The first thing to do is to download and install the `lattice` package, which is R's equivalent of S-Plus's Trellis graphics environment. To do this, as I alluded to earlier, R first wants you to select from the package menu

Packages
Install package(s)…
lattice

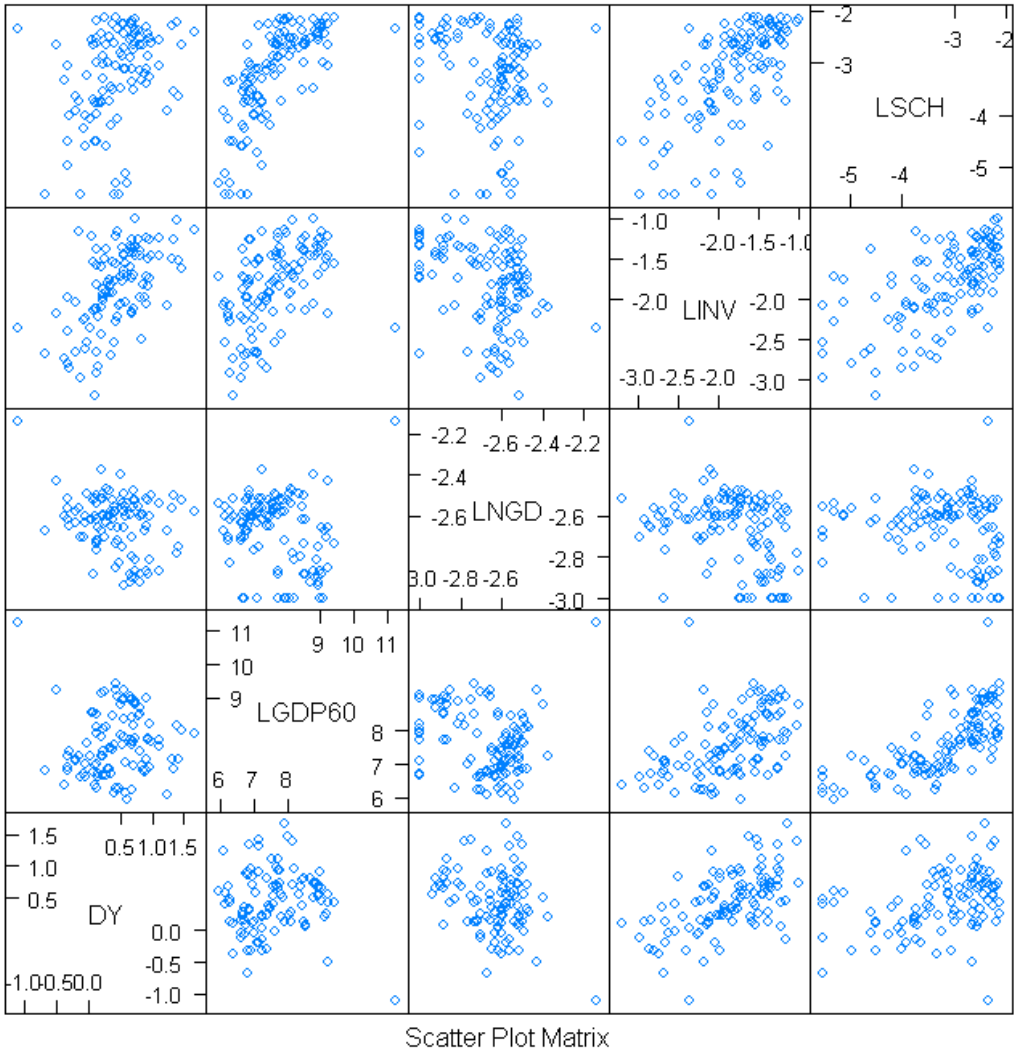
Then you can load the lattice package by going back to the menu as selecting

Packages
Load package…
lattice

First, consider the scatterplot matrix. It is like visual representation of what a regression does, except that it does not remove the partial effects of what would be the independent variables. You could remove the partial effects by hand though, by estimating the residuals from regressions among the independent variables.

```
> mydata <- read.table("C:/temple.csv",header=T,sep=",")
> mymatrix <-
cbind(mydata$DY,mydata$LGDP60,mydata$LNGD,mydata$LINV,mydata$LSCH)
> splom(~mymatrix, varnames = c("DY","LGDP60","LNGD", "LINV","LSCH"))
```

Figure 7. Scatterplot Matrix of Basic Solow Model in R



The way to read the scatterplot matrix is to look at each scatterplot within the matrix individually. Once you focus on an individual panel, the way to read the axes is as follows. To know what label to use for the horizontal axis, simply look up or down the column to find the label, while for the vertical axis, simply look left or right to find the label.

Spotting the Multicollinearity

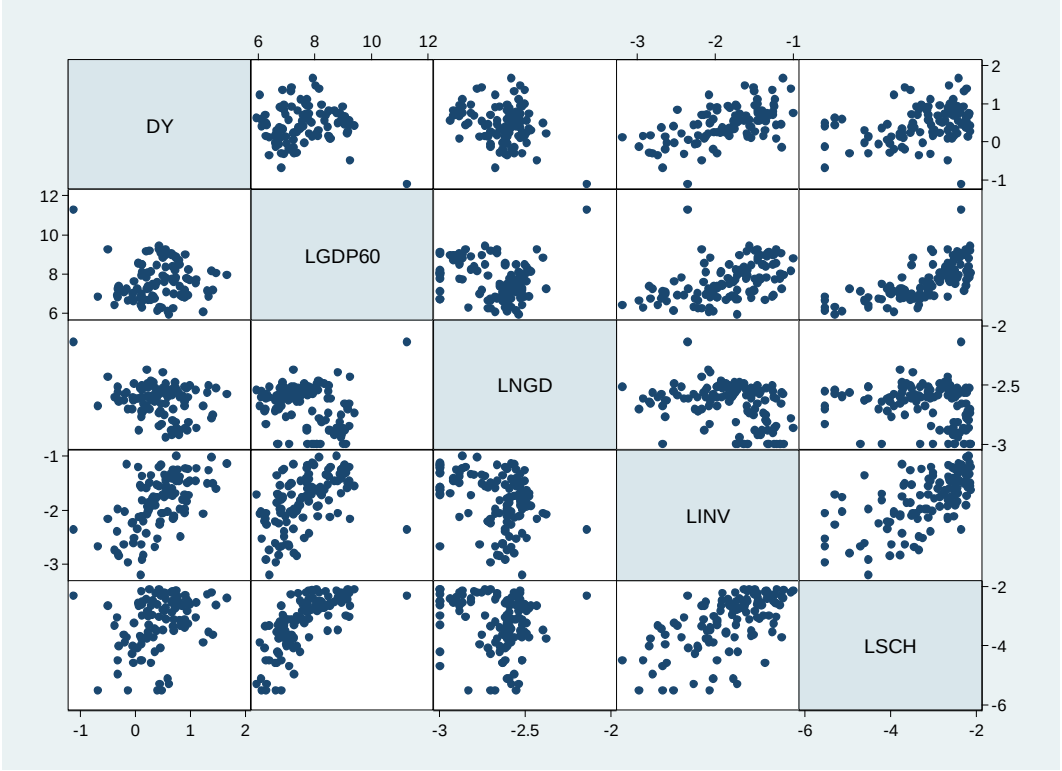
If you look at the panels for what I have until now been referring to as independent variables LGDP60, LNGD, LINV, LSCH, you see that there are correlations between LINV and LSCH, and also for LGDP60 and LINV and LGDP60 and LSCH. That will make it difficult to interpret the marginal impact of those variables.

The STATA equivalent can be generated as follows

```
drop lsch
generate LSCH=ln(school/100)
graph matrix dy lgdp60 lngd linv LSCH
```

I added the line **`drop lsch`**, and capitalized the variable name, because when I run the scatterplot matrix the other variables appears capitalized, while **`lsch`** appears in lowercase. Notice that the labels are read from top left to bottom right in STATA, whereas in R the labels go from bottom left to top right.

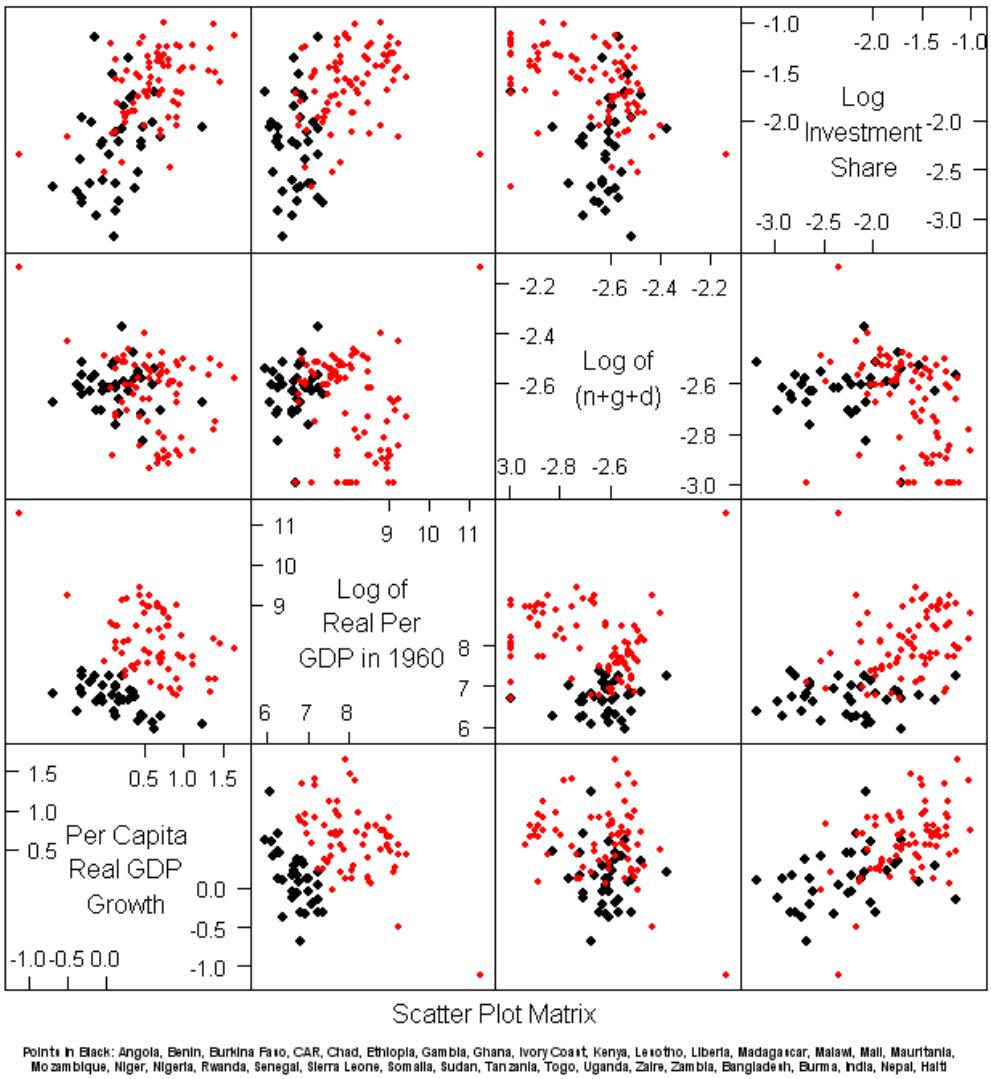
Figure 8. Scatterplot Matrix of Basic Solow Model in STATA



I can make a slightly fancier version of this scatterplot matrix in R with more text. I can also select certain points within the graph as follows. You could do this in STATA, but only panel by panel, using the overlaid plot command. I'll drop the schooling variable so that the text fits nicely in the boxes.

```
> solowmatrix <- cbind(mydata$DY,mydata$LGDP60,mydata$LNGD,mydata$LINV)
> splom( ~ solowmatrix, varnames = c("Per Capita\n\Real GDP\n\Growth","Log
of\n\Real Per\n\GDP in 1960","Log of\n\(n+g+d)","Log\n\Investment\n\Share"),
panel = function(x, y){
i <- c(2,3,5,6,8,9,12,14:24,27:33,35,37,38,40,41,42,46,47,49,58,97)
j <- c(1,4,7,10,11,13,25,26,34,36,39,43:45,48,50:57,59:96,98:121)
panel.splom(x[i],y[i], pch = 16, cex = 0.7, col = 1)
panel.splom(x[j],y[j], pch = 16, cex = 0.4, col = 2)
},sub=list("Points In Black: Angola, Benin, Burkina Faso, CAR, Chad,
Ethiopia, Gambia, Ghana, Ivory Coast, Kenya, Lesotho, Liberia, Madagascar,
Malawi, Mali, Mauritania,
Mozambique, Niger, Nigeria, Rwanda, Senegal, Sierra Leone, Somalia, Sudan,
Tanzania, Togo, Uganda, Zaire, Zambia, Bangladesh, Burma, India, Nepal,
Haiti",cex=0.5))
```

Figure 9. Scatterplot Matrix of Basic Solow Model in R with Certain Points Selected



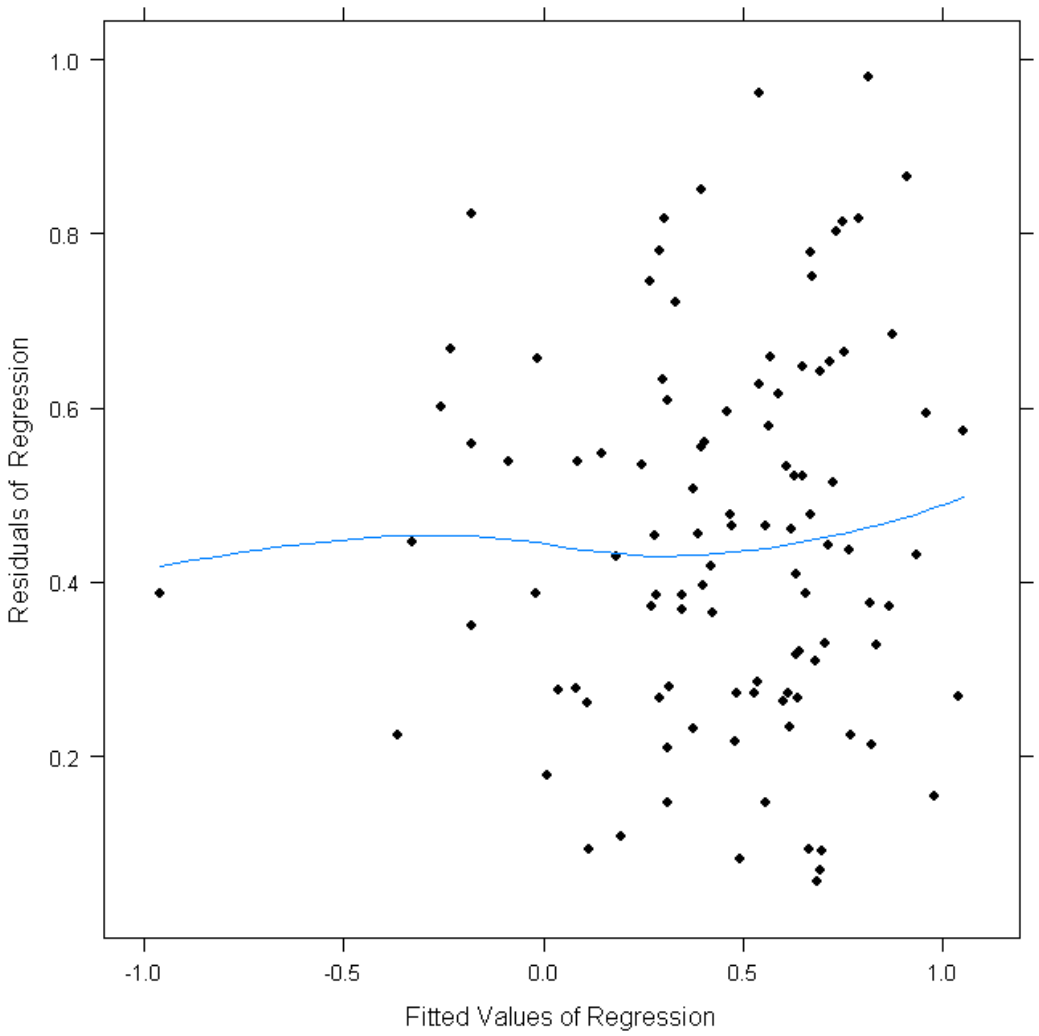
Visualizing Heteroskedasticity

Cleveland (1993) discusses one technique that can be used to visualize heteroskedasticity in cross-sectional data. In this case, the heteroskedasticity is visualized simply by adding a **loess** (short for "local estimation") regression line to a scatterplot of the **myreg** residuals against the fitted values of **myreg** that were estimated at the beginning. This is not to be confused with **lowess**, which STATA has, which is the predecessor and stands for locally-weighted estimation, which I believe was also created by Cleveland. I can't remember why, but I think **loess** is better than **lowess**, but STATA users don't seem to know of this yet, at least not as I was writing this.

```
> xyplot(sqrt(abs(residuals(myaugreg))) ~ fitted.values(myaugreg), panel =
function(x, y)
{panel.xyplot(x, y, pch = 16, cex = 0.7, col = 1)
panel.loess(x, y, span = 1, degree = 2, family = 'symmetric')
}, aspect = 1,xlab="Fitted Values of Regression",ylab="Residuals of
Regression")
```

If there were no heteroskedasticity the **loess** line would be flat. Instead we see curvature, i.e., there is heteroskedasticity in the cross section of data.

Figure 10. Residual-Fit Plot



Quantile Regressions Part I: Mean (OLS) vs. Median (LAD) Regressions

The regressions estimated previously applied the method of ordinary least squares (OLS), proposed by Gauss, when he was about 18 years old (see Wikipedia). That is, you choose the parameter values that minimize the sum of squared residuals. You can see appendix 1 for a formulation of the problem. However, in the history of statistics, there is a regression estimator that predates the method of least squares. Koenker (2006) cites a reference pointing to Boscovich as the originator of the median regression that predates OLS. This estimator chooses parameters that minimize the sum of absolute errors. Here again, a formal presentation of this is found in appendix 1. If you know the difference between the mean and median, you also know the difference between OLS and the median regression: they're multi-variate extensions of the simple location statistics. STATA has a routine for quantile regressions, and the median regression is the default (I'll get to the other quantiles later, but for now just think of median vs. mean). R does not have a canned package, so now STATA laughs at R, saying "haha, you don't even have quantile regressions". To which R responds, "Not so fast, you can just download the routine written by Roger Koenker, the co-creator of regression quantiles, himself!" Now since you've already seen how to load the lattice package, try same thing but this time go to the Packages menu and under Install Packages(s) select **`quantreg`** from the R web-site. Then you go back to the Packages menus and under Load package select quantreg again. With this done, replace the **`lm()`** command with **`rq()`**, which produces the following

R Code:

```
> myqreg <- rq(DY~LGDP60+LNGD+LINV,data=mydata)
> summary.rq(myqreg)

       Call: rq(formula = DY ~ LGDP60 + LNGD + LINV, data = mydata)
       tau: [1] 0.5
       Coefficients:
                   coefficients lower bd upper bd
       (Intercept)  1.85431      1.06070  3.18443
       LGDP60      -0.20447     -0.25420 -0.09781
       LNGD        -0.57574     -0.83916 -0.12555
       LINV         0.73441      0.54269  0.93355
```

STATA translation:

```
qreg dy lgdp60 lngd linv

       Median regression                       Number of obs = 105
         Raw sum of deviations 39.07244 (about .45447299)
         Min sum of deviations 27.83085        Pseudo R2 = 0.2877
       dy     Coef.        Std. Err.     t      P>t    [95% Conf. Interval]

       lgdp60 -.2044671     .0535138     -3.82  0.000   -.3106242 -.0983101
       lngd   -.5757396     .3311618     -1.74  0.085  -1.232675   .0811963
       linv    .7344133     .1047563      7.01  0.000    .5266049  .9422216
       _cons  1.85431       1.041517      1.78  0.078   -.2117789 3.920399
```

Now we can compare the average and median regressions. The intercept for the average is higher (**`2.01428`** vs. **`1.85431`**). However, I once asked Gilbert Bassett about the intercept and I believe he said that in quantile regressions it does not have a straightforward interpretation. So, perhaps they are not comparable here. Convergence for the average is a bit weaker as growth rates are just a bit less sensitive to initial income (**`-0.18296`** vs. **`-0.20447`**), population growth has a weaker effect on

growth for the average relative to the median (`-0.42477` vs. `-0.57574`), investment has a slightly smaller marginal effect than at the median (`0.69322` vs. `0.73441`). That's for the basic Solow model. You'll also notice that R doesn't produce standard errors for quantile regressions. A former World Bank colleague who learned econometrics from Roger Koenker told me he doesn't believe in the usual goodness of fit measures in classical statics, such as standard errors, or R-squared, for quantile regressions, because he also doesn't believe in the Central Limit Theorem. I don't mean that he has disproven it, but if you read the first page of the Koenker and Bassett Econometrica article, they quote the Statistician Cramer recalling Poincare's aphorism

> … everyone believes in the [Gaussian] law of errors, the experimenters because they think it is a mathematical theorem, the mathematicians because they think it is an experimental fact.

You could still replace experimenters with "many applied econometricians" and mathematicians with "many econometric theorists." The lesson is, don't be in awe of the Central Limit Theorem; it is limited in its usefulness. So, what about the Augmented Solow model for the median?

R Code:

```
> myaugqreg <- rq(DY~LGDP60+LNGD+LINV+LSCH,data=mydata)
> summary.rq(myaugqreg)

Call: rq(formula = DY ~ LGDP60 + LNGD + LINV + LSCH, data = mydata)
tau: [1] 0.5
Coefficients:
            coefficients lower bd upper bd
(Intercept)  2.60694      2.02883  3.33940
LGDP60      -0.30527     -0.37056 -0.24153
LNGD        -0.68896     -0.91396 -0.54166
LINV         0.56529      0.44944  0.74418
LSCH         0.17994      0.09593  0.28354
```

STATA translation:

```
qreg dy lgdp60 lngd linv lsch

Median regression                               Number of obs = 104
  Raw sum of deviations 38.77058 (about .45447299)
  Min sum of deviations 25.65635                Pseudo R2 = 0.3383
dy     Coef.        Std. Err.     t      P>t    [95% Conf. Interval]

lgdp60 -.3052676     .0406877     -7.50  0.000   -.3860008 -.2245343
lngd   -.6889622     .2171608     -3.17  0.002  -1.119856  -.258068
linv    .5652871     .0787763      7.18  0.000    .408978   .7215963
lsch    .179945      .0510167      3.53  0.001    .0787167  .2811732
_cons  2.606932      .7782992      3.35  0.001   1.062618  4.151247
```

When you add in education, the intercept for the average is higher (`3.11285` vs. `2.60694`), convergence for the average is a bit weaker as growth rates are less sensitive to initial income (`-0.29732` vs. `-0.30527`), population growth has a weaker effect on growth for the average relative to the median (`-0.50668` vs. `-0.68896`), investment has a smaller marginal effect than at the median (`0.55286` vs. `0.56529`), and schooling is associated with slightly higher growth rates (`0.21645` vs. `0.17994`). I hope you got something out of the basics of data manipulation and regression.

Quantile Regressions Part II: What About Regressions for Non-Central Locations

In the language of statistics, the mean and median are univariate measures of location. They are also measures of central tendency for the typical observation (the average and middle, respectively). Likewise, OLS and the median regression are multivariate measures of location, and likewise, they provide estimates of central tendency. If you read most journal articles they focus on central tendency too. Often, but not always, the most interesting questions in economics have nothing at all to do with central tendency. Here are some examples: 1) compensation packages for executives within a corporation [think Bill Gates vs. the middle-level managers], 2) caloric intake or consumption levels for really impoverished people, 3) really large exchange rates devaluations, 4) really high or low test scores, 5) peaks and troughs of business cycles. There are many other examples.

The point is, as you're doing your research it's a good idea to ask yourself if what you are looking to analyse is a typical phenomenon, or an extreme event. If it's a typical phenomenon, then it's okay to use means and medians, and their multi-variate brothers, OLS and median regressions. On the other hand, if you are dealing with extreme events, it's a good idea to know a little something about quantile regressions.

As we saw earlier, you can look at non-central locations of the univariate distribution in R using the `quantile()` function, or in STATA by asking for various percentiles in the `tabstat` command (recall quantile is another way to say percentile). Well, we can also do the same thing in multivariate settings, with quantile regressions.

We already compared the mean and median regressions. Now let's look at the regressions for the 25th and 75th percentiles (the 10th and 90th are also commonly reported, but you might also consider trying something as amazing as Koenker's analysis of Engel's data over the entire range of percentiles, reported in his Vignette).

First let's consider the 25th percentile. This can answer the question, how does the 25th percentile country GDP growth rate vary with increases in the "exogenous variables" (remember, unlike the average reported by OLS this may represent an actual observation)? Comparing with the median, for slower growth (25th percentile) countries, we see that the intercept is lower (`1.60514` vs. `2.60694`), convergence is weaker in the sense that growth rates are less sensitive to initial income (`-0.24688` vs. `-0.30527`), population growth has an even stronger association with lower growth (`-0.82764` vs. `-0.68896`), investment has a lower marginal effect than at the median (`0.49098` vs. `0.56529`), but schooling is more strongly associated with growth (`0.23418` vs. `0.17994`). To run this command, try the following R and STATA codes.

R Code:

```
> myaugq25reg <- rq(DY~LGDP60+LNGD+LINV+LSCH,tau=0.25,data=mydata)
> summary.rq(myaugq25reg)
```

```
Call: rq(formula=DY~LGDP60+LNGD+LINV+LSCH,tau=0.25,data=mydata)
tau: [1] 0.25
Coefficients:
            coefficients lower bd upper bd
(Intercept)  1.60514     -0.33042  3.71041
LGDP60      -0.24688     -0.39319 -0.18849
LNGD        -0.82764     -1.43023 -0.27781
LINV         0.49098      0.24756  0.71308
LSCH         0.23418      0.04809  0.36995
```

STATA Translation:

```
qreg dy lgdp60 lngd linv lsch, q(0.25)
```

```
[Reported Iterations Omitted]

.25 Quantile regression                              Number of obs = 104
  Raw sum of deviations 30.98486 (about .13145199)
  Min sum of deviations 21.47646                     Pseudo R2 = 0.3069
dy     Coef.         Std. Err.     t      P>t    [95% Conf. Interval]

lgdp60 -.2468809      .0764857    -3.23  0.002   -.3986452  -.0951166
lngd   -.8276225      .3447963    -2.40  0.018  -1.511773   -.1434719
linv    .4909817      .161306      3.04  0.003    .1709156   .8110479
lsch    .2341849      .108606      2.16  0.033    .018687    .4496827
_cons  1.6052        1.298624      1.24  0.219   -.9715513  4.181951
```

Now how about the the 75th percentile? This can answer the question, how does the 75th percentile country GDP growth rate vary with increases in the "exogenous variables?" The story is almost the opposite. Comparing with the median, for higher growth (75th percentile) countries, we see that the intercept is higher (**`4.81039`** vs. **`2.60694`**), convergence is stronger as growth rates are more sensitive to initial income (**`-0.42394`** vs. **`-0.30527`**), the association between income and population growth and is almost halved (**`-0.39788`** vs. **`-0.68896`**), investment has a slightly stronger marginal effect than at the median (**`0.58262`** vs. **`0.56529`**), but this time schooling is also more strongly associated with growth (**`0.26671`** vs. **`0.17994`**).

R Code:

```
> myaugq75reg <- rq(DY~LGDP60+LNGD+LINV+LSCH,tau=0.75,data = mydata)
> summary.rq(myaugq75reg)
```

```
Call: rq(formula=DY~LGDP60+LNGD+LINV+LSCH,tau=0.75,data = mydata)
tau: [1] 0.75
Coefficients:
            coefficients lower bd upper bd
(Intercept)  4.81039      4.17548  6.10656
LGDP60      -0.42394     -0.43502 -0.25899
LNGD        -0.39788     -0.50518  0.18528
LINV         0.58262      0.39199  0.83023
LSCH         0.26671      0.08699  0.34739
```

STATA Translation:

```
qreg dy lgdp60 lngd linv lsch, q(0.75)
```

```
       [Reported Iterations Omitted]

       .75 Quantile regression                              Number of obs = 104
         Raw sum of deviations 30.13928 (about .73368597)
         Min sum of deviations 20.31923                     Pseudo R2 = 0.3258

       dy     Coef.         Std. Err.     t      P>t    [95% Conf. Interval]

       lgdp60 -.4239411      .0641093    -6.61  0.000   -.551148   -.2967343
       lngd   -.3978789      .3409185    -1.17  0.246  -1.074335    .2785775
       linv    .5826213      .1219667     4.78  0.000    .3406128   .8246298
       lsch    .2667149      .0771809     3.46  0.001    .1135711   .4198586
       _cons  4.810404      1.085709      4.43  0.000   2.656122   6.964685
```

Quantile Regressions are Robust to Heteroskedasticity

There's another really good reason to use quantile regressions. Koenker and Bassett (1982) propose a test for heteroskedasticity based on regression quantiles. The basic idea is simple: if the slope coefficients are identical across quantiles, then the errors are homoskedastic, otherwise, there is heteroskedasticity. Unlike OLS, heteroskedasticity is not a problem for regression quantiles. By quickly eyeballing the median, 25th and 75th percentile coefficients you see that they're not equal. This result confirms what you saw in Figure 10., with the non-parametric visualization check for heteroskedasticity. The point is, though, that it's there, and Mankiw, Romer and Weil don't look for it. The other point is that you can gladly forget about Breusch-Pagan-Godfrey, until the referee tells you that they want to see it.

Resampling

The idea behind resampling is to "create" more data from an existing sample of data to get at problems of efficiency. You might have a small sample of data, and would like to get a sense of whether an estimator is biased and efficient. There are two workhorses in resampling, the older Jackknife method proposed by Richard Von Mises but named by John Tukey, and the newer Bootstrapping method, proposed by Efron. With a Jackknife suppose you have a sample of *N* observations. Now imagine that instead of applying your estimator to the sample of size *N*, you create *N* sub-samples without replacement of size *N-1* (by sampling without replacement, I mean "once you choose it, you lose it," in the sense that it if you pull it out of the bag and look at it, you do not then put it back in the bag to be chosen again). In other words, for our sample of 105 countries, a Jackknife would involve creating 105 different sub-samples of 104 countries, each with a different country removed. If your estimator is a linear regression, then each time you estimate the regression on the sub-sample of 104 observations, you collect the regressions coefficients, and when you have done this 105 times, you can compute the sample averages and standard deviations of the coefficients and compare them with what you get after applying OLS to the entire sample. In a worst case, you might be able to do a Jackknife by hand if the sample is not too large. I don't have much experience with the Jackknife, so if I find someone's codes to Jackknife regression coefficients, I'll show you how to do this. With modern computational power, there is another possibility.

Unlike the Jackknife, Bootstrapping involves generating samples with replacement. Sampling with replacement means you pull it out of the bag, look at it and keep it in memory, and put it right back in the bag so that it might be chosen again. If the sample is of size N, then you might create X new samples of size N, but some of the observations may be repeated, since it is sampled with replacement. In the Solow model example, then you might create X = 1000 or 10,000 new samples of size 105, but each sample may contain the same observation a few times (i.e., the investment, schooling rate, or GDP growth rate for Germany or Zimbabwe has the potential to appear a few times in each sample, while others may not appear). In this way, Bootstrapping gives you the possibility of creating a much larger dataset to explore the effectiveness of the model, relative to the Jackknife. Also, it is worth mentioning that there are non-parametric and parametric bootstraps. I will apply non-parametric bootstrapping to the Solow regression from above in R. The idea with the Non-Parametric Bootstrap is to estimate with coefficients of the model but from the resampled data. I'll get to the Parametric Bootstrapping later and it is relatively straightforward to run your own Parametric Bootstrap codes.

The following lines of code can be summarized as follows. The first block sets up the Non-Parametric Bootstrap. The second block does the actual bootstrapping of regression coefficients. The third block computes the standard errors of the coefficients. The last two blocks of code are more cosmetic in nature. The fourth block reports a table of the OLS, Bootstrap coefficients, and their standard errors, and the fifth block reports in a table the standard errors from the original OLS model and the bootstrap standard errors.[14]

[14] I wrote these codes by combining R codes for bootstrapping from two different sources. The first is from http://zoonek2.free.fr/UNIX/48_R/16.html#13, and the second is from http://www.stat.psu.edu/~dhunter/R/2006test.html.

A Nonparametric Bootstrap

Setting up the Bootstrap

[In a nutshell: you estimate the original model, save the coefficients that will be used for comparison using the fitted values, and you also save the residuals, and create the matrix of the right-hand side using the **model.matrix** command]

```
> mydata <- read.table("C:/temple.csv",header=T,sep=",")
> mod1 <- lm(DY~LGDP60+LNGD+LINV,data=mydata)
> mod1coefs <- coef(mod1)
> fit <- fitted(mod1)
> e <- residuals(mod1)
> X <- model.matrix(mod1)
```

The Actual Bootstrap

[In a nutshell: you use the **s <- sample** command to get a grouped-sample of the values for **LGDP60**, **LNGD**, **LINV** for 105 countries. Since it is sampling with replacement, in each of the 1000 samples, Germany, or Zimbabwe may appear more than once. The variable **DY** is "simulated" in that you take an actual resampled error, and combine that with the original fitted values, and then you re-estimate the regression (minus the intercept, since the intercept is already in the model matrix X)]

```
> mod2 <- NULL
> for (i in 1:1000)
{
       s <- sample(length(X[,1]),replace=T)
       y <- fit + e[s]
mod2 <- rbind(mod2, lm(y~-1+X)$coef)
}
```

You can compute the standard errors by computing the covariance matrix of the model coefficients, and taking the square root of the diagonal components

```
> cov(mod2)
> se <- sqrt(diag(cov(mod2)))
```

To bring all the information together, I then run the codes below (the table is not as nice as something STATA would give you)

```
> OLS <- c(mod1coefs[1],mod1coefs[2],mod1coefs[3],mod1coefs[4])
> BootStrap <- c(mean(mod2[,1]),mean(mod2[,2]),mean(mod2[,3]),
mean(mod2[,4]))
> Bias <- c(mean(mod2[,1])-mod1coefs[1],mean(mod2[,2])-mod1coefs[2],
    mean(mod2[,3])-mod1coefs[3],mean(mod2[,4])-mod1coefs[4])
> table <- cbind(OLS,BootStrap,Bias)
```

	OLS	BootStrap	Bias
(Intercept)	2.0142775	2.0109613	-0.0033162486
LGDP60	-0.1829637	-0.1833834	-0.0004196922
LNGD	-0.4247713	-0.4294536	-0.0046822769
LINV	0.6932226	0.6958266	0.0026039566

```
> coefsum <- coef(summary(mod1))
> OLSSE <- c(coefsum[1,2],coefsum[2,2],coefsum[3,2],coefsum[4,2])
> tableSE <- data.frame(OLSSE,BootSE=se,row.names=c("Intercept","LGDP60",
"LINV","LSCH"))

                OLSSE    BootSE
Intercept 0.83695283 1.1749340
LGDP60    0.04163735 0.0559561
LINV      0.26517377 0.3890783
LSCH      0.08368177 0.1113108
```

STATA Translation:

```
clear
set mem 100m
insheet using "C:\temple.csv", comma
drop lsch
generate lsch=ln(school/100)
regress dy lgdp60 lngd linv lsch
bs "regress dy lgdp60 lngd linv" "_b[_c] _b[lgdp60] _b[lngd] _b[linv]",
reps(1000)
```

[this is the old version of STATA's regression coefficient bootstrapping command]

```
bootstrap _b, reps(1000) bca: regress dy lgdp60 lngd linv estat bootstrap,
all
```

[this is the new version of STATA's regression coefficient bootstrapping command, and it gives you the following output, which is similar to what the old command does]

```
                           Observed     Bootstrap
             Coef.         Bias         Std. Err.
lgdp60       -.1829637      .0069137    .0452954
lngd         -.42477123     .0340677    .23587004
linv          .69322258    -.0086114    .09962956
_cons        2.0142776      .0207707    .83168788
```

A Parametric Bootstrap

In the Parametric Bootstrap, the idea is that you fix the coefficients and see how the estimator performs. This is often applied when you want to see how well an estimator does relative to others, for different types of error distributions (like fat-tailed or potentially skewed distributions [I'll note that I have yet to see anyone do Bootstrapping for skewed distributions though]) and perhaps sample sizes. An example of a Parametric Bootstrap is produced below, in which OLS is compared to the median regression. What makes it parametric is that, as you will see, I assume that I know that the true regression has a slope of 2, and an intercept equal to zero. I assume that the independent variable follows a discrete time random walk with a drift of 4.2%, and a volatility parameter of 12.5%. The structure of the codes was inspired by some S-Plus codes I found on Eric Zivot's web-page for his financial econometrics teachings. To get my dependent variable, I then combine this independent variable with a t-3 random error term, in a regression of the form: **`trial.indret = a + b*trial.mktret + trial.res`**, with a = 0 and b = 2. I randomly draw samples of size 250, and run the simulation 10,000 times. Vectors are first created to be filled with the estimates from each loop (the **`rep(0,`** … creates a vector of length 10,000 that is then filled with estimates). You should hopefully have the quantile regression package installed from earlier, but you may need to Load the **`quantreg`** package again

```
> Rw <- 0.04287973
> sigmaw <- 0.1245768
> sample <- 250
> n.trial <- 10000
> alphaols <- rep(0,n.trial)
> betaols <- rep(0,n.trial)
> alphalad <- rep(0,n.trial)
> betalad <- rep(0,n.trial)
> set.seed(200)
> a <- 0
> b <- 2
> for (trial in 1:n.trial) {
       trial.mktret <- Rw+sigmaw*rnorm(sample,mean=0,sd=1)
       trial.res <- rt(sample,df=3)
       trial.indret <- (a + b*trial.mktret + trial.res)
       trial.ols <- lm(trial.indret~trial.mktret)
       tmpols <- coef(trial.ols)
       alphaols[trial] <- tmpols[1]
       betaols[trial] <- tmpols[2]
       trial.lad <- rq(trial.indret~trial.mktret)
       tmplad <- coef(trial.lad)
       alphalad[trial] <- tmplad[1]
       betalad[trial] <- tmplad[2]
}
```

To see how OLS compares with the median regression, for this model, consider first the means of the coefficients for the alpha (intercept) and beta (slope)

```
> c("mean ols alpha"=mean(alphaols),"mean lad alpha"=mean(alphalad),"mean
ols beta"=mean(betaols),"mean lad beta"=mean(betalad))

       mean ols alpha mean lad alpha  mean ols beta  mean lad beta
         0.0009543169   0.0012759275   1.9979784349   2.0010737045
```

The values are quite close. To see how close from the truth, we can calculate the bias as the difference from the true values (a = 0, b = 2)

```
> c("ols alpha bias"=(mean(alphaols)-a),"lad alpha bias"=(mean(alphalad)-
a),"ols beta bias"=(mean(betaols)-b),"lad beta bias"=(mean(betalad)-b))

       ols alpha bias lad alpha bias  ols beta bias  lad beta bias
         0.0009543169   0.0012759275  -0.0020215651   0.0010737045
```

If a bootstrap average is unbiased, it will be zero out to two decimal places. So both regression methods give unbiased coefficients when the error term is sampled from a t-3 distribution, which is quite heavy tailed. We can also look at the efficiency.

```
> c("stdev ols alpha"=sd(alphaols),"stdev lad alpha"=sd(alphalad),"stdev ols
beta"=sd(betaols),"stdev lad beta"=sd(betalad))

        stdev ols alpha stdev lad alpha  stdev ols beta  stdev lad beta
             0.11572855      0.09123227      0.88464982      0.69998224
```

What you see is that when the errors are sampled from a t-3 distribution, which is much heavier-tailed than the normal distribution, the median regression is more efficient than OLS, due to the smaller standard errors for the intercept (`0.09123227` vs. `0.1157286`) and the slope (`0.6999822` vs. `0.8846498`). Put another way, it's a more informative summary.

Merging Data Files

As a pretext to discussing panel data analysis, I'll begin by discussing the act of merging two data files, which may seem a trivial task, but it can quickly become complicated, especially when working with panel data. The larger the data file, and the more complex the structure of the data, the trickier it becomes to see what you are doing, unless you happen to think like a computer. STATA is quite good for managing large data files, and is quite useful at letting you know what it just did. R is also good, but, if you take a look at Grant Farnsworth's Econometrics with R handout, you apparently have to know how R uses memory to read in large data files.

First, I'll go through the process of merging two cross-sectional data files. In this simple example, assume that you wanted to add to the data file "**mydata**", another file that has as a random disturbance for each country (I don't know when you'd ever do this, but assume it was a variable you wanted). Also, assume the variable with the country names was called "countryname" in the second file, whereas in "**mydata**," recall that the corresponding variable was simply called "country." For the purpose of demonstrating how to merge I'll first create this new data set called "**myrandom**".[15]

```
> myrandom <-
data.frame("countryname"=mydata$country,"Random"=rnorm(121,0,1))
```

Having created this data object, how would I add it to the main file? The answer is with the following command

```
> mynewdata <- merge(mydata,myrandom,by.x="country",by.y="countryname")
```

What this does is to tell R to create a new data object called "mynewdata," which merges the file/dataframe called "**myrandom**" into the file called "**mydata**". The options "**by.x="country"**" and "**by.y="countryname"**" tell R that you want to assign any observation associated with a particular country in **myrandom$countryname** to the same country listed under the variable country in **mydata$country.**

As easy as that looked, STATA's equivalent looks much more complicated. First, off, remember that STATA, unlike R, can only keep one data set in memory, so while I could create this new data set almost without thinking in R (just one line of code), I need several lines to do the same thing in STATA. Also, STATA will not merge two datasets if the variable(s) that you are using to assign values in the merging file into the masterfile do not have the same name. So, here is the STATA equivalent of the operation I just did for R. I'll first describe it in words, and then below, you'll see the codes to tell STATA exactly how to do all the steps I'm about to describe.

First I clear the memory, then I tell STATA how much memory to use (which is necessary when handling large datasets). This is followed by a request to import the "temple.csv" file, which is a comma separated file. I'll now save this file, using the name **mydata.** By the way, the **replace** option after the **save** command is useful if you are going to run these codes repeatedly, meaning you'll have to overwrite the

[15] This command creates a data object called a "dataframe", which can combine both character and numeric data. This I need in order to merge with "mydata", since it too combines character and numeric data.

files, each time you open up STATA using the do-file. If you leave out that option, STATA will stop running, and complain each time your run the command that the file already exists. Now you can create the **myrandom** dataset first by keeping only the variable named country using the **keep** command. Then after creating a random sample of data, using the **invnormal** command (and here's another area where R is way better than STATA: random number generation), you save that data set. Note, unlike R, the variables that you are using to merge must have the same name in the two STATA datasets, otherwise, it will NOT merge, saying "I don't know where that variable is." Now you can merge **myrandom** into **mydata**. The first thing you'll typically have to do is to sort the data using the **sort country** command. Note, we didn't have to sort in R, because it figured it all out by itself [though I don't know how accurate it is]. Then after you **merge country using** … you can ask STATA what it just did with the command **tab _merge**, because STATA automatically creates a variable called **_merge** each time you merge two data files. This command is a shortened form of tabulate, and it will give you an idea of how many observations were common to both files, and how many observations were unique to each of the two files. There are three possibilities: if you see a 1, that means it was only found in the master file, if you see a 2, that means that the observation was only found in the merging file, and if you see a 3, that means the observation was common to both files. In this case, you see only 3's. Note that you will have to drop this variable if you plan to merge in other data too, because STATA will not rename the variable **_merge** for you. Also, there are times when the information contained in the variable **_merge** can be used as a dummy variable. So you should always ask yourself does that variable having meaning in my analysis, in which case, you should just rename it and not drop it. Finally, you can save this new file called **mynewdata.dta**. So, the process was more complicated, but you seem to be able to do a number of things with this.

```
clear
set mem 100m
insheet using "C:\temple.csv", comma
save "C:\mydata.dta", replace
keep country
gen random=invnormal(uniform())
sort country
save "C:\myrandom.dta", replace
clear
use "C:\mydata.dta"
sort country
merge country using "C:\myrandom.dta"
tab _merge
drop _merge
sort country
save "C:\mynewdata.dta", replace
```

Note the **tab _merge** command will give you something like this

```
_merge        Freq.        Percent      Cum.
3             121          100.00       100.00
Total         121          100.00
```

Now, this is a really simple merge operation. You're only merging by one variable, whose content is common to both data files. However, if you work with panel data, you may often have two or three layers that you would have to account for before you can merge. So, I'll show you how to generalize it for situations when you have to merge panel data files for R and STATA (it's not hard, just a bit tricky, since the more

complex the file, the greater is the likelihood for making mistakes). Imagine you have two panel data sets named **panelmaster** and **panelmerging** with observations for cities within countries over time. The two files have common names **country**, **city** and **year**. The first thing to do is make sure you sort BOTH files, on **country** first, then **city** and then **year**. Also, I'll assume this is an unbalanced panel, so you don't have observations for all cities, or countries in each year. So it might look like

county	**city**	**year**	**othervariables** …
Afghanistan	Kabul	1961	
Afghanistan	Khandahar	1961	
Afghanistan	Kabul	1962	
Afghanistan	Khandahar	1962	
…			
Afghanistan	Khandahar	2003	
Afghanistan	Kabul	2004	
Belguim	Brussels	1973	
Belguim	Brussels	1974	
Belguim	Ghent	1974	
…			
Belguim	Brussels	2000	
Belguim	Ghent	2000	
…			
Zimbabwe	Harare	1965	
Zimbabwe	Harare	1966	
…			
Zimbabwe	Harare	2002	

In R, you can merge a file called **panelmerging** into a file called **panelmaster**. The two files have three variables in common, but in this case the one pair of merging variables have different names. The character variables with the names of the countries are called, respectively, **countryname** and **country**, and for the cities, it is **city** in one and **cities** in the other. This simply requires the following line of code

```
> merge(panelmaster,panelmerging,by.x=c("countryname","city"),
by.y=c("country","cities"))
```

R merges **country** into **countryname** first, and then **cities** into **city**, which is the reason why you see the order in **by.x** and **by.y** written as such. Note STATA, by default, sorts in ascending order. Once you sort, then you can do the merge. Recall the names of the merging variables must be the same, so the STATA equivalent is

```
clear
set mem 100m
use "C:\panelmaster.dta"
sort country city year
merge country city using "C:\panelmerging.dta"
tab _merge
drop _merge
sort country city year
save "C:\newpanel.dta", replace
```

Here again, the **tab _merge** command will be extremely useful, and it will help you find out if there's a problem with the merge. While this may seem a trivial operation, I learned on the job, and it probably took me about three or four weeks to get used to doing this with large household surveys. So, it's better that you see this now rather than later. In addition to household surveys, the management of high-frequency asset price data sets is another context in which it could be useful to know how to manage so-called *k*-large datasets, and merge operations might be crucial here too.

Dummy Variables and Panel Regression

A Digression on Creating Dummy Variables from Categories in STATA

You saw one way to use dummy variables in the section on replicating empirical work, when I estimated regressions, if a particular variable was equal to 1 (the other observations not included in the estimation were associated with that variable equaling zero). However, in that exercise, there was no attempt to extract quantitative information based on those variables. R is pretty flexible when it comes to dummy variables: you can use the variable(s) as a factor, or in numeric format. Here are two character variables, which are examples of factors

hhhead	**region**
female	Africa
male	Australia
male	Asia
female	Asia
…	…
male	Europe
female	South America
female	North America

Alternatively, while this information could have been stored in two character variables, the same information could also be have been expressed in numeric format as a series of columns with ones and zeros

dFEM	**dAFR**	**dASA**	**dAUS**	**dEUR**	**dNAM**	**dSAM**
1	1	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
1	0	1	0	0	0	0
…						
0	0	0	0	1	0	0
1	0	0	0	0	0	1
1	0	0	0	0	1	0

So the first row, could be interpreted as female & Africa, the second row as male & Australia, the third row as male & Asia, etc. As long as you input all of this information into R's memory, it will interpret the two variations as equivalent. For instance, if you are running a regression of household consumption (**`hhcon`**) against a bunch of variables, including sex of the household head (**`hhhead`**), region in which they reside (**`region`**), if you have the data written as in the first case, or (**`dFEM`**, **`dAFR`**, **`dAUS`**, **`dASA`**, **`dEUR`**, **`dNAM`**, and **`dSAM`**), in the second case, and other variables, you would get the same results whether you ran

```
lm(hhcon~hhhead+region+othervars,data=mycondata)
```

or

```
lm(hhcon~dFEM+dAFR+dAUS+dASA+dEUR+dNAM+dSAM+othervars,data=mycondata)
```

because R will automatically and internally create the numeric dummy variable **dFEM** since female is alphabetically first in the example above, from the character variable **hhhead**, and **dAFR, dASA, dAUS, dEUR, dNAM**, and **dSAM** from the character variable **region**. STATA will not. So, if your STATA dataset has the variables in character format, you must turn them all into numeric format. To do this, it's quite simple

```
tab hhhead, gen(d)
rename d1 dFEM
drop d2
```

The first line tabulates the different kinds of categories, and the "**, gen(d)**" portion of the command creates dummy variables, that begin with **d**. Since there are two categories, it creates two dummy variables, **d1** and **d2**. I wrote the second line because when STATA runs the first line, it sorts the character variable alphabetically, in this case, **female** is before **male**, so **d1** takes a 1 for every female observation, and zero otherwise, and **d2** takes a 1 for every male observation and zero otherwise. I then rename **d1** as **dFEM**. For the other, multiple-category variable, **region**, it will create six dummy variables **d1, d2, d3, d4, d5** and **d6**, which I will rename

```
tab region, gen(d)
rename d1 dAFR
rename d2 dASA
rename d3 dAUS
rename d4 dEUR
rename d5 dNAM
rename d6 dSAM
```

The STATA equivalent of the R dummy-variable regression command above is

```
reg hhcon dFEM dAFR dAUS dASA dEUR dNAM dSAM othervars
```

Basic Panel Regressions: Fixed Effects Estimation

Panel datasets, as I showed earlier in the discussion about merging, usually combine both cross-section and time dimensions. I'll not get into too much detail, because I'm no expert on the subject, and there are better references out there than what I can offer you (there is an excellent introduction by Hun Myoung Park that you can get from: http://www.indiana.edu/~statmath/stat/all/panel/panel.pdf). But I will show you how to work with a data set in both R and STATA. Find the web-site for benchmark data sets used to test the efficacy of estimators: http://www.stanford.edu/~clint/bench/. The Grunfeld dataset is found by doing "ctrl"+"f" on "grunfeld.xls", or by going to http://www.stanford.edu/~clint/bench/grunfeld.xls and downloading this file to your favorite location. Open up the data file and you'll see it has five variables, labeled FIRM, YEAR, I, F, and K. SAVE THIS FILE as a comma-separated file, so that it's in the same format as the **temple.csv** file. The first variable is the cross-sectional indicator, and the second is (obviously) the time series operator, and the [annoyingly] unlabeled variables and firm names can actually be found in Bond and DeWit (1960) on pages 27-28. I is millions of dollars of real investment, F is millions of dollars of the previous year's firm value (number of equity shares issued by the firm times the price, plus year end total outstanding firm debt), and K is millions of dollars of the firm's plant equipment minus depreciation from the previous year. Given this dataset, a natural issue to explain is firm current investment behaviour, as a function of lagged firm value and plant equipment.

As before, load the panel data as follows

```
> mypanel <- read.table("C:/grunfeld.csv",header=T,sep=",")
```

Least Squares Dummy Variables (LSDV)

Now, suppose you wish to estimate a fixed-effects regression across the firms, where firm one is the benchmark. The Least Square Dummy Variable (LSDV) equation is

3) $$I_t = \beta_0 + \delta_2 d_{firm2} + \cdots + \delta_{10} d_{firm10} + \beta_1 F_{t-1} + \beta_2 K_{t-1} + \varepsilon_i$$

I will show you one way to do this in R, since R is not picky about whether your data is expressed as a factor or as a dummy variable, and then three equivalent ways to do this in STATA, since STATA can be picky. In the first case, to demonstrate that R can turn factors into dummy variables, you can run the following commands

```
> xtfereg <- lm(I~factor(FIRM)+F+K,data=mypanel)
> sumfe <- summary(xtfereg)
> fecoef <- sumfe$coefficients
> fesigma <- sumfe$sigma
> sumfe

Call:
lm(formula = I ~ factor(FIRM) + F + K, data = mypanel)

Residuals:
      Min        1Q    Median        3Q       Max
-184.0086  -17.6432    0.5634   19.1922  250.7097
Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     -70.29672   49.70796  -1.414    0.159
factor(FIRM)2   172.20253   31.16126   5.526 1.08e-07 ***
factor(FIRM)3  -165.27512   31.77556  -5.201 5.14e-07 ***
factor(FIRM)4    42.48742   43.90988   0.968    0.334
factor(FIRM)5   -44.32010   50.49226  -0.878    0.381
factor(FIRM)6    47.13542   46.81068   1.007    0.315
factor(FIRM)7     3.74324   50.56493   0.074    0.941
factor(FIRM)8    12.75106   44.05263   0.289    0.773
factor(FIRM)9   -16.92555   48.45327  -0.349    0.727
factor(FIRM)10   63.72887   50.33023   1.266    0.207
F                 0.11012    0.01186   9.288  < 2e-16 ***
K                 0.31007    0.01735  17.867  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 52.77 on 188 degrees of freedom
Multiple R-Squared: 0.9441,     Adjusted R-squared: 0.9408
F-statistic: 288.5 on 11 and 188 DF,  p-value: < 2.2e-16
```

Now, what about an interpretation? The intercept has an awkward interpretation, in this application, because it suggests that firms with no prior value, or capital stock, disinvest. In this case, Firm 1 is the benchmark. This means any dummy variable coefficient, representing a particular firm, tells you whether that firm on average invests more or less than the benchmark (Firm 1). So if the benchmark average is -70, then Firm 2 on average invests, -70 million + 172 million ≈ 102 million, if Firm 2 had no value or capital stock, etc. Also, the coefficients for `F` and `K` tell you that on average, a one million dollar increase in firm value is associated with a 110,120 USD (think 11% of one million dollars) increase in investment, while a one million dollar increase in firm capital is associated with a 310,070 USD increase (think 31% of one

million dollars) in investment. You will see that this is exactly the same output as the STATA **xi: reg**, or interaction expansion, applied to the regression command. As before, use the insheet command to read in the data, and when you use the **xi: reg** command, you must specify which variable, in this case **firm**, is to be used to create dummy variables, using **i.firm** .

```
insheet using "C:\grunfeld.csv", comma
xi: reg i f k i.firm
```

```
      Source       SS           df     MS           Number of obs = 200
                                                    F( 11,   188) = 288.50
      Model        8836465.8    11     803315.073   Prob > F = 0.0000
      Residual     523478.114   188    2784.45805   R-squared = 0.9441
                                                    Adj R-squared = 0.9408
      Total        9359943.92   199    47034.8941   Root MSE = 52.768

      i      Coef.         Std. Err.     t      P>t    [95% Conf. Interval]

      f          .1101238    .0118567     9.29  0.000      .0867345   .1335131
      k          .3100653    .0173545    17.87  0.000      .2758308   .3442999
_Ifirm_2       172.2025     31.16126      5.53  0.000   110.7319   233.6732
_Ifirm_3      -165.2751     31.77556     -5.20  0.000  -227.9576  -102.5927
_Ifirm_4        42.4874     43.90987      0.97  0.334   -44.13197  129.1068
_Ifirm_5       -44.32013    50.49225     -0.88  0.381  -143.9243    55.28406
_Ifirm_6        47.13539    46.81068      1.01  0.315   -45.20629  139.4771
_Ifirm_7         3.743212   50.56493      0.07  0.941   -96.00433  103.4908
_Ifirm_8        12.75103    44.05263      0.29  0.773   -74.14994   99.652
_Ifirm_9       -16.92558    48.45326     -0.35  0.727  -112.5075    78.65636
_Ifirm_10       63.72884    50.33023      1.27  0.207   -35.55572  163.0134
_cons          -70.29669    49.70796     -1.41  0.159  -168.3537    27.76035
```

An alternative way to do the exact same thing in STATA is to first generate dummy variables for each of the firms, using the **tab firm, gen(d)** command. Leaving out the first firm as the benchmark case, the command with the firm dummy variables produces the exact same output

```
tab firm, gen(d)
reg i f k d2 d3 d4 d5 d6 d7 d8 d9 d10
```

```
      Source       SS           df     MS           Number of obs = 200
                                                    F( 11,   188) = 288.50
      Model        8836465.8    11     803315.073   Prob > F = 0.0000
      Residual     523478.114   188    2784.45805   R-squared = 0.9441
                                                    Adj R-squared = 0.9408
      Total        9359943.92   199    47034.8941   Root MSE = 52.768

      i      Coef.         Std. Err.     t      P>t    [95% Conf. Interval]

      f          .1101238    .0118567     9.29  0.000      .0867345   .1335131
      k          .3100653    .0173545    17.87  0.000      .2758308   .3442999
      d2       172.2025     31.16126      5.53  0.000   110.7319   233.6732
      d3      -165.2751     31.77556     -5.20  0.000  -227.9576  -102.5927
      d4        42.4874     43.90987      0.97  0.334   -44.13197  129.1068
      d5       -44.32013    50.49225     -0.88  0.381  -143.9243    55.28406
      d6        47.13539    46.81068      1.01  0.315   -45.20629  139.4771
      d7         3.743212   50.56493      0.07  0.941   -96.00433  103.4908
      d8        12.75103    44.05263      0.29  0.773   -74.14994   99.652
      d9       -16.92558    48.45326     -0.35  0.727  -112.5075    78.65636
      d10       63.72884    50.33023      1.27  0.207   -35.55572  163.0134
      _cons    -70.29669    49.70796     -1.41  0.159  -168.3537    27.76035
```

Deviations-From-Mean Estimator

The second block of STATA codes for the LSDV approach is good if you have a few firms and you would like the option to change the benchmark firm. For instance, if for some reason you wish to make firm 7 the benchmark, you can take out **d7** and put in **d1** and run the following command: **reg i d1 d2 d3 d4 d5 d6 d8 d9 d10 f k.** However, as the number of different factors increases, this can create problems as the estimators then become inconsistent (see the Park handout and references therein). In this case, there is a better way to do this, simply by demeaning each of the variables. In this case, I'm going to compute the difference for each firm across the firm-specific mean over time for each variable, but in some applications you might be interested in taking the difference between the variable and the cross-sectional average in each time period. The demeaned version that I am referring to is

4) $$I_{it} - \bar{I}_i = \beta_0 + \beta_1\left(F_{i,t-1} - \bar{F}_i\right) + \beta_2\left(K_{i,t-1} - \bar{K}_i\right) + \varepsilon_{it}$$

I'll first show you how to do this in STATA and then use it to show you how to do the same thing in R. STATA has a command to run this deviations-from-mean version. Before you run this, however, you must first tell STATA that you are working with time series data, using the **tsset** command. I looked in the help for tsset, and the last line has the generic command for a panel dataset **tsset panelid yearvar, yearly**, where panelid is like factor, in this case the firm, so

```
tsset firm year, yearly
xtreg i f k, fe
```

```
Fixed-effects (within) regression               Number of obs      =       200
Group variable (i): firm                        Number of groups   =        10

R-sq:  within  = 0.7668                         Obs per group: min =        20
       between = 0.8194                                        avg =      20.0
       overall = 0.8060                                        max =        20
                                                F(2,188)           =    309.01
corr(u_i, Xb)  = -0.1517                        Prob > F           =    0.0000

------------------------------------------------------------------------------
           i |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           f |   .1101238   .0118567     9.29   0.000     .0867345    .133513
           k |   .3100653   .0173545    17.87   0.000     .2758308    .344299
       _cons |  -58.74393   12.45369    -4.72   0.000    -83.31086   -34.177
-------------+----------------------------------------------------------------
     sigma_u |  85.732501
     sigma_e |  52.767964
         rho |  .72525012   (fraction of variance due to u_i)
------------------------------------------------------------------------------
F test that all u_i=0:     F(9,188) = 49.18                    Prob > F = 0.0000
```

STATA has a nice and neat table format for this regression. This is great if you'd like to tell an average story. However, you may wish to try something related to non-central tendency (i.e., quantiles). In this case it will help to know a little about how to program, so that you can change the estimator. You can see the link to figure out how to demean the variables. It's not difficult, and you can refer to http://www.wws.princeton.edu/wwac/stata/more/fixed.html for more panel tricks in STATA. You can simply type

```
sort firm year
by firm: egen meani = mean(i)
by firm: egen meanf = mean(f)
by firm: egen meank = mean(k)
gen demeani = i - meani
gen demeanf = f - meanf
gen demeank = k - meank
reg demeani demeanf demeank
```

```
      Source       SS           df       MS          Number of obs = 200
                                                     F(  2,   197) = 323.81
      Model        1720874.1     2      860437.05    Prob > F = 0.0000
      Residual     523478.127    197    2657.24938   R-squared = 0.7668
                                                     Adj R-squared = 0.7644
      Total        2244352.23    199    11278.1519   Root MSE = 51.549

      demeani      Coef.     Std. Err.   t     P>t    [95% Conf. Interval]

      demeanf       .1101238  .0115827  9.51  0.000    .0872818   .1329658
      demeank       .3100653  .0169534 18.29  0.000    .2766318   .3434989
      _cons        1.93e-08  3.64503    0.00  1.000  -7.188288   7.188288
```

You will notice that the coefficients are the same, however, the standard errors are biased downward. You can correct them by multiplying them by the square root of the degrees of freedom, $\sqrt{(NT-V)/(NT-N-V)}$, where *N* is the number of firms (10), *T* is the number of time periods (20 years), and *V* is the number of right-hand side variables (2, NOT including the intercept). I'll show you how to do the same thing in R and how to adjust the standard errors. The first trick is to demean the variables as Farnsworth (2006) shows

```
> g <- mypanel
> for (i in unique(mypanel$FIRM)){
      Timemean <- mean(mypanel[mypanel$FIRM==i,])
      g$I[mypanel$FIRM==i] <- mypanel$I[mypanel$FIRM==i]-timemean["I"]
      g$F[mypanel$FIRM==i] <- mypanel$F[mypanel$FIRM==i]-timemean["F"]
      g$K[mypanel$FIRM==i] <- mypanel$K[mypanel$FIRM==i]-timemean["K"]
      }
```

Now run the regression and create a summary

```
> xtferegdemean <- lm(I~F+K,data=g)
> sumfedemean <- summary(xtferegdemean)
```

```
      Call:
      lm(formula = I ~ F + K, data = g)

      Residuals:
            Min        1Q    Median        3Q       Max
      -184.0086  -17.6432    0.5634   19.1922  250.7097

      Coefficients:
                   Estimate Std. Error  t value Pr(>|t|)
      (Intercept) 6.221e-15  3.645e+00 1.71e-15        1
      F           1.101e-01  1.158e-02    9.508   <2e-16 ***
      K           3.101e-01  1.695e-02   18.289   <2e-16 ***
      ---
      Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

      Residual standard error: 51.55 on 197 degrees of freedom
      Multiple R-Squared: 0.7668,     Adjusted R-squared: 0.7644
      F-statistic: 323.8 on 2 and 197 DF,  p-value: < 2.2e-16
```

To correct the standard errors, as I mentioned above, you multiply the standard error by $\sqrt{(NT-V)/(NT-N-V)}$. For the *NT*, I just use the length of the data set **(length(mypanel$FIRM)**, which is 200 observations. For *N*, I use the **max(mypanel$FIRM)** since I know that that is a number, and the maximum value will represent the same thing as the total number of different firms. I haven't figured out a generic way to tell R how many independent variables to use, so for *V*, I just subtract 2 from the terms in the numerator and denominator, since I know that's how many right-hand side variables I have.

```
> fedemeancoef <- coef(sumfedemean)
> sevar1 <- fedemeancoef[2,2]*sqrt((length(mypanel$FIRM)-
2)/(length(mypanel$FIRM)-max(mypanel$FIRM)-2))
> sevar2 <- fedemeancoef[3,2]*sqrt((length(mypanel$FIRM)-
2)/(length(mypanel$FIRM)-max(mypanel$FIRM)-2))
```

I can then create a small table of the coefficients and the corrected standard errors, and below that the corrected standard error of regression using the following

```
> Coefficients <- c(F=fedemeancoef[2,1],K=fedemeancoef[3,1])
> S.E. <- c(sevar1,sevar2)
> SSR <- deviance(xtferegdemean)
> sigma <- sqrt(SSR/(length(mypanel$FIRM)-2))
> sigmacorrect <- sqrt(SSR/(length(mypanel$FIRM)-max(mypanel$FIRM)-2))
> table <- cbind(Coefficients,S.E.)

  Coefficients       S.E.
F    0.1101238 0.01188675
K    0.3100653 0.01739849

> tablesigma <- cbind(sigma,sigmacorrect)

         sigma sigmacorrect
[1,] 51.41818     52.76797
```

If you take a look at the STATA output for the **xtreg** from above, you will see that the standard errors are now the same, and the **sigmacorrect** here is the same as the **sigma_e** that is also reported in STATA's **xtreg** table.

Basic Panel Regressions: Random Effects Estimation

In most applications, fixed effects are probably what you're interested in estimating. Pushkar Maitra has pointed out to me that in empirical work related to experimental economics, random effects make more sense because you are presuming your subjects are a random sample from a larger population, and hence you have to take that into account. Another reason to think about random effects is that there might be something specific to the subject under study that affects the dependent and at least some of the independent variables. In this application, perhaps organizational form or firm governance might affect the capital stock and/or firm value, in addition to the firm's investment decisions. To capture this, you would add a firm-specific, but time-invariant, random shock, u_i, so that the problem becomes

5) $$I_{it} - \bar{I}_i = \beta_0 + \beta_1\left(F_{i,t-1} - \bar{F}_i\right) + \beta_2\left(K_{i,t-1} - \bar{K}_i\right) + \left(u_i + \varepsilon_{it}\right)$$

To estimate this in R, you must Install and then Load the linear and nonlinear mixed effects model, or **nlme**, package.

Once the package is installed, the model can be estimated as follows. First, the argument **`random=~1`** indicates that random effect is only in the intercept, and **`| FIRM`** specifies the factor

R Code:

```
> xtrereg <- lme(I~F+K,data=mypanel,random=~1| FIRM)
> sumre <- summary(xtrereg)

       Linear mixed-effects model fit by REML
        Data: mypanel
              AIC      BIC    logLik
         2205.851 2222.267 -1097.926

       Random effects:
        Formula: ~1 | FIRM
                (Intercept) Residual
       StdDev:    85.83119 52.73922

       Fixed effects: I ~ F + K
                       Value Std.Error  DF   t-value p-value
       (Intercept) -57.86442 29.377757 188 -1.969668  0.0503
       F             0.10979  0.010527 188 10.429581  0.0000
       K             0.30819  0.017171 188 17.947893  0.0000
        Correlation:
          (Intr) F
       F -0.328
       K -0.019 -0.368

       Standardized Within-Group Residuals:
               Min          Q1         Med          Q3         Max
       -3.43192891 -0.34984287  0.02104578  0.35919156  4.81447454

       Number of Observations: 200
       Number of Groups: 10
```

STATA Translation:

```
xtreg i f k, re

       Random-effects      GLS regression      Number of obs =      200
       Group variable      (i): firm    Number of groups    =      10

       R-sq:  within = 0.7668      Obs per group: min  =      20
       between       = 0.8196      avg    =      20.0
       overall       = 0.8061      max    =      20

       Random effects      u_i ~ Gaussian      Wald chi2(2)  =      657.67
       corr(u_i, X)  = 0 (assumed) Prob > chi2   =      0.0000

       i      Coef.         Std. Err.     z      P>z    [95% Conf.    Interval]

       f         .1097811     .0104927   10.46  0.000      .0892159    .1303464
       k         .308113      .0171805   17.93  0.000      .2744399    .3417861
       _cons  -57.83441     28.89893     -2.00  0.045  -114.4753     -1.193537
```

You'll see that the coefficients and standard errors produced by R and STATA are not the same, but they are close. One hint as to why this might be the case is that R uses a Maximum Likelihood method to solve this (I peaked at the help for **`lme`**), while if you look at the top line of the STATA output, the estimator used is (Feasible) Generalized Least Squares (GLS). So, you can't expect the output to be identical. With that in mind, how about applying the Hausman test?

Conducting the Hausman Test

In STATA it's really easy to apply the Hausman specification test to help you decide whether you should use fixed or random effects. However, recall that STATA's memory is just about as short as the last thing you told it to do. So, to run the Hausman test, you actually need to re-run the fixed-effects regression command, then you should store the information about the estimation in something called fixed (**`est store fixed`**), then you estimate the analogous random-effects regression model, and finally, you can apply the Hausman test to the object you called fixed

```
xtreg i f k, fe
est store fixed
xtreg i f k, re
hausman fixed
```

```
             (b)          (B)          (b-B)        sqrt(diag(V_b-V_B))
             Fixed        .            Difference   S.E.
       f     .1101238     .1097811     .0003427     .0055213
       k     .3100653     .308113      .0019524     .0024516

       b = consistent under Ho and Ha; obtained from xtreg
       B = inconsistent under Ha, efficient under Ho; obtained from xtreg

       Test:  Ho: difference in coefficients not systematic
              chi2(2)      =      (b-B)'[(V_b-V_B)^(-1)](b-B)
                     =     2.33
       Prob>chi2    =     0.3119
```

Since the model has two right-hand-side variables the theoretical Chi-Square value can be obtained by looking up a Chi-Square table for values associated with two degrees of freedom. STATA computes a test statistic is 2.33, which lies between the value associated with the 50% level of significance (1.38629) and the 90% level of significance (4.60517). This is much lower than the value associated with the 95% significance level (7.81473). In fact, STATA computes the exact p-value of 0.3119, and you'd probably want it to be about 0.05 or lower in most cases. So, you cannot reject the null hypothesis that the difference in the coefficients is not systematic, i.e., fixed effects and random effects produce more or less the same thing. Put another way, you can get away with using random effects. That said, Baltagi (2001) notes that it is a mistake to stop with just the Hausman test as there are other tests described in the book that should be considered too. Also, in your own work, ask yourself intuitively if your data is a random sample (random-effect) from a larger population or are you actually interested in just those subjects (fixed-effect). Also, ask yourself if there might be omitted variables that might affect both the dependent and at least one independent variable.

Can the same thing be done in R? Well, R does not have this test built in, but here is how to construct the test in R. The test statistic is often expressed in matrix form, which, unless you have a natural tendency to think in terms of the Matrix, can be awkward. So, I'll stop take a brief detour to explain the test statistic. You'll often see the test statistic written something like

$$
\begin{aligned}
6) \qquad m &= \left(b_{re} - b_{fe}\right)' \Sigma^{-1} \left(b_{re} - b_{fe}\right) \\
&= \underbrace{\left(b_{re} - b_{fe}\right)'}_{1\times j} \underbrace{\left(V\left(b_{fe}\right) - V\left(b_{re}\right)\right)}_{j\times j}{}^{-1} \underbrace{\left(b_{re} - b_{fe}\right)}_{j\times 1}
\end{aligned}
$$

where *j* is the number of variables, excluding dummy variables and the intercept, *'* is the transpose, the –1 refers to the inverse of the matrix, the *fe* and *re* subscripts refer to fixed effects and random effects, respectively. So, the statistic *m* is just the test statistic, a number obtained from potentially many operations. First, I'll show you what this would look like in this example

$$
7) \qquad m = \underbrace{\left(b_{F,re} - b_{F,fe} \quad b_{K,re} - b_{K,fe}\right)'}_{1\times 2} \underbrace{\begin{pmatrix} V_{b_F}^{fe} - V_{b_F}^{re} & V_{b_F,b_K}^{fe} - V_{b_F,b_K}^{re} \\ V_{b_F,b_K}^{fe} - V_{b_F,b_K}^{re} & V_{b_K}^{fe} - V_{b_K}^{re} \end{pmatrix}^{-1}}_{2\times 2} \underbrace{\begin{pmatrix} b_{F,re} - b_{F,fe} \\ b_{K,re} - b_{K,fe} \end{pmatrix}}_{2\times 1}
$$

where the subscripts F and K represent the coefficients for firm value and capital stock, respectively. You can verify for yourself what these components actually look like below. The terms V_{b_F,b_K}^{fe} and V_{b_F,b_K}^{re} are the diagonal components of the fixed effects and random effects estimators' respective covariance matrix, and V_{b_F,b_K}^{fe}, V_{b_F,b_K}^{re} are the off-diagonal components of the fixed effects and random effects estimator's covariance matrix. To compute this, first run the Least Squares Dummy Variable model, and save the coefficients, and do the same for the corresponding random effects model.

```
> xtfereg <- lm(I~factor(FIRM)+F+K,data=mypanel)
> sumfe <- summary(xtfereg)
> fecoef <- sumfe$coefficients
> fecov <- vcov(xtfereg)
```

```
                    Estimate  Std. Error     t value     Pr(>|t|)
(Intercept)     -70.2967175 49.70795884 -1.41419441 1.589588e-01
factor(FIRM)2   172.2025312 31.16125808  5.52617390 1.080747e-07
factor(FIRM)3  -165.2751236 31.77556202 -5.20132810 5.142745e-07
factor(FIRM)4    42.4874229 43.90987577  0.96760517 3.344847e-01
factor(FIRM)5   -44.3200953 50.49225679 -0.87776024 3.811942e-01
factor(FIRM)6    47.1354223 46.81068479  1.00693725 3.152592e-01
factor(FIRM)7     3.7432439 50.56492909  0.07402846 9.410664e-01
factor(FIRM)8    12.7510602 44.05262731  0.28945062 7.725555e-01
factor(FIRM)9   -16.9255550 48.45326669 -0.34931711 7.272422e-01
factor(FIRM)10   63.7288739 50.33023205  1.26621459 2.070030e-01
F                 0.1101238  0.01185669  9.28790117 3.921108e-17
K                 0.3100653  0.01735450 17.86656439 2.220007e-42
```

```
> xtrereg <- lme(I~F+K,data=mypanel,random=~1| FIRM)
> sumre <- summary(xtrereg)
> recoef <- sumre$tTable
> recov <- vcov(xtrereg)
```

```
                  Value   Std.Error  DF   t-value      p-value
(Intercept) -57.8644245 29.37775735 188 -1.969668 5.034528e-02
F             0.1097897  0.01052676 188 10.429581 2.192183e-20
K             0.3081881  0.01717127 188 17.947893 1.293819e-42
```

Having estimated the fixed and random effects models, it is now time to construct the Hausman test statistic. The R code is meant to get the numbers needed to fill in

equation 7). So, I have to collect rows 11 and 12 of the **xtfereg** coefficients and then rows and columns 11 and 12 of the **xtfereg** covariance matrix. I then collect the fixed effects coefficients and covariance matrix generated by the Maximum Likelihood (MLE) random effects estimator.[16]

```
> bfe <- coef(xtfereg)[11:12]
> Vfe <- vcov(xtfereg)[11:12,11:12]
> bre <- fixef(xtrereg)[-1]
> Vre <- summary(xtrereg)$varFix[-1,-1]        # or you could type
> Vre <- vcov(xtrereg)[-1,-1]
```

Once I have extracted that information from the regressions I can then begin to build the Hausman test statistic in equation 7). First I have to compute the difference between the random and fixed effects coefficients for F and K, as well as the transpose of that. Then I have to compute the difference between the analogous components of the covariance matrix for the fixed and random effects. Then I can compute the statistic by inverting the difference between the two covariance matrices and then pre- and post-multiplying it by the difference in the coefficients, where the percent signs surrounding the star, **%*%**, tells R to do a matrix multiplication, instead of a simple multiplication, and the **solve()** command computes the inverse of the matrix

```
> bdifft <- bre-bfe
> bdiff <- t(bre-bfe)
> Vdiff <- (Vfe-Vre)
> m <- bdiff%*%solve(Vdiff)%*%bdifft
> pvalue <- 1 - pchisq(m,df=2)
> Hausmantable <- c("Hausman"=m,"P-Value"=pvalue)
> Hausmantable

    Hausman   P-Value
1.7916645 0.4082677
```

You see that this Hausman test statistic, generated by a MLE method is slightly lower than the one STATA computes using the GLS esimator. However, in each case you can't reject the null hypothesis that the fixed and random effects output are systematically different. In other words, they produce essentially the same thing, and you can get away with using random effects. But again, perhaps a better way is to ask if the application you are considering is an analysis of a random sample, or if the subjects are of particular interest.

[16] After doing a search on "nlme random effects hausman" I found two web-sites that confirmed what I was doing was correct: http://jackman.stanford.edu/classes/350C/hausman.r http://research.bus.wisc.edu/jfrees/Book/BookAnalysisR/Chap7AnalysisR.txt.

Seemingly Unrelated

Another useful estimator you might consider is: Zellner's (1961) Seemingly Unrelated (or SUR). SUR allows you to have different right-hand side variables, and the slopes and intercepts might differ across equations, hence you can capture different elasticities. For instance, your system of equations might look like this, where the *y*'s, *x*'s are time series vectors, the *b*'s are coefficients and the *e*'s are vectors of errors

8)
$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} b_{10} & 0 & b_{12}x_{12} & 0 \\ b_{20} & b_{21}x_{21} & b_{22}x_{22} & b_{23}x_{23} \\ b_{30} & 0 & b_{32}x_{32} & b_{33}x_{33} \\ b_{40} & b_{41}x_{41} & b_{42}x_{42} & 0 \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{pmatrix}$$

In general you might estimate the following

9)
$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} b_{10} & b_{11}x_{11} & b_{12}x_{12} & b_{13}x_{13} \\ b_{20} & b_{21}x_{21} & b_{22}x_{22} & b_{23}x_{23} \\ b_{30} & b_{31}x_{31} & b_{32}x_{32} & b_{33}x_{33} \\ b_{40} & b_{41}x_{41} & b_{42}x_{42} & b_{43}x_{43} \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{pmatrix}$$

With SUR you now have the cross-equation correlation between errors to think about

$$\begin{pmatrix} \mathrm{cov}(e_1) & \mathrm{cov}(e_1,e_2) & \mathrm{cov}(e_1,e_3) & \mathrm{cov}(e_1,e_4) \\ \mathrm{cov}(e_2,e_1) & \mathrm{cov}(e_2) & \mathrm{cov}(e_2,e_3) & \mathrm{cov}(e_2,e_4) \\ \mathrm{cov}(e_3,e_1) & \mathrm{cov}(e_3,e_2) & \mathrm{cov}(e_3) & \mathrm{cov}(e_3,e_4) \\ \mathrm{cov}(e_4,e_1) & \mathrm{cov}(e_4,e_2) & \mathrm{cov}(e_4,e_3) & \mathrm{cov}(e_4) \end{pmatrix}$$

and this is where the seemingly unrelated comes from, since you only see the actual relationship through the errors. Also, the *b*'s will be functions of the regression standard errors. To estimate this in R, first Install and then Load the **`systemfit`** package, and then run the following codes

R Code:

```
> mypanel <- read.table("C:/grunfeld.csv",header=T,sep=",")
> model <- I ~ F + K
> SUR <- systemfitClassic("SUR",model,"FIRM","YEAR",data=
mypanel,rcovformula=0)
> summary(SUR)
```

This first matrix is the starting values of the error covariance matrix so that…

```
       1      2      3      4      5      6      7      8      9      10
1  7160.3 -1967.0 607.5 -282.8 -217.5  73.9  371.9  126.2  146.3 -21.7
2 -1967.0  7904.7 978.5  367.8  162.8 241.7 -211.8  511.5  208.2  62.4
3   607.5   978.5 660.8  -21.4   -3.4  88.2   -4.1  176.4   89.3  14.4
4  -282.8   367.8 -21.4  149.9    5.8  18.4  -12.9   13.3    7.2   1.4
5  -217.5   162.8  -3.4    5.8   69.8  13.2   10.6    0.3  -10.0   1.0
6    73.9   241.7  88.2   18.4   13.2  55.5    8.7   36.3  -10.8   2.9
7   371.9  -211.8  -4.1  -12.9   10.6   8.7   75.4   11.3   13.9  -1.9
8   126.2   511.5 176.4   13.3    0.3  36.3   11.3   88.7   41.4   5.4
9   146.3   208.2  89.3    7.2  -10.0 -10.8   13.9   41.4   70.4   2.2
10  -21.7    62.4  14.4    1.4    1.0   2.9   -1.9    5.4    2.2   1.0
```

you can can get to the final error covariance matrix…

```
       1      2      3      4      5      6      7      8      9      10
1  7227.2 -2226.6  583.8 -320.8 -222.4  61.5  405.3  135.7 157.3 -35.8
2 -2226.6  7980.8 1282.4  418.9  213.3 312.7 -228.0  609.1 248.1  69.9
3   583.8  1282.4  707.5    3.7   -4.4  96.6   -8.9  204.1 103.7  16.2
4  -320.8   418.9    3.7  154.0   15.5  22.7  -15.8   16.8  10.5   2.8
5  -222.4   213.3   -4.4   15.5   71.7  19.2    9.8    6.3  -9.7   1.5
6    61.5   312.7   96.6   22.7   19.2  58.4    7.4   41.1  -8.9   3.6
7   405.3  -228.0   -8.9  -15.8    9.8   7.4   76.4   11.8  14.5  -2.5
8   135.7   609.1  204.1   16.8    6.3  41.1   11.8   96.8  45.9   6.1
9   157.3   248.1  103.7   10.5   -9.7  -8.9   14.5   45.9  72.5   1.9
10  -35.8    69.9   16.2    2.8    1.5   3.6   -2.5    6.1   1.9   1.1
```

This third matrix is the error correlation matrix…

```
       1      2      3      4      5      6      7      8      9      10
1   1      -0.29    0.26 -0.30  -0.31  0.09   0.55  0.16    0.22  -0.41
2  -0.29    1       0.54  0.38   0.28  0.46  -0.29  0.69    0.33   0.76
3   0.26    0.54    1      0.01  -0.02  0.48  -0.04  0.78    0.46   0.59
4  -0.30    0.38    0.01  1       0.15  0.24  -0.15  0.14    0.10   0.22
5  -0.31    0.28   -0.02  0.15   1      0.30   0.13  0.08   -0.13   0.18
6   0.09    0.46    0.48  0.24   0.30  1       0.11  0.55   -0.14   0.45
7   0.55   -0.29   -0.04 -0.15   0.13  0.11   1      0.14    0.19  -0.28
8   0.16    0.69    0.78  0.14   0.08  0.55   0.14  1       0.55   0.61
9   0.22    0.33    0.46  0.10  -0.13 -0.14   0.19  0.55    1      0.22
10 -0.41    0.76    0.59  0.22   0.18  0.45  -0.28  0.61    0.22   1
```

Now you can get to the output, and you will notice that across the ten equations, the intercepts and slopes are different, as are the coefficient and regression standard errors. Finally, you may notice that the first equation is quite different from the others, at least from the perspective of the intercept. This may be because firm 1 is an "outlier", but unlike fixed effects, the damage here is contained within that equation.

```
The determinant of the residual covariance matrix: 2.07996e+19
OLS R-squared value of the system: 0.853443
McElroy's R-squared value for the system: 0.884686

      SUR estimates for '1' (equation 1)
      Model Formula: I.1 ~ F.1 + K.1
                    Estimate Std. Error   t value Pr(>|t|)
      (Intercept) -135.606136  72.293585  -1.87577 0.077967   .
      F.1            0.113814   0.016746  6.796643    3e-06 ***
      K.1            0.386124   0.029738 12.984106        0 ***
      ---
      Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

      Residual standard error: 92.209217 on 17 degrees of freedom
      Number of observations: 20 Degrees of Freedom: 17
      SSR: 144543.176447 MSE: 8502.539791 Root MSE: 92.209217
      Multiple R-Squared: 0.92062 Adjusted R-Squared: 0.911281

      SUR estimates for '2' (equation 2)
      Model Formula: I.2 ~ F.2 + K.2
                   Estimate Std. Error   t value Pr(>|t|)
      (Intercept) -10.905983   82.50626 -0.132184 0.896391
      F.2            0.162766   0.040113   4.05772 0.000818 ***
      K.2            0.340626   0.101623  3.351877 0.003782  **
      ---
      Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

      Residual standard error: 96.898023 on 17 degrees of freedom
      Number of observations: 20 Degrees of Freedom: 17
      SSR: 159616.855029 MSE: 9389.226766 Root MSE: 96.898023
      Multiple R-Squared: 0.465763 Adjusted R-Squared: 0.402911
```

```
SUR estimates for '3' (equation 3)
Model Formula: I.3 ~ F.3 + K.3
             Estimate Std. Error   t value Pr(>|t|)
(Intercept) -15.895901  20.730655 -0.766782 0.453728
F.3           0.034963   0.009346  3.740922 0.001627  **
K.3            0.12573   0.020424  6.155995  1.1e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 28.850037 on 17 degrees of freedom
Number of observations: 20 Degrees of Freedom: 17
SSR: 14149.519082 MSE: 832.324652 Root MSE: 28.850037
Multiple R-Squared: 0.684505 Adjusted R-Squared: 0.647388

SUR estimates for '4' (equation 4)
Model Formula: I.4 ~ F.4 + K.4
            Estimate Std. Error   t value Pr(>|t|)
(Intercept) 1.804327  11.002599  0.163991 0.871672
F.4         0.067844   0.015981   4.24524 0.000546 ***
K.4         0.307553   0.025363 12.126172        0 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 13.46178 on 17 degrees of freedom
Number of observations: 20 Degrees of Freedom: 17
SSR: 3080.731697 MSE: 181.219512 Root MSE: 13.46178
Multiple R-Squared: 0.911177 Adjusted R-Squared: 0.900727

SUR estimates for '5' (equation 5)
Model Formula: I.5 ~ F.5 + K.5
            Estimate Std. Error  t value Pr(>|t|)
(Intercept) 26.46736   6.065146 4.363845 0.000423 ***
F.5         0.127447   0.045832 2.780763 0.012813   *
K.5         0.011987    0.01792 0.668912 0.512534
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.181756 on 17 degrees of freedom
Number of observations: 20 Degrees of Freedom: 17
SSR: 1433.178947 MSE: 84.304644 Root MSE: 9.181756
Multiple R-Squared: 0.672092 Adjusted R-Squared: 0.633515

SUR estimates for '6' (equation 6)
Model Formula: I.6 ~ F.6 + K.6
             Estimate Std. Error  t value Pr(>|t|)
(Intercept) -6.193451   3.439334 -1.80077 0.089506   .
F.6          0.133311   0.017792  7.49289    1e-06 ***
K.6          0.054005   0.059739 0.904023 0.378616
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.287174 on 17 degrees of freedom
Number of observations: 20 Degrees of Freedom: 17
SSR: 1167.513224 MSE: 68.677248 Root MSE: 8.287174
Multiple R-Squared: 0.949687 Adjusted R-Squared: 0.943767
```

```
SUR estimates for '7' (equation 7)
Model Formula: I.7 ~ F.7 + K.7
             Estimate Std. Error   t value Pr(>|t|)
(Intercept) -9.770131   8.763981 -1.114805 0.280447
F.7          0.113465   0.045716  2.481938 0.023811   *
K.7           0.12818   0.014621   8.76692        0 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.479343 on 17 degrees of freedom
Number of observations: 20 Degrees of Freedom: 17
SSR: 1527.585138 MSE: 89.857949 Root MSE: 9.479343
Multiple R-Squared: 0.760335 Adjusted R-Squared: 0.732139

SUR estimates for '8' (equation 8)
Model Formula: I.8 ~ F.8 + K.8
            Estimate Std. Error  t value Pr(>|t|)
(Intercept) 3.149097   5.056158 0.622824 0.541666
F.8         0.053701   0.008265 6.497604    5e-06 ***
K.8         0.043362   0.034586 1.253753   0.2269
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.670396 on 17 degrees of freedom
Number of observations: 20 Degrees of Freedom: 17
SSR: 1935.574958 MSE: 113.85735 Root MSE: 10.670396
Multiple R-Squared: 0.72105 Adjusted R-Squared: 0.688232

SUR estimates for '9' (equation 9)
Model Formula: I.9 ~ F.9 + K.9
             Estimate Std. Error   t value Pr(>|t|)
(Intercept) -3.156864    7.29719 -0.432614 0.670734
F.9          0.076595   0.021057  3.637549 0.002036 **
K.9          0.065425    0.02195  2.980576 0.008395 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.237338 on 17 degrees of freedom
Number of observations: 20 Degrees of Freedom: 17
SSR: 1450.583062 MSE: 85.328415 Root MSE: 9.237338
Multiple R-Squared: 0.655242 Adjusted R-Squared: 0.614682

SUR estimates for '10' (equation 10)
Model Formula: I.10 ~ F.10 + K.10
             Estimate Std. Error   t value Pr(>|t|)
(Intercept)   1.98935   1.177681  1.689209 0.109434
F.10        -0.016129   0.015746 -1.024321  0.32004
K.10         0.376847   0.057306  6.576083    5e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.117089 on 17 degrees of freedom
Number of observations: 20 Degrees of Freedom: 17
SSR: 21.214088 MSE: 1.247888 Root MSE: 1.117089
Multiple R-Squared: 0.622001 Adjusted R-Squared: 0.577531
```

STATA Translation:

```
clear
insheet using "C:\grunfeld.csv", comma
browse
reshape wide i f k, i(year) j(firm)
sureg (i1 f1 k1) (i2 f2 k2) (i3 f3 k3) (i4 f4 k4) (i5 f5 k5) (i6 f6 k6) (i7
f7 k7) (i8 f8 k8) (i9 f9 k9) (i10 f10 k10)
```

This last line actually gives you the same output as R. I'll not report it, but instead will show you STATA output corrected for the sample size, since there are a couple

of options for you to choose to adjust for small samples. I'll get to the small sample code and output shortly, but first take note of the third line that begins with the command **reshape**. You may ask youself why is that line necessary, and why did I put **browse** in front of it? The reason is that I'd like for you to take note that the data is stacked as follows

firm	**year**	**i**	**f**	**k**
1	1935		…	
…				
1	1954		…	
2	1935		…	
…				
2	1954		…	
…				
10	1935		…	
…				
10	1954		…	

STATA can't run SUR on data arranged like that. I don't know why exactly, but to run the estimation, I have to "unstack" the data. It will still be stacked but only in the time dimension, as the firm-specific data will be rearranged parallel-wise using

```
reshape wide i f k, i(year) j(firm)
```

which actually transforms the data so that it comes out looking like this

	for firm 1			*for firm 2*				*for firm 10*		
year	i1	f1	k1	i2	f2	k2	…	i10	f10	k10
1935							…			
…										
1954							…			

What it does is take the variable **firm**, and makes firm-specific subscripts for the variables **i**, **f**, and **k**. Formatted this way, you can run **sureg** in STATA, but with 20 observations per firm, I'd add on the small sample option "**, dfk**", which adjusts the standard errors for the sample size (think small sample, less information, hence slightly larger standard errors), as well as "**corr**" to get the correlation matrix

```
sureg (i1 f1 k1) (i2 f2 k2) (i3 f3 k3) (i4 f4 k4) (i5 f5 k5) (i6 f6 k6) (i7
f7 k7) (i8 f8 k8) (i9 f9 k9) (i10 f10 k10), dfk corr
```

```
Seemingly unrelated regression
Equation     Obs    Parms  RMSE    "R-sq" chi2   P
i1     20     2      85.01269      0.9206 242.20 0.0000
i2     20     2      89.33556      0.4658 27.85  0.0000
i3     20     2      26.59842      0.6845 44.01  0.0000
i4     20     2      12.41115      0.9112 177.48 0.0000
i5     20     2      8.465161      0.6721 32.51  0.0000
i6     20     2      7.640396      0.9497 380.77 0.0000
i7     20     2      8.739524      0.7603 68.09  0.0000
i8     20     2      9.837619      0.7210 65.50  0.0000
i9     20     2      8.516405      0.6552 34.68  0.0000
i10    20     2      1.029905      0.6220 37.30  0.0000
```

```
             Coef.         Std. Err.     z      P>z    [95% Conf.    Interval]

       f1         .1138135   .0181631     6.27  0.000      .0782145     .149413
       k1         .3861235   .0322556    11.97  0.000      .3229038     .449343
       _cons  -135.6061     78.4134       -1.73  0.084  -289.2935      18.08134

       f2         .1627658   .0435082     3.74  0.000      .0774912     .248040
       k2         .3406261   .1102251     3.09  0.002      .1245889     .556663
       _cons   -10.90599    89.4906       -0.12  0.903  -186.3043     164.4924

       f3         .0349626   .0101371     3.45  0.001      .0150942     .054831
       k3         .1257302   .022153      5.68  0.000      .0823112     .169149
       _cons   -15.8959     22.48555      -0.71  0.480   -59.96677     28.17497

       f4         .0678437   .017334      3.91  0.000      .0338698     .101818
       k4         .3075528   .0275097    11.18  0.000      .2536347     .361471
       _cons     1.804334   11.93399       0.15  0.880   -21.58586     25.19453

       f5         .1274473   .0497115     2.56  0.010      .0300145     .224880
       k5         .0119871   .0194373     0.62  0.537     -.0261094     .050084
       _cons    26.46736     6.578575     4.02  0.000    13.57359     39.36113

       f6         .133311    .0192977     6.91  0.000      .0954879     .171134
       k6         .054005    .0647958     0.83  0.405     -.0729922     .181003
       _cons    -6.193452    3.730481    -1.66  0.097   -13.50506      1.118157

       f7         .1134649   .0495862     2.29  0.022      .0162776     .210652
       k7         .1281802   .0158586     8.08  0.000      .097098      .159263
       _cons    -9.770128    9.505873    -1.03  0.304   -28.4013       8.861041

       f8         .0537015   .0089644     5.99  0.000      .0361315     .071272
       k8         .0433622   .0375137     1.16  0.248     -.0301633     .116888
       _cons     3.149101    5.484173     0.57  0.566    -7.59968     13.89788

       f9         .0765949   .0228392     3.35  0.001      .0318308     .121359
       k9         .0654245   .0238084     2.75  0.006      .0187608     .112088
       _cons    -3.156863    7.914914    -0.40  0.690   -18.66981     12.35608

       f10       -.0161291   .017079     -0.94  0.345     -.0496034     .017345
       k10        .3768475   .0621568     6.06  0.000      .2550223     .498673
       _cons     1.98935     1.277374     1.56  0.119     -.5142579    4.492958

Correlation matrix of residuals:

     i1      i2      i3      i4      i5      i6      i7      i8      i9     i10
 i1   1
 i2  -0.26   1
 i3   0.28   0.43    1
 i4  -0.27   0.34   -0.07    1
 i5  -0.31   0.22   -0.02    0.06    1
 i6   0.12   0.36    0.46    0.20    0.21    1
 i7   0.51  -0.27   -0.02   -0.12    0.15    0.13    1
 i8   0.16   0.61    0.73    0.12    0.00    0.52    0.14    1
 i9   0.21   0.28    0.41    0.07   -0.14   -0.17    0.19    0.52    1
i10  -0.26   0.70    0.56    0.11    0.12    0.39   -0.22    0.57    0.26    1
```

Observe that across firms there are different coefficients and levels of significance. Compare this with the earlier result from random and fixed effects, without interaction effects. The intercepts range from –135 (firm 1) to over 26 (firm 5). The range of marginal impact of an additional million dollars in firm value is from –0.016 (firm 10) to over 0.16 (firm 2), compared with 0.11 that you saw for random and fixed effects without interaction. Finally, the marginal impacts from an additional million dollars in capital range from just over 0.01 (firm 5) to 0.38 (firm 1), instead of the basic random and fixed effects 0.31.

Dichotomous Limited Dependent Variable Techniques: Logit

Sometimes your dependent variable might be a dichotomous dummy variable, as when you want to estimate the probability or odds of something happening, given your data. If it's a continuous distribution you want, you can use the probit technique that estimates a (normal cumulative) probability distribution function. If you'd like to estimate the odds of one thing happening, estimate a log odds, or logit regression. While logit and probit are generally consistent, logit results are easier to interpret. As a useless example, assume you have data on initial income, the population growth (MRW's $n + g + d$ variable), investment and schooling. Try predicting the probability that the country is African. Here, I am less interested in the exact coefficients, which is good because of the multi-collinearity. To estimate the odds that, given the right hand side variables, the country in African ($d_i = 1$) or not ($d_i = 0$), the model is

10)

$$\ln\left(\frac{p(d_i=1)}{p(d_i=0)}\right) = \beta_0 + \beta_1 \ln(y_{1960,i}) + \beta_2 \ln(n_i + g + \delta) + \beta_3 \ln(inv_i) + \beta_4 \ln(sch_i) + \varepsilon_i$$

Since **mydata** is already in memory, type

R Code:

```
> logitreg <- glm(AFRICA~LGDP60+LNGD+LINV+LSCH,data=mydata,
family=binomial(link="logit"))
> logitsum <- summary(logitreg)
> prob <- logitreg$fitted.values
```

The last line computes the probability. Having just said not to pay attention to the estimated marginal impacts, I'll now tell you how you would interpret them if there was no multi-collinearity. The parameters for the right hand side variables measure the marginal impact on the log odds that the country is African. A positive (negative) coefficient means the marginal impact of an increase in the right hand side variable increases (decreases) the likelihood that the dependant variable equals one. Still the coefficients reported in the table below do not have a straightforward interpretation.

```
> logitsum

Call:
glm(formula = AFRICA ~ LGDP60 + LNGD + LINV + LSCH, family =
binomial(link = "logit"), data = mydata)
Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.97541  -0.27410  -0.05734   0.27205   2.47848

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  18.9360    10.7434   1.763 0.077973 .
LGDP60       -2.1271     0.8146  -2.611 0.009023 **
LNGD          4.0143     2.9493   1.361 0.173475
LINV          2.1131     0.9310   2.270 0.023225 *
LSCH         -3.1544     0.8409  -3.751 0.000176 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Dispersion parameter for binomial family taken to be 1)
    Null deviance: 141.809  on 109  degrees of freedom
Residual deviance:  54.507  on 105  degrees of freedom
AIC: 64.507

Number of Fisher Scoring iterations: 7
```

For a more straightforward interpretation of the coefficients, apply the exponential transformation in R by typing the following simple command, minus the intercept, since its interpretation is not intuitive

```
> oddratios <- exp(coef(logitreg)[-1])  # the [-1] removes the intercept

          LGDP60        LNGD        LINV        LSCH
      0.11918160 55.38439655  8.27421721  0.04266532
```

The following STATA document http://www.ats.ucla.edu/stat/stata/library/sg124.pdf gives you lots of details about interpreting limited dependant variable regression output. Using information from the table above, a unit increase in initial income reduces the odds of the country being African by 88.02%, or [0.11918 – 1]·100%. An acceleration of population growth increases the odds that the country is African by over 5400%, or [55 – 1]·100% (while it seems outrageous, it just means, definitely yes populations were accelerating across Africa). A unit increase in the investment share increases the odds that the country is African by 727%, or [8.27 – 1]·100%. On the other hand, a unit increase in schooling rates decreases by 95%, or [0.0427 – 1]·100%, reflecting low schooling rates across Africa. STATA produces the same output.

STATA Translation:

```
clear
set mem 100m
insheet using "C:\temple.csv", comma
drop lsch
generate LSCH=ln(school/100)
logit africa lgdp60 lngd linv LSCH
```

```
       [Reported Iterations Omitted]

       Logistic regression                             Number of obs = 110
                                                       LR chi2(4) = 87.30
                                                       Prob > chi2 = 0.0000
       Log likelihood = -27.253642                     Pseudo R2 = 0.6156

       Africa Coef.         Std. Err.     z      P>z    [95% Conf.    Interval]

       lgdp60 -2.12711        .8146113   -2.61  0.009  -3.723718      -.5305007
       lngd    4.014294      2.94924      1.36  0.173  -1.766109      9.794697
       linv    2.113147       .9310173    2.27  0.023    .2883867     3.937908
       lsch   -3.154368       .8409443   -3.75  0.000  -4.802589     -1.506148
       _cons  18.93607      10.7434        1.76  0.078  -2.120607     39.99274
```

or instead of using the canned logit command, you could do pretty much the same thing as what R did, using STATA's generalized linear model or **`glm`** function, specifying the "family" being a binomial distribution, and the link being **`logit`**

```
glm africa lgdp60 lngd linv lsch, family(binomial) link(logit)

[Reported Iterations Omitted]

      Generalized linear models                 No. of obs = 110
      Optimization     : ML                     Residual df = 105
                                                Scale parameter = 1
      Deviance = 54.50728474                    (1/df) Deviance = .519117
      Pearson  = 65.20591106                    (1/df) Pearson  = .6210087

      Variance function: V(u) = u*(1-u)         [Bernoulli]
      Link function    : g(u) = ln(u/(1-u))     [Logit]

                                                AIC = .5864299
      Log likelihood = -27.25364237             BIC = -439.0432

      africa Coef.         Std. Err.     z      P>z    [95% Conf.    Interval]

      lgdp60 -2.127109       .8146169   -2.61  0.009  -3.723728      -.5304889
      lngd    4.014292      2.949256     1.36  0.173  -1.766143      9.794726
      linv    2.113147       .9310195    2.27  0.023    .2883822     3.937912
      lsch   -3.154367       .8409486   -3.75  0.000  -4.802596     -1.506139
      _cons  18.93606      10.74346      1.76  0.078  -2.120738     39.99285
```

To get the odds ratios in STATA, just add the "**, or**" option at the end of the line

```
logit africa lgdp60 lngd linv lsch, or

      Iteration 0:   log likelihood = -70.904606
      Iteration 1:   log likelihood = -35.725849
      Iteration 2:   log likelihood = -29.496707
      Iteration 3:   log likelihood = -27.622049
      Iteration 4:   log likelihood = -27.271906
      Iteration 5:   log likelihood = -27.253704
      Iteration 6:   log likelihood = -27.253642

      Logistic regression                       Number of obs = 110
                                                LR chi2(4) = 87.30
                                                Prob > chi2 = 0.0000
      Log likelihood = -27.253642               Pseudo R2 = 0.6156

      africa Odds Ratio    Std. Err.     z      P>z    [95% Conf.    Interval]

      lgdp60   .1191813      .0970864   -2.61  0.009   .024144          .5883
      lngd   55.38418      163.3412      1.36  0.173   .170997      17938.37
      linv    8.274241       7.703461    2.27  0.023  1.334273        51.3111
      lsch     .0426653      .0358792   -3.75  0.000   .0082085         .2218
```

In STATA to compute the estimated country-specific probability that it is African, type

```
predict prob, pr
```

More Complicated Limited Dependent Variable Techniques: Ordered Logit

Finally, I have seen an area where STATA probably does better than R: ordered logit. STATA is also the only program that I am aware of that has generalized ordered logit. That's probably because there are many more STATA users than R users of these estimators, so the odds of observing a well-tested function are much higher. So, if you think your research may take you down this avenue, you might consider as part of the process of writing your dissertation writing an R function/package as well as a writeup in LaTex, which you would submit to the CRAN website. The ordered logit estimator makes it possible to estimate the odds of a limited dependent variable that can take on more than two possible values, where the values are ordered. Generalized ordered logit relaxes the assumption that the marginal effects are the same across thresholds, so it's sort of, but not exactly, like running quantile regressions but for limited dependent variables (i.e., ordered logit would be like OLS , while generalized ordered logit would be like quantile regressions). In other contexts, your multiple outcome limited dependent variable might be unordered, for instance, if you're trying to explain the odds of a red, green, blue, orange, or yellow ball being drawn from an urn, where colors have no preference. In an economic context, you might use this if you are estimating a utility function over several goods, specifically by identifying the utility maximum. In such cases, you might use the multinomial logit technique, which is a multivariate extension of the multinomial distribution. There are plenty of other limited dependent variable techniques to consider as well, depending on your application. Here I'll focus on ordered outcomes. For instance, perhaps you have different levels of quality ranging from A (being the highest) to D (being the lowest), and you want to explain the variation in quality. In this case the underlying variable, the qualitatitive score, is what's called a latent variable. That means it does not necessarily represent the truth, but it is a proxy for it. Alternatively, the ordered outcome may not be latent, but an actual count, such as number of time a person visits the doctor. Here you may use ordered probit or logit, but to be consistent with the previous section, consider ordered logit.

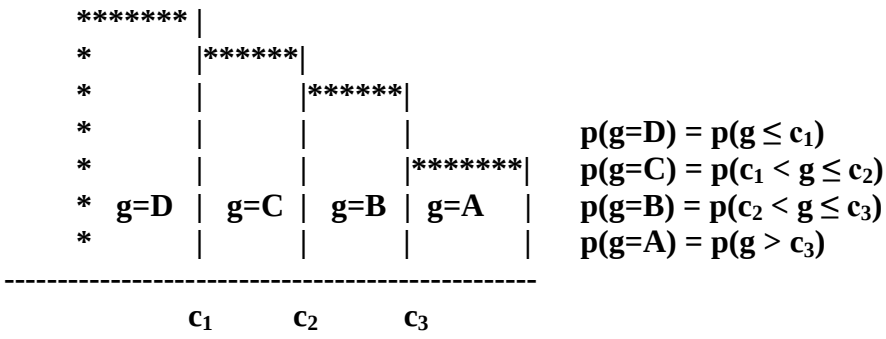
Ordered Logit Regression

The ordered logit regression technique used here is the most basic extension of the logit estimator for cases in which there is more than two possible values of the dependent variable, and when the order matters. As an example, consider the fact that the quality of a country's national income data may be related to the country's performance; i.e., due to capacity constraints, the poorer the country, the poorer the quality of its data. The quality of the country's data is taken from the Penn World Table http://pwt.econ.upenn.edu/Documentation/append61.pdf, and the table of numbers is provided in the appendix 4. Without a formal theory to test such a hypothesis, let's try first to to identify the association between the quality grade assigned to the data, and some of the right hand side variables used by Mankiw, Romer and Weil (1993). For instance, replacing the left-hand side variable in equation 2), a country's GDP growth between 1960 and 1985, with a country's national income data quality, and also dropping the investment variable, to get

$$11) \qquad \log\left(\frac{p(grade_i > c_k)}{p(grade_i \le c_k)}\right) = \beta_0 + \beta_1 \ln(y_{1960,i}) + \beta_2 \ln(n_i + g + \delta) + \beta_3 \ln(sch_i) + \varepsilon_i$$

This equation will make it possible to explore whether countries with higher initial income, faster population growth, or higher schooling rates increase the odds of having better quality data. Consider a generic graph for four outcomes

Figure 11. The Distribution of a Hypothetical Ordered Limited Dependent Variable



In Figure 11, c_1, c_2 and c_3 each tell you when you've reached the boundary between two thresholds. In Appendix 3 you'll see a slightly more formal presentation of the cut points in the discussion of ordered and generalized ordered logit. To estimate equation 11) in R, you first have to Load the MASS package, which is already in R's memory, but it needs to be activated. So you run

R Code:

```
> mydata <- read.table("C:/temple.csv",header=T,sep=",")
> gradedata <- read.table("C:/templegr.csv",header=T,sep=",")
> mynewgradedata <- merge(mydata,gradedata,by.x="country", by.y="country")
> mynewgradedata$grade <-
ordered(mynewgradedata$grade,levels=c("D","C","B","A"))
> ologitreg <- polr(factor(graden)~LGDP60+LNGD+LSCH,data=
mynewgradedata,method="logistic")
```

```
Call:
polr(formula = factor(graden) ~ LGDP60 + LNGD + LSCH, data =
mynewgradedata, method = "logistic")

Coefficients:
    LGDP60       LNGD       LSCH
 1.0712738 -4.4471456  0.6058889

Intercepts:
     1|2      2|3      3|4
15.83211 19.56957 20.63841

Residual Deviance: 199.7299
AIC: 211.7299
```

Unfortunately, R in this case cannot seem to produce a summary as there seems to be a problem with convergence of the Hessian matrix, which is the matrix of second partial derivatives for the information matrix for this maximum likelihood estimator.

```
> summary(ologitreg)

Re-fitting to get Hessian
Error in optim(start, fmin, gmin, method = "BFGS", hessian = Hess, ...) :
        initial value in 'vmmin' is not finite
```

I can still get the right coefficients by running

```
> ologitoddratios <- exp(coef(ologitreg))

          LGDP60       LNGD       LSCH
      2.91909541 0.01171195 1.83288077
```

I'll interpret the coefficients later when we get to STATA since it reports the same coefficients.

STATA Translation:

```
clear
set mem 100m
insheet using "C:\templegr.csv", comma
sort country
save "C:\gradedata.dta", replace
clear
insheet using "C:\temple.csv", comma
sort country
save "C:\mydata.dta", replace
sort country
merge country using "C:\gradedata.dta"
tab _merge
drop if _merge==1
drop _merge
sort country
tab grade, gen(d)
gen gradea = d1*4
gen gradeb = d2*3
gen gradec = d3*2
gen graded = d4
egen graden = rsum(gradea gradeb gradec graded)
drop d1-graded
drop lsch
generate lsch=ln(school/100)
save "C:\mymergegradedata.dta", replace
ologit graden lgdp60 lngd lsch

       [Reported Iterations Omitted]

       Ordered logistic    regression    Number of obs =      111
                     LR chi2(3)    =      59.58
                     Prob > chi2   =      0.0000
       Log likelihood =    -99.864954    Pseudo R2     =      0.2298

       graden   Coef.      Std. Err.     z      P>z    [95% Conf. Interval]

       lgdp60 1.071284      .3116099     3.44   0.001  .4605401     1.682028
       lngd   -4.44718      1.352637     -3.29  0.001  -7.098299    -1.79606
       lsch   .6058809      .3306531     1.83   0.067  -.0421873    1.253949

       /cut1  15.8323       4.689003                   6.642025     25.02258
       /cut2  19.56977      4.810637                   10.1411      28.99845
       /cut3  20.63861      4.902334                   11.03021     30.24701
```

```
ologit graden lgdp60 lngd lsch, or

       [Reported Iterations Omitted]

       Ordered logistic    regression    Number of obs =      111
                     LR chi2(3)    =      59.58
                     Prob > chi2   =      0.0000
       Log likelihood =    -99.864954    Pseudo R2     =      0.2298

       graden   Coef.      Std. Err.     z      P>z    [95% Conf. Interval]

       lgdp60 2.919126      .9096284      3.44   0.001  1.58493       5.37645
       lngd    .0117116     .0158415     -3.29  0.001   .0008265      .1659514
       lsch   1.832866      .6060428      1.83   0.067   .9586902     3.504154

       /cut1  15.8323       4.689003      6.642025      25.02258
       /cut2  19.56977      4.810637      10.1411       28.99845
       /cut3  20.63861      4.902334      11.03021      30.24701
```

The cut points just tell you when you've reached the next highest threshold, so `/cut1` is where you reach the threshold between zero and one, `/cut2` is the threshold between one and two, etc. Here, a one percent increase in initial income is associated with a 192% (or [2.92 – 1]·100%) increase in the odds that a country has better quality data. A one percent increase in population growth rate is associated with almost a 99% (or [0.012 – 1]·100%) decline in the odds that the country's data is given a higher grade. Finally, a percentage increase in average years of schooling is associated with an 83% (or [1.83 – 1]·100%) increase in the odds that the country's data is rated higher

Generalized Ordered Logit Regression

Compared to the previous result, now consider the possibility that the sensitivity of a country's data quality may vary across thresholds. For instance, if tax revenues are low in poorer countries, officials may not be able to divert sufficient resources to collect better data, and this will be reflected in the poorer grade. Fu (1998) has written a program to measure such differences across thresholds, and Williams (2006) has generalized this. Do a search from all sources in STATA for **`ologit`**. This will give a number of entries. From that list you will see the following

```
gologit from http://fmwww.bc.edu/RePEc/bocode/g
    'GOLOGIT': module to estimate generalized ordered logit models / The
    gologit command estimates regression models for ordinal / dependent
    variables. The actual values taken on by the dependent / variable are
    irrelevant except that larger values are assumed to / correspond to

gologit2 from http://fmwww.bc.edu/RePEc/bocode/g
    'GOLOGIT2': module to estimate generalized logistic regression models
    for ordinal dependent variables / gologit2 estimates generalized ordered
    logit models for ordinal / dependent variables. A major strength of
    gologit2 is that it can / also estimate three special cases of the
    generalized model:
```

You can download both. When you click on either one, you will see `click here to install` to the right. This will download the necessary commands to your computer (presuming you have admininstrative rights). The command to run Fu's **`gologit`** is

```
gologit graden lgdp60 lngd lsch, or

       [Reported Iterations Omitted]

       Generalized Ordered Logit Estimates     Number of obs = 111
                                               Model chi2(9) = 82.65
                                               Prob > chi2 = 0.0000
       Log Likelihood = -88.3304128            Pseudo R2 = 0.3187

       graden
              Odds Ratio   Std. Err.     z      P>z    [95% Conf.   Interval]

       mleq1
       lgdp60 1.077246     .4635424     0.17   0.863   .4634894     2.503742
       lngd   1.224681     2.041523     0.12   0.903   .046675     32.13377
       lsch   2.315016     .9092796     2.14   0.033  1.072071      4.999015

       mleq2
       lgdp60 3.750871     1.596502      3.11  0.002  1.628662      8.6384
       lngd    .002687     .0044603     -3.57  0.000   .0001038      .0695413
       lsch   5.881634     5.273871      1.98  0.048  1.014503     34.0991

       mleq3
       lgdp60 8.808661     7.313714      2.62  0.009  1.730488     44.83852
       lngd    .0068789     .0178907    -1.91  0.056   .000042      1.125476
       lsch   4.147469     5.457869      1.08  0.280   .3145163    54.69192
```

How about an interpretation? Well again, it's kind of like the ordered limited dependent variable version of quantile regressions in that you get a different slope coefficient for different thresholds, as I suggested earlier. In this sense we see that there is variation across thresholds. First, consider that mleq1, mleq2, and mleq3 reflect the probability of moving from grade D to C, D or C to B, and D, C or B to A, respectively. So for instance, a one percent increase in initial income (lgdp60) is hardly makes a difference in moving from the D to C threshold (only 7%). However, the odds increase by 275% in moving from D or C to B, and 780% in moving from D, C, or B to A. Next, while the odds of moving from D to C, increases slightly (although it's statistically insignificant) for a one percent increase in the population growth rate, the odds of a country having higher data quality drop sharply (over 99%) at the higher thresholds. Finally, higher schooling rates always seem to be associated with higher data quality, although there is significant variability of the results at the highest threshold (the move from D, C, or B to A).

Basic Time Series

A time dimension in the data was already introduced in the section on panel data, but time series analysis is a entire area of specialization in and of itself. Pfaff (2006) has written an excellent book on Times Series in R, which has inspired me in my choice of what to discuss here. I'll do some basic unit root tests and test for co-integration in R, ARMA-GARCH and R/S statistic for long range dependence. As far as data sets, I'll begin with the Johansen and Juselius's dataset, which I got from http://www.stanford.edu/~clint/bench/finnish.xls. However, once again, the data has no labels, so you can go to Johansen's web-site where he labels the data: http://www.math.ku.dk/~sjo/data/finnish_data.html. Johansen and Juselius (1991) try to estimate the cointegrating relationships in money demand for Finland from summer 1958 through fall 1984. They have 106 quarters of observations. I'll first show you how to how to prepare a time series object in R (as well as in STATA) and then how to conduct the Augmented Dickey-Fuller (ADF) test. While in your own work you should probably run the Phillips-Perron test, and consider any more recent tests, including the ones discussed by Pfaff (2006), the process is similar, so in the interest of space I'll just discuss the ADF test. Later, I'll report on an E-Views exercise I once wrote up as a teaching device to give you another example of how to do the cointegration analysis in practice.

Dickey-Fuller Tests

Typically, you start with an equation like

$$12) \qquad \Delta p_{it} = \beta_0 + \beta_1 t + \beta_2 p_{i,t-1} + \sum_{l=3}^{L} \beta_l \Delta p_{i,t-l+2} + \varepsilon_{it}$$

which characterizes a regression between changes in the variable *p* against a constant, a time trend, one lagged value of *p*, and *L* lags of the change in the variable *p*. The objective of the test is to determine if there is serial correlation in the residuals. The first thing you'd like to test is whether $\beta_2 = 0$ is zero or not. If it's statistically significantly different from zero you can reject the null that there's a unit root, and stop there. If the absolute value of the ADF test is greater than the absolute vale of the 1% critical value, then the hypothesis of non-stationarity is rejected, and the integrated series can be included in the co-integration analysis. However, you might simply have a random walk with a constant mean and no time trend. Then, you want to test the joint hypothesis that $\beta_1 = \beta_2 = 0$. Finally, you might have a pure random walk in which there is no trend and the constant is zero, in which case you test whether $\beta_0 = \beta_1 = \beta_2 = 0$. By taking first differences, stationarity is restored in the sense that the ADF tests values are now lower than the theoretical values at the five percent level. To run this test in R, you'll first have to Install and then Load the `urca` package.

In more recent versions of the `urca` package you can actually select the "optimal" number of lags to include in the regression by using one of three selection criteria. Here the the BIC, or "Bayes Information Criterion," is used as it controls for the number of lags included in the regression equation.

Results for The M1 Monetary Aggregate

R Code:

```
> johansen <- read.table("C:/johansen.csv",header=T,sep=",")
> lrm1 <- ts(johansen$lrm1, start=c(1958,2),end=c(1984,3),frequency=4)
> lrm1df.ct <- ur.df(lrm1, type = "trend", lags = 6,selectlags = "BIC")
> summary(lrm1df.ct)

       ###############################################
       # Augmented Dickey-Fuller Test Unit Root Test #
       ###############################################

       Test regression trend


       Call:
       lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)

       Residuals:
              Min          1Q      Median          3Q         Max
       -0.2139332 -0.0309009  0.0002432  0.0369146  0.1425951

       Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
       (Intercept)  0.4873168  0.1881050   2.591  0.01114 *
       z.lag.1     -0.1681760  0.0661381  -2.543  0.01267 *
       tt           0.0016737  0.0006817   2.455  0.01597 *
       z.diff.lag1 -0.1300637  0.1081278  -1.203  0.23211
       z.diff.lag2  0.0848455  0.1014305   0.836  0.40505
       z.diff.lag3 -0.2177778  0.1014727  -2.146  0.03449 *
       z.diff.lag4  0.3043583  0.0996525   3.054  0.00295 **
       ---
       Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

       Residual standard error: 0.05421 on 92 degrees of freedom
       Multiple R-squared: 0.4531,     Adjusted R-squared: 0.4174
       F-statistic:  12.7 on 6 and 92 DF,  p-value: 2.150e-10

       Value of test-statistic is: -2.5428 3.2245 3.2341

       Critical values for test statistics:
             1pct  5pct 10pct
       tau3 -3.99 -3.43 -3.13
       phi2  6.22  4.75  4.07
       phi3  8.43  6.49  5.47
```

You have to do a little eyeball work to interpret these results. Specifically, at the bottom of the output, you see
"`Value of test-statistic is: -2.5428 3.2245 3.2341`" and below that you see a table of theoretical values

Or I could reconstruct this table using the following commands

```
> cbind(t(lrm1df.ct@teststat),lrm1df.ct@cval)
           statistic  1pct  5pct 10pct
       tau3 -2.542799 -3.99 -3.43 -3.13
       phi2  3.224486  6.22  4.75  4.07
       phi3  3.234111  8.43  6.49  5.47
```

To translate what I just did, note that if you type `lrm1df.ct@teststat` you get a row of the test statistic values "`-2.5428 3.2245 3.2341`" but if you type `lrm1df.ct@teststat` you get a table. So, I'd like to join the row of values as a column to the previous table, so I use the transpose command, `t()`, and put it to the left of the table values.

This is the result for the model with a trend, but as long as I cannot reject the hypothesis that there is a unit root using the `tau3` statistic, I should continue with two more tests, first in which the model has only a non-zero constant, and then finally, I should test to see if it's a pure random walk without the trend, and the constant itself is be zero. Those results are

```
> lrm1df.c <- ur.df(lrm1, type = "drift", lags = 6,selectlags = "BIC")
> cbind(t(lrm1df.c@teststat),lrm1df.c@cval)

             statistic  1pct  5pct 10pct
      tau2 -0.6469693 -3.46 -2.88 -2.57
      phi1  1.7297178  6.52  4.63  3.81

> lrm1df.nc <- ur.df(lrm1, type = "none", lags = 6,selectlags = "BIC")
> cbind(t(lrm1df.nc@teststat),lrm1df.nc@cval)

           statistic  1pct  5pct 10pct
      tau1   1.67893 -2.58 -1.95 -1.62
```

You see that the none of the test statistics magnitudes, i.e., the absolute values, in the left column are greater than the magnitudes of the critical values at the 1 percent level. So it's time to go to differences, but I'll do the STATA translation first.

STATA Translation:

```
clear
set mem 100m
insheet using "C:\johansen.csv", comma
tsset date
```

Just as with R, some functions for STATA, such as the **`adfuller`**, **`pperron`**, and **`vec`**, require that you specify that your data is in fact a time series. This was the case in the panel data section, when I introduced the concept of time series setting the data, using the **`tsset`** command in order to use the **`xtreg`** command. Before you can do that you also have to let STATA know which variable is ordered chronologically. Unfortunately, when I read in the data, the last command, **`tsset date`**, will not run because STATA does not actually recognize the variable **`date`** as a time object, but instead it reads it as a string/character variable. I tried, to no avail, a number of different things to get STATA to read **`date`** as a time variable. Finally I gave up and did an internet search on "create + dates + stata" and found http://research.umbc.edu/economics/econ612/exercise2_1.pdf, and this lead me to write the following lines of code to create a new, quarterly time variable

```
display q(1958q2)
generate periods = _n
** you don't need this line, but it's just to see what STATA did **
browse
generate time = periods-8
format time %tq
tsset time
** you don't need this line, but it's just to see what STATA did **
browse
```

The first line tells you what is the numeric equivalent for the second quarter 1958, which is the first period for the Finnish data, and the answer is –7. This might seem strange, until you find out that January 1, 1960 (for daily data), January 1960 (for monthly data), first quarter 1960 (for quarterly data) and 1960 (for annual data) are

each equal to 0. Since I have quarterly data, I have to go back seven quarters to reach the second quarter 1958. With knowledge of the numerical equivalent of that date, I can create a new time variable, which I'll call **periods**. You might ask yourself, what's the "**_n**" in that command? If you run that second line and then **browse** or **edit** the data, you will see that STATA created a new variable from 1 to the maximum number of rows there are in the data, in this case 106. I could create a time variable out of this, except recall that observation 1 would be second quarter 1960 (since the first quarter 1960 equals zero). So, I have to subtract 8 from the variable **periods**, in order to begin at –7, which STATA will recognize as second quarter 1958. I call this new variable **time**, which equals **periods - 8**. This new variable ranges from –7 to 98. To create a STATA time variable from this, simply type **format time %tq**, which tells STATA that it is a quarterly time object. Now STATA has creates your time variable, so you can **tsset** (time series set) the data. Once this is done, you can run unit root tests to your heart's content.

In STATA you can either take the same approach as you saw for R, to determine how many lags to include when performing the ADF test, or you could use the command, **varsoc**, which gives you a number of information criteria to help choose. I begin with six lags as the maximum, and look at the Akaike Information Criteria (AIC) to determine how many lags to include.

```
varsoc lrm1, maxlag(6) exog(time)
```

```
Selection order criteria
Sample:  1959q4   1984q3                  Number of obs = 100
lag     LL      LR      df    p      FPE       AIC      HQIC      SBIC
----+--------------------------------------------------------------------
0  -21.5492                        .091915   .450984   .461528   .477036
1   124.251   291.6    1  0.000  .005078  -2.44501  -2.42392  -2.39291
2   137.589  26.677    1  0.000  .003967  -2.69178  -2.66015  -2.61362
3   138.977  2.7763    1  0.096  .003937  -2.69954  -2.65737  -2.59533
4   147.119  16.284    1  0.000  .003413  -2.84238  -2.78966  -2.71212
5   150.554  6.8693*   1  0.009  .003251* -2.89107* -2.82781* -2.73476*
6   151.503  1.8988    1  0.168  .003255  -2.89006  -2.81625   -2.7077
```

The AIC criteria, and the stars, seem to point to five lags as the best number to include. I'll run the test for five lags, and then show you that the STATA output for the ADF test produces essentially the same output as R's **urca()** package.

First, take a look at the ADF test for five lags.

```
dfuller lrm1, lags(5) trend regress
```

```
Augmented Dickey-Fuller test for unit root     Number of obs = 100
                    ---------- Interpolated Dickey-Fuller ---------
           Test         1% Critical       5% Critical      10% Critical
         Statistic           Value             Value            Value

Z(t)   -2.270              -4.040             -3.450        -3.150
MacKinnon approximate p-value for Z(t) = 0.4505
```

```
D.lrm1 Coef.        Std. Err.     t      P>t    [95% Conf.    Interval]

lrm1
L1.    -.1548821    .0682251     -2.27  0.026  -.2903829     -.0193812
LD.    -.1199797    .1084045     -1.11  0.271  -.3352804      .095321
L2D.    .0546685    .1082833      0.50  0.615  -.1603915      .2697285
L3D.   -.2237636    .1011125     -2.21  0.029  -.4245818     -.0229455
L4D.    .2809028    .1030158      2.73  0.008   .0763046      .485501
L5D.   -.0782641    .1033961     -0.76  0.451  -.2836177      .1270895
_trend  .0015411    .0007004      2.20  0.030   .0001502      .0029321
_cons   .4506341    .1936926      2.33  0.022   .0659438      .8353244
```

Now look at the ADF test with four lags to compare with R's output above.

dfuller lrm1, lags(4) trend regress

```
Augmented Dickey-Fuller test for unit root     Number of obs = 101
                     ---------- Interpolated Dickey-Fuller ---------
           Test         1% Critical       5% Critical      10% Critical
         Statistic           Value             Value            Value

Z(t)   -2.546              -4.040              -3.450        -3.150

MacKinnon approximate p-value for Z(t) = 0.3054

D.lrm1 Coef.        Std. Err.     t      P>t    [95% Conf.    Interval]

lrm1
L1.    -.1673291    .0657251     -2.55  0.013  -.2978279     -.0368303
LD.    -.1251288    .1071441     -1.17  0.246  -.337866       .0876083
L2D.    .0930794    .1002433      0.93  0.356  -.105956       .2921148
L3D.   -.2084082    .1001273     -2.08  0.040  -.4072133     -.0096031
L4D.    .3046843    .0982484      3.10  0.003   .1096098      .4997588
_trend  .0016357    .0006744      2.43  0.017   .0002966      .0029748
_cons   .4868339    .1870136      2.60  0.011   .115514       .8581537
```

You'll notice that this is very close to the regression output that R reports, and that R's tau statistic is the same thing as STATA's Z(t) statistic, although the critical values are slightly different. Since I'm not an expert in times series, I can't offer an explanation, but I can tell you that the results are consistent. See that the lagged level term **L1**, **_trend** and **_cons** are all statistically significant, and the `Z(t)` is not more negative than the 1% critical value, so you can't reject the null hypothesis that there is unit root in the money supply series. Still, I'll rerun the ADF test for the series without the constant and no trend, and then with no constant or trend, but only report the Z(t) statistic, by removing the **regress** option.

dfuller lrm1, lags(4) drift

```
Augmented Dickey-Fuller test for unit root     Number of obs = 101
                     ---------- Interpolated Dickey-Fuller ---------
           Test         1% Critical       5% Critical      10% Critical
         Statistic           Value             Value            Value
Z(t)   -0.760              -2.366              -1.661        -1.291
```

dfuller lrm1, lags(4) noconstant

```
Augmented Dickey-Fuller test for unit root     Number of obs = 101
                     ---------- Interpolated Dickey-Fuller ---------
           Test         1% Critical       5% Critical      10% Critical
         Statistic           Value             Value            Value
Z(t)   1.720               -2.600              -1.950        -1.610
```

So I should probably take the first difference to induce stationarity. If this works, then money growth can be called integrated of order 1, often denoted I(1). If it was in fact stationary right away, then it would have been integrated of order zero, or I(0). So, since the M1 series is in logs, the first difference will be the growth rate.
Next I can run the test for first differences in the natural log of money growth. First, you can compute the first difference, using the **`diff(lrm1)`** command, and then you can apply the Dickey-Fuller test.

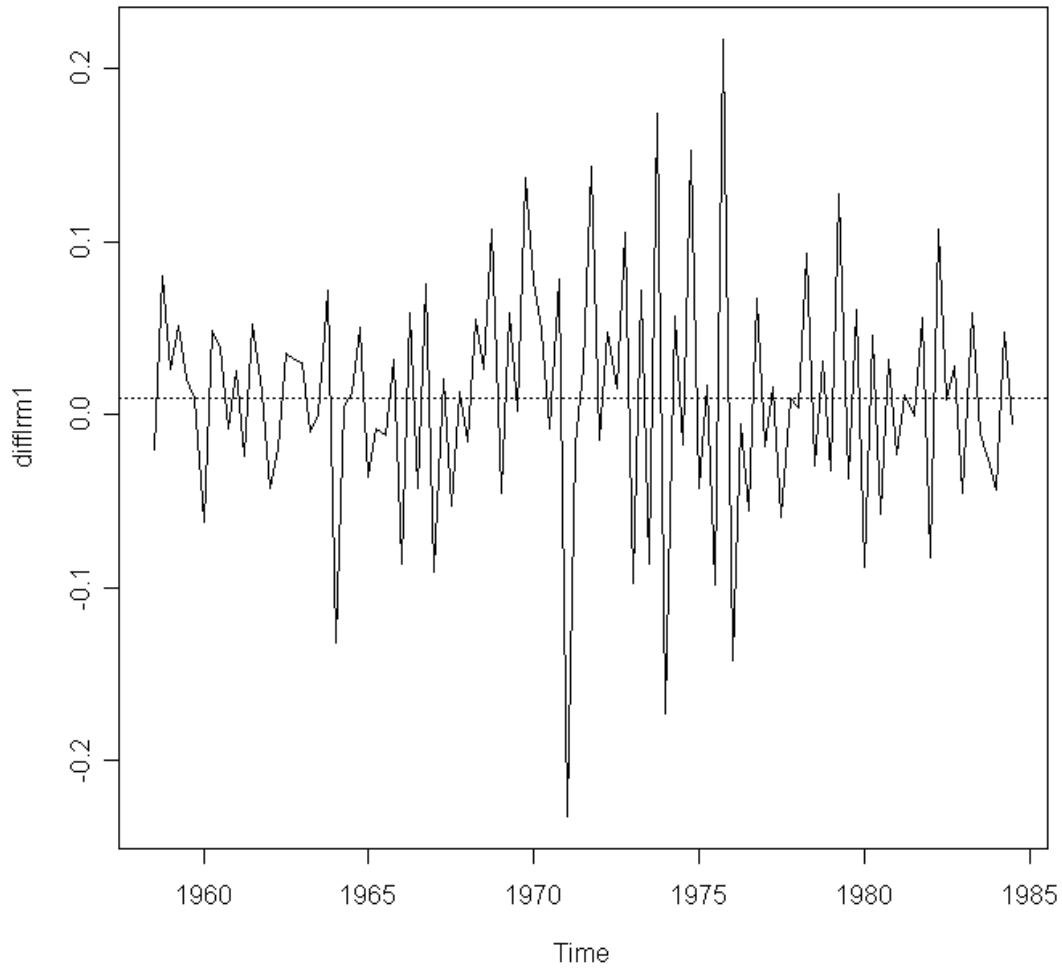
R Code:

```
> gm <- diff(lrm1)
> gm.ct <- ur.df(gm,type = "trend", lags = 6,selectlags = "BIC")
> cbind(t(gm.ct@teststat),gm.ct@cval)

           statistic  1pct  5pct 10pct
      tau3 -4.855305 -3.99 -3.43 -3.13
      phi2  7.872344  6.22  4.75  4.07
      phi3 11.808422  8.43  6.49  5.47

> gm.c <- ur.df(gm,type = "drift", lags = 6,selectlags = "BIC")
> cbind(t(gm.c@teststat),gm.c@cval)

           statistic  1pct  5pct 10pct
      tau2 -4.883229 -3.46 -2.88 -2.57
      phi1 11.923059  6.52  4.63  3.81

> gm.nc <- ur.df(gm,type = "none", lags = 6,selectlags = "BIC")
> cbind(t(gm.nc@teststat),gm.nc@cval)

           statistic  1pct  5pct 10pct
      tau1 -4.478248 -2.58 -1.95 -1.62
```

So this you can be pretty sure is stationary, around a constant mean. To get a visual sense of what this means just run the following plot command and the second line adds a horizontal dotted line, to help you visualize the cycle around a flat line, representing a constant, but non-zero mean.

```
> plot.ts(gm,start=c(1958,3),end=c(1984,3),frequency=4)
> abline(mean(gm),0,lty=3)
```

Figure 12. Finnish Money Growth: That looks pretty stationary!



STATA Translation:

```
gen gm = lrm1-lrm1[_n-1]
varsoc gm, maxlag(6) exog(time)
```

```
  Selection order criteria
  Sample:  1960q1   1984q3                  Number of obs = 99

lag     LL      LR      df    p      FPE       AIC      HQIC      SBIC
----+--------------------------------------------------------------------
0   121.855                       .005199  -2.42132  -2.40011   -2.3689
1   135.519  27.327    1  0.000  .004026  -2.67715  -2.64533  -2.59851
2   136.818  2.5977    1  0.107  .004002  -2.68319  -2.64076  -2.57834
3   145.037  16.438    1  0.000  .003459  -2.82902  -2.77599  -2.69796
4   148.361   6.649*   1  0.010    .0033* -2.87598* -2.81235*  -2.7187*
5   149.344   1.966    1  0.161  .003302  -2.87564   -2.8014  -2.69215
6   149.737  .78588    1  0.375  .003343  -2.86338  -2.77853  -2.65367
```

You see that again, the command suggests keeping in four lags, but to keep it comparable with R, I'll work down to three lags.

```
dfuller gm, lags(4) trend

      Augmented Dickey-Fuller test for unit root     Number of obs = 100

                        ---------- Interpolated Dickey-Fuller ---------
           Test         1% Critical       5% Critical      10% Critical
         Statistic           Value             Value            Value
Z(t)   -5.168              -4.040              -3.450       -3.150

dfuller gm, lags(3) trend

      Augmented Dickey-Fuller test for unit root     Number of obs = 101

                        ---------- Interpolated Dickey-Fuller ---------
           Test         1% Critical       5% Critical      10% Critical
         Statistic           Value             Value            Value
Z(t)   -5.012              -4.040              -3.450       -3.150

dfuller gm, lags(3) drift

      Augmented Dickey-Fuller test for unit root     Number of obs = 101

                        ---------- Interpolated Dickey-Fuller ---------
           Test         1% Critical       5% Critical      10% Critical
         Statistic           Value             Value            Value
Z(t)   -5.040              -2.366              -1.661       -1.290

dfuller gm, lags(3) noconstant

      Augmented Dickey-Fuller test for unit root     Number of obs = 101

                        ---------- Interpolated Dickey-Fuller ---------
           Test         1% Critical       5% Critical      10% Critical
         Statistic           Value             Value            Value
Z(t)   -4.657              -2.600              -1.950       -1.610
```

So, first I started off with a trend and then, since it wasn't statistically significant, I removed it, leaving just the non-zero constant, as well as the lagged level and differences. Now, I am pretty sure that the series is stationary, it is I(1). For the other series, I'll suppress the output, but will show you some codes, and the results only in the interest of time.

Results for GDP

So, quarterly GDP may have a unit root, but no trend.

R Code:

```
> lny <- ts(johansen$lny, start=c(1958,2),end=c(1984,3),frequency=4)
> lnydf.ct <- ur.df(lny,type = "trend", lags = 6,selectlags = "BIC")
> cbind(t(lnydf.ct@teststat),lnydf.ct@cval)

          statistic  1pct  5pct 10pct
      tau3 -1.387176 -3.99 -3.43 -3.13
      phi2  4.403456  6.22  4.75  4.07
      phi3  1.943562  8.43  6.49  5.47

> lnydf.c <- ur.df(lny,type = "drift", lags = 6,selectlags = "BIC")
> cbind(t(lnydf.c@teststat),lnydf.c@cval)

          statistic  1pct  5pct 10pct
      tau2 -1.679225 -3.46 -2.88 -2.57
      phi1  6.068247  6.52  4.63  3.81
```

```
> lnydf.nc <- ur.df(lny,type = "none", lags = 6,selectlags = "BIC")
> cbind(t(lnydf.nc@teststat),lnydf.nc@cval)

           statistic  1pct  5pct 10pct
      tau1  2.854122 -2.58 -1.95 -1.62
```

So, I then take first differences of the GDP series and compute the test statistics from which I can conclude that GDP growth is trend-stationary

```
> gy <- diff(lny)
> gy.ct <- ur.df(gy,type = "trend", lags = 6,selectlags = "BIC")
> cbind(t(gy.ct@teststat),gy.ct@cval)

           statistic  1pct  5pct 10pct
      tau3 -4.762204 -3.99 -3.43 -3.13
      phi2  7.596229  6.22  4.75  4.07
      phi3 11.365183  8.43  6.49  5.47

> gy.c <- ur.df(gy,type = "drift", lags = 6,selectlags = "BIC")
> cbind(t(gy.c@teststat), gy.c@cval)

           statistic  1pct  5pct 10pct
      tau2  -4.56585 -3.46 -2.88 -2.57
      phi1  10.45243  6.52  4.63  3.81

> gy.nc <- ur.df(gy,type = "none", lags = 6,selectlags = "BIC")
> cbind(t(gy.nc@teststat),gy.nc@cval)

           statistic  1pct  5pct 10pct
      tau1 -3.288913 -2.58 -1.95 -1.62
```

So, again, you can probably say that GDP growth is trend-stationary.

STATA Translation:

```
varsoc lny, maxlag(6) exog(time)
dfuller lny, lags(5) trend

      Augmented Dickey-Fuller test for unit root     Number of obs = 100

                           ---------- Interpolated Dickey-Fuller ---------
               Test         1% Critical       5% Critical      10% Critical
             Statistic           Value             Value            Value
      Z(t)   -1.593             -4.040             -3.450       -3.150

dfuller lny, lags(5) drift

      Augmented Dickey-Fuller test for unit root     Number of obs = 100

                           ---------- Interpolated Dickey-Fuller ---------
               Test         1% Critical       5% Critical      10% Critical
             Statistic           Value             Value            Value
      Z(t)   -1.624             -2.367             -1.661       -1.291

dfuller lny, lags(5) noconstant

      Augmented Dickey-Fuller test for unit root     Number of obs = 100

                           ---------- Interpolated Dickey-Fuller ---------
               Test         1% Critical       5% Critical      10% Critical
             Statistic           Value             Value            Value
      Z(t)   2.374              -2.600             -1.950       -1.610
```

Again, I can't reject the hypothesis of non-stationarity. So, I compute the log differences to calculate GDP growth.

```
gen gy = lny-lny[_n-1]
varsoc gy, maxlag(6) exog(time)
dfuller gy, lags(4) trend

       Augmented Dickey-Fuller test for unit root     Number of obs = 100

                            ---------- Interpolated Dickey-Fuller ---------
                  Test         1% Critical       5% Critical      10% Critical
                 Statistic           Value             Value            Value
       Z(t)   -3.891              -4.040              -3.450       -3.150

dfuller gy, lags(4) drift regress

       Augmented Dickey-Fuller test for unit root     Number of obs = 100

                            ---------- Interpolated Dickey-Fuller ---------
                  Test         1% Critical       5% Critical      10% Critical
                 Statistic           Value             Value            Value
       Z(t)   -3.664              -2.367              -1.661       -1.291

dfuller gy, lags(4) noconstant regress

       Augmented Dickey-Fuller test for unit root     Number of obs = 100

                            ---------- Interpolated Dickey-Fuller ---------
                  Test         1% Critical       5% Critical      10% Critical
                 Statistic           Value             Value            Value
       Z(t)   -2.579              -2.600              -1.950       -1.610
```

Results for The Money Interest Rate

The money rate at first appears close to being stationary, however, when I apply the third of the three tests, I wind up with contradictory results, so I'll ultimately have to take first differences to get more convincing results of stationarity

R Code:

```
> lnmr <- ts(johansen$lnmr, start=c(1958,2),end=c(1984,3),frequency=4)
> lnmrdf.ct <- ur.df(lnmr,type = "trend", lags = 6,selectlags = "BIC")
> cbind(t(lnmrdf.ct@teststat),lnmrdf.ct@cval)

           statistic  1pct  5pct 10pct
      tau3 -4.958365 -3.99 -3.43 -3.13
      phi2  8.233197  6.22  4.75  4.07
      phi3 12.322775  8.43  6.49  5.47

> lnmrdf.c <- ur.df(lnmr,type = "drift", lags = 6,selectlags = "BIC")
> cbind(t(lnmrdf.c@teststat),lnmrdf.c@cval)

           statistic  1pct  5pct 10pct
      tau2 -4.932651 -3.46 -2.88 -2.57
      phi1 12.192698  6.52  4.63  3.81

> lnmrdf.nc <- ur.df(lnmr,type = "none", lags = 6,selectlags = "BIC")
> cbind(t(lnmrdf.nc@teststat),lnmrdf.nc@cval)

           statistic  1pct  5pct 10pct
      tau1 -1.277184 -2.58 -1.95 -1.62
```

Since this last test contradicts the results from the previous two attempts, take first differences and then rerun the tests.

```
> gr <- diff(lnmr)
> gr.ct <- ur.df(gr,type = "trend", lags = 6,selectlags = "BIC")
> cbind(t(gr.ct@teststat),gr.ct@cval)
```

```
          statistic  1pct  5pct 10pct
       tau3 -8.198998 -3.99 -3.43 -3.13
       phi2 22.408311  6.22  4.75  4.07
       phi3 33.612391  8.43  6.49  5.47

> gr.c <- ur.df(gr,type = "drift", lags = 6,selectlags = "BIC")
> cbind(t(gr.c@teststat),gr.c@cval)

          statistic  1pct  5pct 10pct
       tau2 -8.232412 -3.46 -2.88 -2.57
       phi1 33.886380  6.52  4.63  3.81

> gr.nc <- ur.df(gr,type = "none", lags = 6,selectlags = "BIC")
> cbind(t(gr.nc@teststat),gr.nc@cval)

          statistic  1pct  5pct 10pct
       tau1 -8.265611 -2.58 -1.95 -1.62
```

This is convincing. So changes in the money interest rate are stationary.

STATA Translation:

```
varsoc lnmr, maxlag(6) exog(time)
dfuller lnmr, lags(1) trend

      Augmented Dickey-Fuller test for unit root     Number of obs = 104

                           ---------- Interpolated Dickey-Fuller ---------
                 Test         1% Critical       5% Critical      10% Critical
               Statistic           Value             Value            Value
      Z(t)   -4.732              -4.039              -3.449       -3.149

dfuller lnmr, lags(1) drift

      Augmented Dickey-Fuller test for unit root     Number of obs = 104

                           ---------- Interpolated Dickey-Fuller ---------
                 Test         1% Critical       5% Critical      10% Critical
               Statistic           Value             Value            Value
      Z(t)   -4.754              -2.364              -1.660       -1.290

dfuller lnmr, lags(1) noconstant

      Augmented Dickey-Fuller test for unit root     Number of obs = 104

                           ---------- Interpolated Dickey-Fuller ---------
                 Test         1% Critical       5% Critical      10% Critical
               Statistic           Value             Value            Value
      Z(t)   -1.354              -2.599              -1.950       -1.610
```

STATA too gives a third test result that contradicts the results from the previous two tests. So then take first differences and rerun the tests.

```
gen gr = lnmr-lnmr[_n-1]
varsoc gr, maxlag(6) exog(time)
dfuller gr, lags(5) trend

      Augmented Dickey-Fuller test for unit root     Number of obs = 99

                           ---------- Interpolated Dickey-Fuller ---------
                 Test         1% Critical       5% Critical      10% Critical
               Statistic           Value             Value            Value
      Z(t)   -6.477              -4.042              -3.451       -3.151
```

```
dfuller gr, lags(5) drift

      Augmented Dickey-Fuller test for unit root     Number of obs = 99

                        ---------- Interpolated Dickey-Fuller ---------
           Test         1% Critical       5% Critical      10% Critical
         Statistic           Value             Value            Value
      Z(t)   -6.501             -2.368             -1.662       -1.291

dfuller gr, lags(5) noconstant

      Augmented Dickey-Fuller test for unit root     Number of obs = 99

                        ---------- Interpolated Dickey-Fuller ---------
           Test         1% Critical       5% Critical      10% Critical
         Statistic           Value             Value            Value
      Z(t)   -6.520             -2.600             -1.950       -1.610
```

Results for The Rate of Inflation

As with the money interest rate, the results for inflation are a bit troubling. So, I'll run the three tests first with the constant and trend, then with just the constant and no trend, and finally without a constant or trend. Then I'll take the first difference of the inflation, which actually gives you the rate of acceleration or deceleration in the price level, and that you can reject it being non-stationary.

R Code:

```
> difp <- ts(johansen$difp, start=c(1958,2),end=c(1984,3),frequency=4)
> difpdf.ct <- ur.df(difp,type = "trend", lags = 6,selectlags = "BIC")
> cbind(t(difpdf.ct@teststat),difpdf.ct@cval)

           statistic  1pct  5pct 10pct
      tau3 -4.085187 -3.99 -3.43 -3.13
      phi2  5.574917  6.22  4.75  4.07
      phi3  8.362128  8.43  6.49  5.47

> difpdf.c <- ur.df(difp,type = "drift", lags = 6,selectlags = "BIC")
> cbind(t(difpdf.c@teststat),difpdf.c@cval)

           statistic  1pct  5pct 10pct
      tau2 -2.434529 -3.46 -2.88 -2.57
      phi1  2.971361  6.52  4.63  3.81

> difpdf.nc <- ur.df(difp,type = "none", lags = 6,selectlags = "BIC")
> cbind(t(difpdf.nc@teststat),difpdf.nc@cval)

           statistic  1pct  5pct 10pct
      tau1 -1.028685 -2.58 -1.95 -1.62

> diffdifp <- diff(difp)
> difpdfdiff.ct <- ur.df(diffdifp,type = "trend", lags = 6,selectlags =
"BIC")
> cbind(t(difpdfdiff.ct@teststat),difpdfdiff.ct@cval)

           statistic  1pct  5pct 10pct
      tau3 -10.17202 -3.99 -3.43 -3.13
      phi2  34.49351  6.22  4.75  4.07
      phi3  51.74023  8.43  6.49  5.47

> difpdfdiff.c <- ur.df(diffdifp,type = "drift", lags = 6,selectlags =
"BIC")
> cbind(t(difpdfdiff.c@teststat),difpdfdiff.c@cval)
```

```
          statistic  1pct  5pct 10pct
       tau2 -10.21271 -3.46 -2.88 -2.57
       phi1  52.14981  6.52  4.63  3.81

> difpdfdiff.nc <- ur.df(diffdifp,type = "none", lags = 6,selectlags =
"BIC")
> cbind(t(difpdfdiff.nc@teststat),difpdfdiff.nc@cval)

          statistic  1pct  5pct 10pct
       tau1 -10.26618 -2.58 -1.95 -1.62
```

STATA Translation:

```
varsoc difp, maxlag(6)
dfuller difp, lags(4) trend regress

      Augmented Dickey-Fuller test for unit root     Number of obs = 101

                  ---------- Interpolated Dickey-Fuller ---------
             Test      1% Critical       5% Critical      10% Critical
           Statistic       Value            Value             Value
      Z(t)   -2.516       -4.040           -3.450             -3.150

dfuller difp, lags(4) drift regress

      Augmented Dickey-Fuller test for unit root     Number of obs = 101

                  ---------- Interpolated Dickey-Fuller ---------
             Test      1% Critical       5% Critical      10% Critical
           Statistic       Value            Value             Value
      Z(t)   -2.509       -2.366           -1.661             -1.291

dfuller difp, lags(4) noconstant regress

      Augmented Dickey-Fuller test for unit root     Number of obs = 101

                  ---------- Interpolated Dickey-Fuller ---------
             Test      1% Critical       5% Critical      10% Critical
           Statistic       Value            Value             Value
      Z(t)   -0.878       -2.600           -1.950             -1.610
```

Now, since the last test does not reject the unit root, it's better to take first differences and then to rerun the tests.

```
gen diffdifp = difp-difp[_n-1]
dfuller diffdifp, lags(4) trend

      Augmented Dickey-Fuller test for unit root     Number of obs = 100

                  ---------- Interpolated Dickey-Fuller ---------
             Test      1% Critical       5% Critical      10% Critical
           Statistic       Value            Value             Value
      Z(t)   -5.655       -4.040           -3.450             -3.150

dfuller diffdifp, lags(4) drift

      Augmented Dickey-Fuller test for unit root     Number of obs = 100

                  ---------- Interpolated Dickey-Fuller ---------
             Test      1% Critical       5% Critical      10% Critical
           Statistic       Value            Value             Value
      Z(t)   -5.641       -2.367           -1.661             -1.291
```

```
dfuller diffdifp, lags(4) noconstant

Augmented Dickey-Fuller test for unit root     Number of obs = 100

                     ---------- Interpolated Dickey-Fuller ---------
        Test         1% Critical       5% Critical      10% Critical
      Statistic         Value             Value            Value
Z(t)   -5.667        -2.600            -1.950           -1.610
```

So, the Finnish data series from Johansen and Juselius (1991) are all difference stationary. I can now check to see if there are any co-integrating relationships among them.

Cointegration Analysis

When you look at the formal presentation of cointegration analysis it looks quite complicated, yet if you boil it down to the intuition it is rather simple. Accordingly, I'll spare the formal details, since you can get that from Pfaff (2006) or plenty of other sources, and jump to the intuition. The idea is simply do two or more variables share a common random walk? You can't address this by looking at non-stationary variables, since when you regress one random walk against another the correlations are spurious. Pfaff has the R codes for you to simulate this. Granger and Engle (1987) propose cointegration analysis, in which the relationship between two or more integrated series can be assessed. That is, by first transforming the non-stationary series to be stationary, the relationship can then be assessed. Typically, taking differences (first or second, or more) of the non-stationary series makes them stationary. If a series is non-stationary, and the first (or nth) difference is stationary, it is called integrated of order one (or *n*), denoted *I(1)* (or *I(n)*). If a series is stationary it is denoted *I(0)*. After determining the stationarity of each variable then an analysis of their interaction can be performed using the aforementioned cointegration analysis. This means bringing all the variables on the left-hand side and normalizing the coefficients by dividing by the coefficient of the original left-hand side variable and testing whether there is co-movement among the variables using the Johansen test. As Pfaff (2006) discusses, you can think about cointegration in a transitory or long run sense. In the transitory case, you have one lag of the vector of variables, as in

$$13) \qquad \Delta x_t = \mu + \underbrace{\Gamma_1 \Delta x_{t-1} + \cdots + \Gamma_k \Delta x_{t-k}}_{\substack{transitory\ cumulative \\ innovations}} + \underbrace{\Pi x_{t-1}}_{\substack{level \\ effect}} + \varepsilon_t$$

while in the long-run case you have the level effect is lagged back *k* periods so that the innovations lead up to the current innovation

$$14) \qquad \Delta x_t = \mu + \underbrace{\Gamma_1 \Delta x_{t-1} + \cdots + \Gamma_k \Delta x_{t-k}}_{long-run\ innovations} + \underbrace{\Pi x_{t-k}}_{\substack{level \\ effect}} + \varepsilon_t$$

There are a number of really intuitive applications of cointegration analysis, which is why I have covered it here. Again, the common theme is simply this: is there a unit root that is common to all of the variables in the system being considered? You will probably have to go further than this in your own work, however, I think the most important issue is whether a collection of variables is co-integrated because it has real world implications. I'll start out with just testing for cointegration, and later I'll show

you the process from E-Views, to give you a feel for the practicality of the technique. I'll begin with the cointegration rank test in R and STATA for the four Finnish data series since they are each I(1).

Useful Application #1: Estimating a Money Demand Equation

You might think of this as an exercise in vanity, because really money demand estimation is much harder (probably impossible) than you think for a central bank given the millions of contracts and transactions that take place each day. But, given that we have central banking, you are almost compelled to do this to discover the match with the money supply. For this last application, I'll just do the rank test. If the trace test statistic is greater than the 1% critical value you know the series are co-integrated.

R Code:

```
> finnish <- data.frame(gm,gy,gr,diffdifp)
> H1 <- ca.jo(finnish,type='trace',K=4)
> cbind(H1@cval,cbind(H1@teststat))

           10pct  5pct  1pct
r <= 3 |  6.50  8.18 11.65  15.68204
r <= 2 | 15.66 17.95 23.52  56.39332
r <= 1 | 28.71 31.52 37.22 113.82506
r = 0  | 45.23 48.28 55.43 191.61472
```

STATA Translation:

```
varsoc gm gy gr diffdifp, maxlag(6) exog(time)

      Selection order criteria
      Sample:  1960q1     1984q3 Number of obs = 99

     lag     LL      LR      df    p      FPE       AIC      HQIC      SBIC
     ----+--------------------------------------------------------------------
     0   739.014                       4.5e-12   -14.768  -14.6831  -14.5582
     1   795.184  112.34   16  0.000  2.0e-12  -15.5795  -15.3249  -14.9504*
     2   810.523  30.677   16  0.015  2.0e-12  -15.5661  -15.1419  -14.5176
     3   859.029  97.012   16  0.000  1.1e-12  -16.2228  -15.6289  -14.7549
     4   896.254   74.45*  16  0.000  7.0e-13* -16.6516*  -15.888* -14.7642
     5   909.065  25.623   16  0.060  7.6e-13  -16.5872  -15.6538  -14.2804
     6   919.527  20.924   16  0.181  8.6e-13  -16.4753  -15.3723  -13.7491

vecrank gm gy gr diffdifp, lags(4) levela

                  Johansen tests for cointegration
      Trend: constant                                  Number of obs = 101
      Sample:   1959q3    1984q3                              Lags = 4
Maximum                                  trace     5% critical   1% critical
rank   parms  LL           eigenvalue statistic       value         value
0      52     816.93904                 191.6147      47.21         54.46
1      59     855.83389      0.53708    113.8250      29.68         35.65
2      64     884.54977      0.43370     56.3933      15.41         20.04
3      67     904.9054       0.33174     15.6820       3.76          6.65
4      68     912.74641      0.14381
```

So, you see that STATA and R are producing the same test statistic here, but the critical values they each report are slightly different.

Useful Application #2: Searching for Segmented Regions in Guinea, West Africa

Having watched my colleague at the World Bank do co-integration analysis for about fifteen regions in Guinea, I then repeated this exercise for three regional rice prices against the world rice price, to be used as a teaching device. What my colleague was looking for was to figure out the regions that were facing the most difficulties in integrating with the rest of the country. By adding in the world rice price we can see if the regional markets are integrated with the world. Put another way, if the market prices of the commodity in one region are cointegrated with the market prices in other regions, then the markets are integrated, or else they are segmented. This exercise can be used to provide some part of the answer to the question: are regions in Guinea, West Africa integrated with the capital, and also the world rice market? An analogy for Australia would be whether the Outback, and Melbourne-Sydney-Brisbane-Perth-Adelaide are integrated with the world market. Your first thought might be that the Outback is segmented, while the cities are well integrated, but that's something you can test. Regional GDP would be an obvious choice of variables, but, especially in a country like Guinea, you may have difficulty getting sufficient GDP data to do the analysis. For about 20 regions, we got about 70 monthly observations of rice prices (70 is a bare minimum at best; in practice you should only think about doing this if you can get more data). First my colleague tested for stationarity, and found that only 15 out of almost 20 regions so were difference-stationary (that is I(1)), so we excluded the series that were non-stationary in differences. Here, I will show you the results for the stationarity tests and then the cointegration analysis to test whether or not Conakry, Boffa, and Boke are economically tied to the world rice market (see the Atlantic coastline in the map above, taken from MapZones.com).

Figure 13. A Map of Guinea, West Africa to Situate the Cointegrated Regions

To test this proposition more formally, cointegration analysis can be used to determine which markets are in the "integrated space". Since these are price series, and since price series are generally non-stationary, a regression of one price on others is spurious (the regressions will look much better than they really are). To do this in practice, the first step is therefore to test the stationarity of each of the four rice price series (Thai 35% broken rice, which is the proxy for the world rice price, and the 35% broken bulk rice price series for each of the three Guinean cities). Augmented Dickey Fuller tests can be estimated as above. The ADF-tests are first run with a constant, trend, and three lags, as shown in equation 12) above. When I did this, I was unaware of the general-to-specific approach that I followed earlier, and which is probably better than what you see here where I dropped everything that is statistically insignificant at the same time; that's because your risk omitting relevant variables, which is more problematic than including irrelevant variables, in this case one or more lags. As seen earlier, if the value of the ADF test is greater in absolute value than the 1% critical value, then the hypothesis of non-stationarity is rejected, and the integrated series can be included in the co-integration analysis. By taking first differences, stationarity is restored in the sense that the ADF tests values are now lower than the theoretical values at the five percent level.

World Price Stationarity Tests

ADF test of World with constant, trend, and three lags

ADF Test Statistic	**-3.418352**	1% Critical Value*	-4.0969
		5% Critical Value	-3.4759
		10% Critical Value	-3.1651

*MacKinnon critical values for rejection of hypothesis of a unit root.

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(LWORLD)
Method: Least Squares
Date: 06/27/03 Time: 16:05
Sample(adjusted): 1995:05 2000:12
Included observations: 68 after adjusting endpoints

Variable	Coefficient	Std. Error	t-Statistic	Prob.
LWORLD(-1)	-0.316913	0.092709	-3.418352	0.0011
D(LWORLD(-1))	0.302266	0.124300	2.431749	0.0179
D(LWORLD(-2))	-0.007906	0.124255	-0.063624	**0.9495**
D(LWORLD(-3))	0.081051	0.124396	0.651559	**0.5171**
C	1.794744	0.522638	3.434010	0.0011
@TREND(1995:01)	4.04E-05	0.000360	0.112220	0.9110

R-squared	0.204933	Mean dependent var	0.002360
Adjusted R-squared	0.140815	S.D. dependent var	0.061709
S.E. of regression	0.057200	Akaike info criterion	-2.800442
Sum squared resid	0.202851	Schwarz criterion	-2.604603
Log likelihood	101.2150	F-statistic	3.196177
Durbin-Watson stat	1.991636	Prob(F-statistic)	0.012455

These results suggest that the trend should not be there, and three lags are probably too much (the p-values on @trend and lag two are far from 0.05). So the trend and two lags are dropped in the table below. The Akaike drops from -2.800 in the table above to -2.899 in the table below, suggesting that that is a better specification. Yes, the first differences are stationary, even though the log of the world is not.

ADF test of World with constant, no trend and one lag

ADF Test Statistic	-3.729186	1% Critical Value*	-3.5253
		5% Critical Value	-2.9029
		10% Critical Value	-2.5886

*MacKinnon critical values for rejection of hypothesis of a unit root.

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(LWORLD)
Method: Least Squares
Date: 06/27/03 Time: 16:06
Sample(adjusted): 1995:03 2000:12
Included observations: 70 after adjusting endpoints

Variable	Coefficient	Std. Error	t-Statistic	Prob.
LWORLD(-1)	-0.277299	0.074359	-3.729186	0.0004
D(LWORLD(-1))	0.270997	0.115346	2.349435	0.0218
C	1.571090	0.420773	3.733821	0.0004

R-squared	0.188554	Mean dependent var	0.002449
Adjusted R-squared	0.164332	S.D. dependent var	0.060813
S.E. of regression	0.055593	Akaike info criterion	-2.899625
Sum squared resid	0.207065	Schwarz criterion	-2.803261
Log likelihood	104.4869	F-statistic	7.784317
Durbin-Watson stat	2.000338	Prob(F-statistic)	0.000912

Conakry Price Stationarity Tests

As before, the trend should not be there (the p-values on @trend is from 0.05), therefore, it is dropped. Notice that the Akaike drops from -2.18 in the table above to -2.23 after dropping the trend, suggesting that that is a better specification. The first differences are stationary, even though the log levels are not.

ADF test of Conakry with constant, trend, and three lags

ADF Test Statistic	-4.237653	1% Critical Value*	-4.0969
		5% Critical Value	-3.4759
		10% Critical Value	-3.1651

*MacKinnon critical values for rejection of hypothesis of a unit root.

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(LCONAK)
Method: Least Squares
Date: 06/27/03 Time: 16:12
Sample(adjusted): 1995:05 2000:12
Included observations: 68 after adjusting endpoints

Variable	Coefficient	Std. Error	t-Statistic	Prob.
LCONAK(-1)	-0.385653	0.091006	-4.237653	0.0001
D(LCONAK(-1))	0.274384	0.115924	2.366927	0.0211
D(LCONAK(-2))	0.265660	0.120653	2.201850	0.0314
D(LCONAK(-3))	0.152819	0.123803	1.234370	0.2217
C	2.558320	0.603203	4.241223	0.0001
@TREND(1995:01)	-0.000146	0.000482	-0.301724	0.7639

R-squared	0.248092	Mean dependent var	0.000427
Adjusted R-squared	0.187454	S.D. dependent var	0.086479
S.E. of regression	0.077954	Akaike info criterion	-2.181307
Sum squared resid	0.376760	Schwarz criterion	-1.985468
Log likelihood	80.16444	F-statistic	4.091381
Durbin-Watson stat	2.051979	Prob(F-statistic)	0.002833

ADF test of Conakry with constant, no trend, and two lags

ADF Test Statistic	-4.215160	1% Critical Value*	-3.5267
		5% Critical Value	-2.9035
		10% Critical Value	-2.5889

*MacKinnon critical values for rejection of hypothesis of a unit root.

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(LCONAK)
Method: Least Squares
Date: 06/27/03 Time: 16:11
Sample(adjusted): 1995:04 2000:12
Included observations: 69 after adjusting endpoints

Variable	Coefficient	Std. Error	t-Statistic	Prob.
LCONAK(-1)	-0.336154	0.079749	-4.215160	0.0001
D(LCONAK(-1))	0.266893	0.114164	2.337790	0.0225
D(LCONAK(-2))	0.258967	0.118185	2.191210	0.0320
C	2.225703	0.527911	4.216057	0.0001

R-squared	0.234928	Mean dependent var	0.001389
Adjusted R-squared	0.199617	S.D. dependent var	0.086212
S.E. of regression	0.077129	Akaike info criterion	-2.230454
Sum squared resid	0.386677	Schwarz criterion	-2.100940
Log likelihood	80.95065	F-statistic	6.653093
Durbin-Watson stat	2.080851	Prob(F-statistic)	0.000548

Boffa Price Stationarity Tests

The tests below suggests that the trend and last lag should not be there (the p-values on @trend and lag are far from 0.05). Therefore, two tables below the trend is dropped. Notice that the Akaike drops from –1.97 in the table above to -2.03 after dropping the trend and one lag, suggesting that that is a better specification. Yes, the first differences are stationary, even though the log of the Boffa's rice price is not.

ADF test of Boffa with constant, trend, and three lags

ADF Test Statistic	-4.262570	1% Critical Value*	-4.0969
		5% Critical Value	-3.4759
		10% Critical Value	-3.1651

*MacKinnon critical values for rejection of hypothesis of a unit root.

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(LBOFFA)
Method: Least Squares
Date: 06/27/03 Time: 16:13
Sample(adjusted): 1995:05 2000:12
Included observations: 68 after adjusting endpoints

Variable	Coefficient	Std. Error	t-Statistic	Prob.
LBOFFA(-1)	-0.381060	0.089397	-4.262570	0.0001
D(LBOFFA(-1))	0.365965	0.115016	3.181859	0.0023
D(LBOFFA(-2))	0.374146	0.123998	3.017353	0.0037
D(LBOFFA(-3))	0.071350	0.135175	0.527831	0.5995
C	2.450476	0.572814	4.277963	0.0001
@TREND(1995:01)	0.000646	0.000581	1.112700	0.2701

R-squared	0.328887	Mean dependent var	-0.001647
Adjusted R-squared	0.274765	S.D. dependent var	0.101517
S.E. of regression	0.086453	Akaike info criterion	-1.974337
Sum squared resid	0.463394	Schwarz criterion	-1.778499
Log likelihood	73.12747	F-statistic	6.076769
Durbin-Watson stat	1.971569	Prob(F-statistic)	0.000122

ADF test of Boffa with constant, no trend, and two lags

ADF Test Statistic	-4.619981	1% Critical Value*	-3.5267
		5% Critical Value	-2.9035
		10% Critical Value	-2.5889

*MacKinnon critical values for rejection of hypothesis of a unit root.

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(LBOFFA)
Method: Least Squares
Date: 06/27/03 Time: 16:13
Sample(adjusted): 1995:04 2000:12
Included observations: 69 after adjusting endpoints

Variable	Coefficient	Std. Error	t-Statistic	Prob.
LBOFFA(-1)	-0.325520	0.070459	-4.619981	0.0000
D(LBOFFA(-1))	0.349267	0.112771	3.097128	0.0029
D(LBOFFA(-2))	0.348722	0.119442	2.919590	0.0048
C	2.113745	0.458099	4.614162	0.0000

R-squared	0.316915	Mean dependent var	-0.000784
Adjusted R-squared	0.285388	S.D. dependent var	0.101023
S.E. of regression	0.085400	Akaike info criterion	-2.026728
Sum squared resid	0.474051	Schwarz criterion	-1.897214
Log likelihood	73.92211	F-statistic	10.05216
Durbin-Watson stat	2.002127	Prob(F-statistic)	0.000016

Boke Price Stationarity Tests

The tests below suggests that the trend and at least one lag should not be there (the p-values on @trend and second lag are far from 0.05). In the tables below, the trend and two lags are dropped. The Akaike rises from –2.226 in the table above to -2.223 after dropping the trend and two lags, however the ADF drops below the 1% critical value with the smaller model suggesting that that is a better specification. Yes, the first differences are stationary, even though the log of Boke's rice price is not.

ADF test of Boke with constant, trend, and three lags

ADF Test Statistic	-3.669111	1% Critical Value*	-4.0969
		5% Critical Value	-3.4759
		10% Critical Value	-3.1651

*MacKinnon critical values for rejection of hypothesis of a unit root.

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(LBOKE)
Method: Least Squares
Date: 06/27/03 Time: 16:14
Sample(adjusted): 1995:05 2000:12
Included observations: 68 after adjusting endpoints

Variable	Coefficient	Std. Error	t-Statistic	Prob.
LBOKE(-1)	-0.301642	0.082211	-3.669111	0.0005
D(LBOKE(-1))	0.326907	0.116393	2.808655	0.0066
D(LBOKE(-2))	-0.005307	0.117462	-0.045185	0.9641
D(LBOKE(-3))	0.254979	0.117183	2.175910	0.0334
C	1.923184	0.524588	3.666085	0.0005
@TREND(1995:01)	0.000471	0.000492	0.956142	0.3427

R-squared	0.222652	Mean dependent var	-0.000406
Adjusted R-squared	0.159962	S.D. dependent var	0.083190
S.E. of regression	0.076246	Akaike info criterion	-2.225596
Sum squared resid	0.360438	Schwarz criterion	-2.029757
Log likelihood	81.67025	F-statistic	3.551668
Durbin-Watson stat	1.973470	Prob(F-statistic)	0.006895

ADF test of Conakry with constant, no trend, and one lag

ADF Test Statistic	-3.828176	1% Critical Value*	-3.5253
		5% Critical Value	-2.9029
		10% Critical Value	-2.5886

*MacKinnon critical values for rejection of hypothesis of a unit root.

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(LBOKE)
Method: Least Squares
Date: 06/27/03 Time: 16:16
Sample(adjusted): 1995:03 2000:12
Included observations: 70 after adjusting endpoints

Variable	Coefficient	Std. Error	t-Statistic	Prob.
LBOKE(-1)	-0.258862	0.067620	-3.828176	0.0003
D(LBOKE(-1))	0.298614	0.112687	2.649952	0.0100
C	1.668596	0.435167	3.834379	0.0003

R-squared	0.208823	Mean dependent var	0.003954
Adjusted R-squared	0.185206	S.D. dependent var	0.086391
S.E. of regression	0.077982	Akaike info criterion	-2.222766
Sum squared resid	0.407440	Schwarz criterion	-2.126402
Log likelihood	80.79682	F-statistic	8.841983
Durbin-Watson stat	1.991415	Prob(F-statistic)	0.000391

Johansen's cointegration test is run to determine how many markets are connected. If the number of cointegrating equations is below the total number of markets minus one, not all markets are contained in the integrated space. The likelihood ratio must be greater than the critical value here. The top portion of the E-Views output is reproduced below; notice, all markets are integrated, although the test statistic is just below the 1% critical value. It should be mentioned here that in addition to constant and trend, you see two exogenous series: **recolte**, which is a dummary variable for the months of the harvest, and **soudure**, which is a dummy variable for the months during which the seeds are planted.

Date: 06/27/03 Time: 17:42
Sample: 1995:01 2000:12
Included observations: 69
Test assumption: Quadratic deterministic trend in the data
Series: LWORLD LCONAK LBOKE LBOFFA
Exogenous series: RECOLTE SOUDURE
Warning: Critical values were derived assuming no exogenous series
Lags interval: 1 to 2

Eigenvalue	Likelihood Ratio	5 Percent Critical Value	1 Percent Critical Value	Hypothesized No. of CE(s)
0.389509	73.00227	54.64	61.24	None **
0.228402	38.95129	34.55	40.49	At most 1 *
0.192250	21.06013	18.17	23.46	At most 2 *
0.087637	6.328472	3.74	6.40	At most 3 *

*(**) denotes rejection of the hypothesis at 5%(1%) significance level
L.R. test indicates 4 cointegrating equation(s) at 5% significance level

Unnormalized Cointegrating Coefficients:

LWORLD	LCONAK	LBOKE	LBOFFA
-0.664074	-2.083896	0.253087	1.995538
1.873414	-1.907785	-0.356755	0.910542
-0.912541	-1.576989	2.837943	-0.996026
-0.212643	-0.494608	-0.319819	-0.477971

Note: Brooks (2002) notes that E-VIEWS 3.1 and 4 generate different Johansen results Also, software packages can usually handle at most about ten markets (vectors).

Here is the other cointegration test result from E-Views. By typing ***coint lworld lconak lboke lboffa*** into the prompt and then including the exogenous variables (under the Vector Error Correction (VEC) option) *soudure* and *recolte*, two seasonal dummy variables representing planting and harvesting respectively, yields.

Date: 06/27/03 Time: 16:22
Sample(adjusted): 1995:04 2000:12
Included observations: 69 after adjusting endpoints
Standard errors & t-statistics in parentheses

Cointegrating Eq:	CointEq1
LWORLD(-1)	1.000000
LCONAK(-1)	3.138048 (1.79161) (1.75153)
LBOKE(-1)	-0.381113 (0.57814) (-0.65920)
LBOFFA(-1)	-3.004992 (1.72836) (-1.73863)
@TREND(95:01)	0.008333
C	-4.749839

Error Correction:	D(LWORLD)	D(LCONAK)	D(LBOKE)	D(LBOFFA)
CointEq1	-0.061992 (0.03983) (-1.55649)	-0.119125 (0.04511) (-2.64102)	-0.036268 (0.05023) (-0.72200)	0.130616 (0.05124) (2.54899)
D(LWORLD(-1))	0.140507 (0.13351) (1.05243)	0.248597 (0.15120) (1.64417)	0.066360 (0.16838) (0.39410)	0.111489 (0.17177) (0.64906)
D(LWORLD(-2))	-0.086166 (0.14216) (-0.60611)	0.050046 (0.16100) (0.31085)	0.133460 (0.17930) (0.74434)	-0.439715 (0.18290) (-2.40408)
D(LCONAK(-1))	0.098395 (0.15455) (0.63666)	-0.126766 (0.17503) (-0.72426)	-0.109514 (0.19492) (-0.56184)	-0.450909 (0.19884) (-2.26771)
D(LCONAK(-2))	0.155684 (0.13428) (1.15943)	0.174093 (0.15207) (1.14482)	0.131779 (0.16935) (0.77812)	-0.152096 (0.17276) (-0.88040)
D(LBOKE(-1))	0.083206 (0.13000) (0.64003)	-0.037126 (0.14723) (-0.25216)	0.110085 (0.16397) (0.67139)	0.075005 (0.16726) (0.44843)
D(LBOKE(-2))	0.090479 (0.12202) (0.74152)	-0.152270 (0.13819) (-1.10190)	-0.477459 (0.15390) (-3.10248)	-0.007649 (0.15699) (-0.04873)
D(LBOFFA(-1))	-0.285572 (0.13687) (-2.08645)	0.165945 (0.15501) (1.07056)	-0.084075 (0.17263) (-0.48704)	0.156309 (0.17610) (0.88764)
D(LBOFFA(-2))	-0.265769 (0.13939) (-1.90669)	-0.089728 (0.15786) (-0.56840)	0.303849 (0.17580) (1.72836)	0.204131 (0.17934) (1.13826)
C	0.007068 (0.01872) (0.37759)	0.043643 (0.02120) (2.05863)	0.055259 (0.02361) (2.34050)	0.065358 (0.02408) (2.71370)
@TREND(95:01)	-0.000142 (0.00037)	-2.08E-05 (0.00041)	-0.000172 (0.00046)	-0.000219 (0.00047)

	(-0.38883)	(-0.05023)	(-0.37274)	(-0.46423)
SOUDURE	0.005913	-0.049656	-0.063267	-0.001422
	(0.02239)	(0.02536)	(0.02824)	(0.02881)
	(0.26411)	(-1.95833)	(-2.24045)	(-0.04935)
RECOLTE	-0.002221	-0.086254	-0.089821	-0.171464
	(0.02373)	(0.02687)	(0.02993)	(0.03053)
	(-0.09357)	(-3.20947)	(-3.00109)	(-5.61608)
R-squared	0.210502	0.488821	0.366980	0.519538
Adj. R-squared	0.041324	0.379282	0.231333	0.416582
Sum sq. resids	0.201431	0.258356	0.320422	0.333433
S.E. equation	0.059975	0.067923	0.075643	0.077163
F-statistic	1.244260	4.462551	2.705403	5.046209
Log likelihood	103.4496	94.86278	87.43490	86.06170
Akaike AIC	-2.621727	-2.372834	-2.157533	-2.117730
Schwarz SC	-2.200809	-1.951916	-1.736615	-1.696812
Mean dependent	0.002354	0.001389	0.002604	-0.000784
S.D. dependent	0.061254	0.086212	0.086278	0.101023
Determinant Residual Covariance		7.55E-11		
Log Likelihood		412.4670		
Akaike Information Criteria		-10.33238		
Schwarz Criteria		-8.519189		

One Step Cointegration Analysis

Although all four markets are cointegrated according to Johansen's test, the one step cointegration analysis, that would be used if not all were found to be in the integrated space, is used here. Beginning with the world price, the first natural test would be to see of the biggest market in Conakry is in the same market space as the world, which would suggest whether or not people in the capital of Guinea are subjected to world rice price shocks. According to the table below, the likelihood ratio is above the 1% threshold. Therefore, Conakry responds to world prices.

Step 1: Sequential Cointegration Analysis for World and Conakry

Date: 06/27/03 Time: 16:27
Sample: 1995:01 2000:12
Included observations: 69
Test assumption: Quadratic deterministic trend in the data
Series: LWORLD LCONAK
Exogenous series: RECOLTE SOUDURE
Warning: Critical values were derived assuming no exogenous series
Lags interval: 1 to 2

Eigenvalue	Likelihood Ratio	5 Percent Critical Value	1 Percent Critical Value	Hypothesized No. of CE(s)
0.167020	19.62593	18.17	23.46	None *
0.096689	7.016468	3.74	6.40	At most 1 **

*(**) denotes rejection of the hypothesis at 5%(1%) significance level
L.R. test indicates 2 cointegrating equation(s) at 5% significance level

Unnormalized Cointegrating Coefficients:

LWORLD	LCONAK
1.722460	-0.462747
-0.556411	1.651630

Normalized Cointegrating Coefficients: 1 Cointegrating Equation(s)

LWORLD	LCONAK	@TREND(95:02)	C
1.000000	-0.268655	-0.000629	-3.858855
	(0.23446)		
Log likelihood	189.8962		

The next step below is to see if Boke is also cointegrated with these two markets to see if it responds to world prices.

Step 2: Sequential Cointegration Analysis for World, Conakry, Boffa

Date: 06/27/03 Time: 18:08
Sample: 1995:01 2000:12
Included observations: 69
Test assumption: Quadratic deterministic trend in the data
Series: LWORLD LCONAK LBOFFA
Exogenous series: RECOLTE SOUDURE
Warning: Critical values were derived assuming no exogenous series
Lags interval: 1 to 2

Eigenvalue	Likelihood Ratio	5 Percent Critical Value	1 Percent Critical Value	Hypothesized No. of CE(s)
0.388402	55.83639	34.55	40.49	None **
0.201960	21.91051	18.17	23.46	At most 1 *
0.087846	6.344303	3.74	6.40	At most 2 *

*(**) denotes rejection of the hypothesis at 5%(1%) significance level
L.R. test indicates 3 cointegrating equation(s) at 5% significance level

Unnormalized Cointegrating Coefficients:

LWORLD	LCONAK	LBOFFA
-0.598392	-1.916755	2.057593
-1.719582	2.026683	-0.804652
-0.295473	-0.657454	-0.593159

Normalized Cointegrating Coefficients: 1 Cointegrating Equation(s)

LWORLD	LCONAK	LBOFFA	@TREND(95:02)	C
1.000000	3.203176	-3.438537	0.008925	-4.837767
	(1.90359)	(1.74384)		
Log likelihood	296.3298			

Normalized Cointegrating Coefficients: 2 Cointegrating Equation(s)

LWORLD	LCONAK	LBOFFA	@TREND(95:02)	C
1.000000	0.000000	-0.582812	0.000930	-1.904916
		(0.16531)		
0.000000	1.000000	-0.891529	0.002496	-0.915607
		(0.08891)		
Log likelihood	304.1129			

By adding in Boke to the other three markets, we see that the market is still cointegrated, although the threshold is lower, but still above the 95% significance level. So at the risk of sounding redundant, all four markets are weakly contained in an integrated market space; they are weakly cointegrated.

Step 3: Sequential Cointegration Analysis for World, Conakry, Boffa, Boke

Date: 06/27/03 Time: 18:10
Sample: 1995:01 2000:12
Included observations: 69
Test assumption: Quadratic deterministic trend in the data
Series: LWORLD LCONAK LBOFFA LBOKE
Exogenous series: RECOLTE SOUDURE
Warning: Critical values were derived assuming no exogenous series
Lags interval: 1 to 2

Eigenvalue	Likelihood Ratio	5 Percent Critical Value	1 Percent Critical Value	Hypothesized No. of CE(s)
0.389509	73.00227	54.64	61.24	None **
0.228402	38.95129	34.55	40.49	At most 1 *
0.192250	21.06013	18.17	23.46	At most 2 *
0.087637	6.328472	3.74	6.40	At most 3 *

*(**) denotes rejection of the hypothesis at 5%(1%) significance level
L.R. test indicates 4 cointegrating equation(s) at 5% significance level

Unnormalized Cointegrating Coefficients:

LWORLD	LCONAK	LBOFFA	LBOKE
-0.664074	-2.083896	1.995538	0.253087
1.873414	-1.907785	0.910542	-0.356755
-0.912541	-1.576989	-0.996026	2.837943
0.212643	0.494608	0.477971	0.319819

Normalized Cointegrating Coefficients: 1 Cointegrating Equation(s)

LWORLD	LCONAK	LBOFFA	LBOKE	@TREND(95:02)	C
1.000000	3.138048	-3.004992	-0.381113	0.008333	-4.749839
	(1.79161)	(1.72836)	(0.57814)		

Log likelihood	412.4670				

Normalized Cointegrating Coefficients: 2 Cointegrating Equation(s)

LWORLD	LCONAK	LBOFFA	LBOKE	@TREND(95:02)	C
1.000000	0.000000	-0.369293	-0.237149	0.000715	-1.758343
		(0.29737)	(0.27106)		
0.000000	1.000000	-0.839917	-0.045877	0.002428	-0.953298
		(0.17418)	(0.15877)		

Log likelihood	421.4126				

Normalized Cointegrating Coefficients: 3 Cointegrating Equation(s)

LWORLD	LCONAK	LBOFFA	LBOKE	@TREND(95:02)	C
1.000000	0.000000	0.000000	-0.591383	0.000263	-1.862092
			(0.12975)		
0.000000	1.000000	0.000000	-0.851543	0.001400	-1.189263
			(0.11711)		
0.000000	0.000000	1.000000	-0.959221	-0.001223	-0.280938
			(0.14189)		

Log likelihood	428.7784				

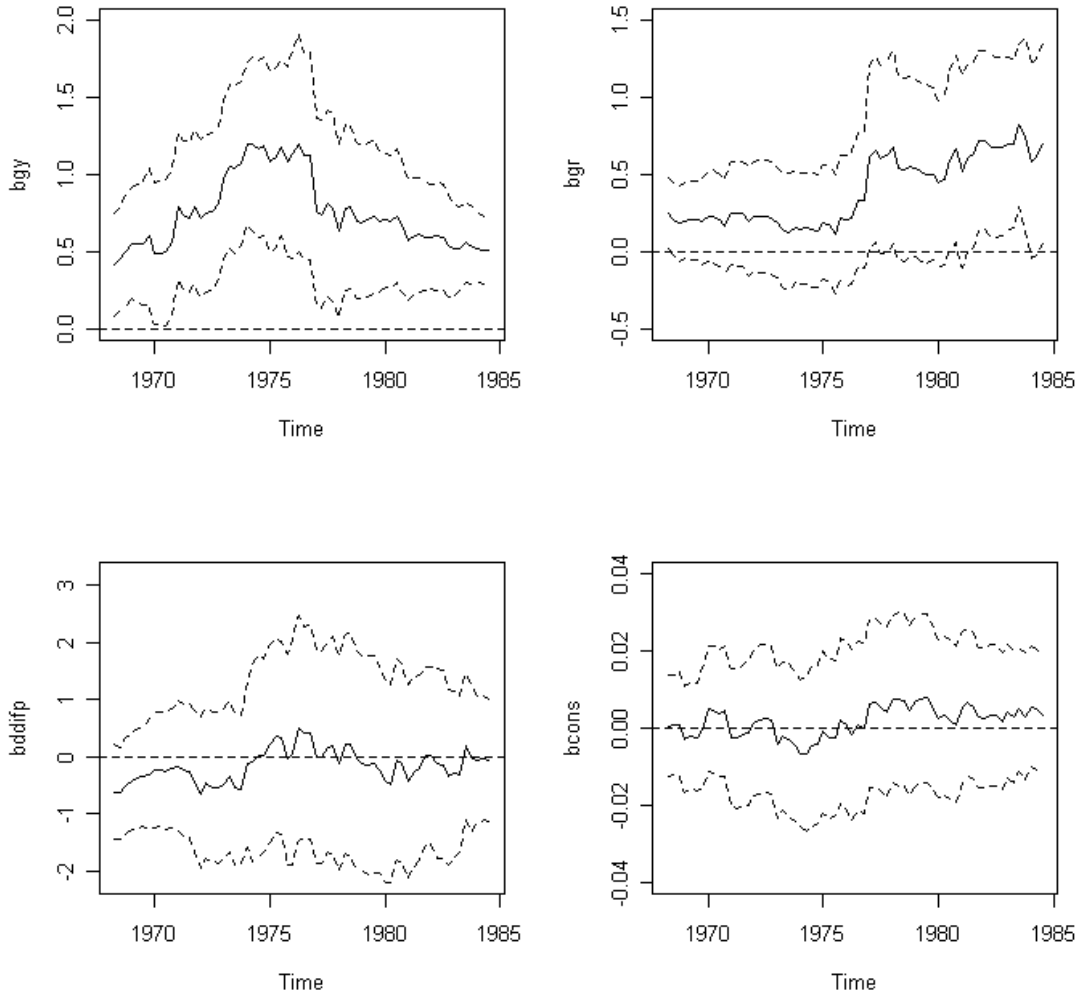
Rolling Regressions

Now I can illustrate what I meant earlier about exploring a simple model over time. Using the same four variables that I used in the cointegration analysis, I'll estimate a rolling money demand equation. In this case, I'll estimate a simple linear regression over time with ten years of quarterly observations, but after each estimation I'll drop the earliest observation and add the latest. I chose ten years rather arbitrarily, as I don't want the sample to be too small, and figured 40 observations per estimation would be small, but not too small. This is probably better done with higher frequency data, but still it can be informative to see the effect of news, or changes in the economy. I'll break up the codes in four sections to: 1) create the data, 2) estimate the rolling regressions, 3) construct the objects to be plotted, and then 4) plot them. I'll then show you the same thing in STATA.

```
# Create a new dataset so that all variables have the same starting date
sample <- length(johansen$date)
finnish <- data.frame(date=johansen$date[2:sample],gm,gy,gr,diffdifp)

# Create objects in R's memory to store the forthcoming regression output
window <- 40
subsample <- sample-window
bgy <- numeric(subsample)
bgy.ci <- numeric(subsample)
bgr <- numeric(subsample)
bgr.ci <- numeric(subsample)
bddifp <- numeric(subsample)
bddifp.ci <- numeric(subsample)
bcons <- numeric(subsample)
bcons.ci <- numeric(subsample)

# Estimate rolling regressions
for (i in 1:subsample) {
       model <- lm(gm~gy+gr+diffdifp,data=finnish,subset=i:(i+window-1))
       modelsum <- summary.lm(model)
       coefs <- coefficients(model)
       bgy[i] <- coefs[2]
       bgy.ci[i] <- 1.96*modelsum$coefficients[2,2]
       bgr[i] <- coefs[3]
       bgr.ci[i] <- 1.96*modelsum$coefficients[3,2]
       bddifp[i] <- coefs[4]
       bddifp.ci[i] <- 1.96*modelsum$coefficients[4,2]
       bcons[i] <- coefs[1]
       bcons.ci[i] <- 1.96*modelsum$coefficients[1,2]
}

# Create time series objects to be plotted shortly from the regression
# coefficients and the coefficient standard errors

bgy <- ts(bgy, start=c(1968,2),end=c(1984,3),frequency=4)
bgr <- ts(bgr, start=c(1968,2),end=c(1984,3),frequency=4)
bddifp <- ts(bddifp, start=c(1968,2),end=c(1984,3),frequency=4)
bcons <- ts(bcons, start=c(1968,2),end=c(1984,3),frequency=4)
bgyplusci <- ts(bgy+bgy.ci, start=c(1968,2),end=c(1984,3),frequency=4)
bgyminusci <- ts(bgy-bgy.ci, start=c(1968,2),end=c(1984,3),frequency=4)
bgrplusci <- ts(bgr+bgr.ci, start=c(1968,2),end=c(1984,3),frequency=4)
bgrminusci <- ts(bgr-bgr.ci, start=c(1968,2),end=c(1984,3),frequency=4)
bddifpplusci<-ts(bddifp+bddifp.ci,start=c(1968,2),end=c(1984,3),frequency=4)
bddifpminusci<-ts(bddifp-ddifp.ci,start=c(1968,2),end=c(1984,3),frequency=4)
bconplusci <- ts(bcons+bcons.ci, start=c(1968,2),end=c(1984,3),frequency=4)
bconminusci <- ts(bcons-bcons.ci, start=c(1968,2),end=c(1984,3),frequency=4)
```

```
# Use the par(mfrow … command to create a multi-panel plot, in this case
# with a 2 row, 2 column layout, and then plot each of the series together
# with the standard errors

par(mfrow=c(2,2))
plot(bgy,type="n",ylim=c(0,2))
abline(0,0,lty=8)
lines(bgy)
lines(bgyplusci,lty=2)
lines(bgyminusci,lty=2)
plot(bgr,type="n",ylim=c(-0.5,1.5))
abline(0,0,lty=8)
lines(bgr)
lines(bgrplusci,lty=2)
lines(bgrminusci,lty=2)
plot(bddifp,type="n",ylim=c(-2.2,3.2))
abline(0,0,lty=8)
lines(bddifp)
lines(bddifpplusci,lty=2)
lines(bddifpminusci,lty=2)
plot(bcons,type="n",ylim=c(-0.04,0.04))
abline(0,0,lty=8)
lines(bcons)
lines(bconplusci,lty=2)
lines(bconminusci,lty=2)
```

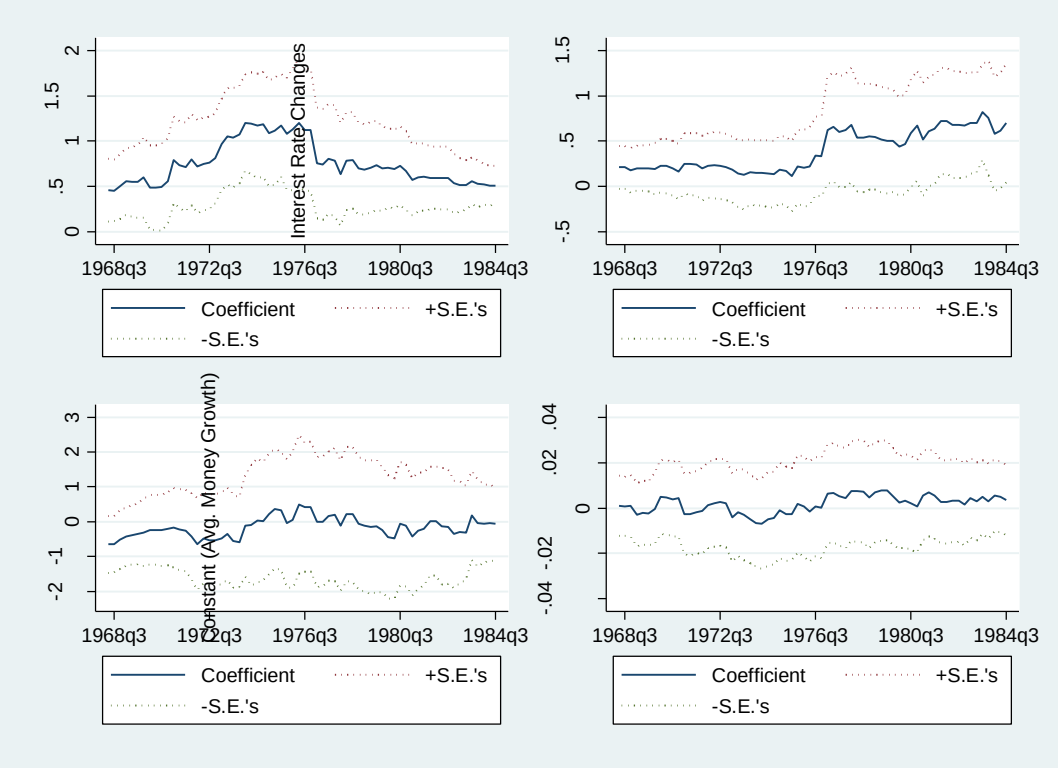
Figure 14. Money Growth Elasticities Against All Variables Plus the Constant



STATA Translation:

```
clear
set mem 100m
insheet using "C:\johansen.csv", comma
generate periods = _n
generate time = periods-8
format time %tq
tsset time
drop if periods==1
gen gm = lrm1-lrm1[_n-1]
gen gy = lny-lny[_n-1]
gen gr = lnmr-lnmr[_n-1]
gen diffdifp = difp-difp[_n-1]

rolling _b _se, window(40) start(1958q3) end(1984q3) clear: reg gm gy gr
diffdifp
gen gyseplus = _b_gy+1.96*_se_gy
gen gyseminus = _b_gy-1.96*_se_gy
twoway (line _b_gy end) (line gyseplus end, lpattern(dot)) (line gyseminus
end, lpattern(dot)), ytitle(GDP Growth) xtitle("") legend(order(1
"Coefficient" 2 "+S.E.'s" 3 "-S.E.'s"))
graph save "C:\GDPGrowth.gph", replace

gen grseplus = _b_gr+1.96*_se_gr
gen grseminus = _b_gr-1.96*_se_gr
twoway (line _b_gr end) (line grseplus end, lpattern(dot)) (line grseminus
end, lpattern(dot)), ytitle(Interest Rate Changes) xtitle("") legend(order(1
"Coefficient" 2 "+S.E.'s" 3 "-S.E.'s"))
graph save "C:\InterestRate.gph", replace

gen diffdifpseplus = _b_diffdifp+1.96*_se_diffdifp
gen diffdifpseminus = _b_diffdifp-1.96*_se_diffdifp
twoway (line _b_diffdifp end) (line diffdifpseplus end, lpattern(dot)) (line
diffdifpseminus end, lpattern(dot)), ytitle(Inflation Acceleration)
xtitle("") legend(order(1 "Coefficient" 2 "+S.E.'s" 3 "-S.E.'s"))
graph save "C:\Inflation.gph", replace

gen conseplus = _b_cons+1.96*_se_cons
gen conseminus = _b_cons-1.96*_se_cons
twoway (line _b_cons end) (line conseplus end, lpattern(dot)) (line
conseminus end, lpattern(dot)), ytitle(Constant (Avg. Money Growth))
xtitle("") legend(order(1 "Coefficient" 2 "+S.E.'s" 3 "-S.E.'s"))
graph save "C:\Constant.gph", replace

graph combine "C:\GDPGrowth.gph" "C:\InterestRate.gph" "C:\Inflation.gph"
"C:\Constant.gph"
```

Figure 15. Money Growth Elasticities Against All Variables Plus the Constant



How about an interpretation? It's not too difficult. You'll see that during the 1970's GDP growth was increasingly correlated with money growth, but that that sensitivity has since declined. At the same time, however, you see that just as the sensitivity of money growth to GDP has fallen, money growth has become more sensitive to interest rate changes. This fact could reflect a change in monetary policy. Perhaps in the 1970's the Finnish Central Bank was following a GDP rule to target inflation (or perhaps not), and perhaps when they realized that wasn't working, they switched to interest rate targests. This is all pure conjecture, and perhaps it's worth exploring, if it hasn't already been looked into. Perhaps this pattern is true for other country's? I don't know.

Higher Frequency Data Analysis: ARMA and GARCH in One Shot

In this section, I can summarize the basics of ARMA and GARCH modelling. These days, in terms of software, my metric for what's a good program to run GARCH models is how many multivariate-, not univariate-, GARCH models it does. I think only S-Plus (not R) and *Ox*© have routines for multivariate GARCH models. I've also seen that Robert Engle has a program for E-Views, which you can download from his web-site, to run a number of multi-variate GARCH models. I'm not sure if it runs "right off the bat" or if you have to do something to your E-Views program to get it to run, but consider looking into it. If you plan to work with high-frequency data, you might consider *Ox*©, which is a program put out by Oxford University, unless you're an expert R programmer, or you can wait for other people to improve what R can do. STATA's not bad, but it's still limited. However, keep in mind that people like Frank Diebold are now calling these models generation I volatility (i.e., they're getting old). The generation II is called realized volatility and betas, and these are quite simple to compute, even though the mathematics look, at times, daunting.

The the quantity and frequency of your data is a crucial component of the kind of estimation you're able to do. You can estimate an auto-regressive moving average (or ARMA) model, which are models of the conditional mean of a variable, with annual data. To estimate generalized auto-regressive conditional heteroskedasticity (or GARCH), which are models of the conditional variance, you'll need higher-frequency data to increase the number of potential observations. You'll probably have to have at least monthly data, and weekly, daily, and intra-daily are better. A rule of thumb that I've read, but can't remember where, is at you'll need at least 100 observations for an ARCH model (I've seen a paper where they estimate an ARCH model with over one-hundred years of annual data), and at least 500 for a GARCH model. Here I'll illustrate with daily data, but in finance you can have intra-daily data, so that you're dealing with millions of data points. I'll do a basic ARMA(1,1)-GARCH(1,1) model, just to illustrate both. But first off, what would such an animal look like?

The ARMA(1,1), means that it's part auto-regressive with 1 lag, and part moving average, with 1 lag. Hence, the conditional mean portion of the model would be

15) $$y_t = \mu + \underbrace{\rho y_{t-1}}_{AR(1)} + \varepsilon_t + \underbrace{\theta \varepsilon_{t-1}}_{MA(1)}$$

So, the μ is the constant portion of the mean, as we've seen in the ADF test (there's no time trend here to simplify), there's one lagged dependent variable, and the ρ is the auto-correlation coefficient, and the θ is the moving average parameter associated with the lagged residual. Before GARCH, most people assumed the errors were drawn from a distribution with a constant variance. However, in the real world, you might see that the volatility can change over time; it would be heteroskedastic. The errors would be said to come from a distribution with a mean equal to zero and a variance expressed by the following equation

16) $$\sigma_t^2 = \omega + \underbrace{\alpha \varepsilon_{t-1}^2}_{ARCH(1)} + \underbrace{\beta \sigma_{t-1}^2}_{GARCH(1)}$$

where ϖ is the constant part of the variance, α captures the marginal impact of lagged squared residuals, and β captures the marginal impact of lagged variance. You should notice that it has the same structure, as the ARMA model, except for the contemporaneous error term. There are, by the way, lots of variations of this. The merits of each should be considered if you plan to do work in this area. You can also put independent variables in either or both of equations 15) or 16). To estimate an ARMA GARCH model in R, you'll need to Install and Load the **`fseries`** package.

Once you have the codes run it's simple to run the model in R, except that I think they are still in the process of developing this program so it looks a bit rough (for instance, I'm still not sure how to calculate the standard errors for the coefficients). As for data, I'll get the Bollerslev and Ghysels (1996) daily British Pound-Deutschmark exchange rate also from http://www.stanford.edu/~clint/bench/#garch, and get the file garch11x.zip. You'll have to save this file as a **`.csv`** file as you've done before to your favourite location. I also changed the names of the variables in the **`.csv`** file to from **Obs** to **obs** and from **Y** to **bpdret**, since it's the daily rate of return on the British Pound-Deutschmark exchange rate. Once you do that you can read in the data, and then run the **`garchFit`** command in R, which will give you the following output, which I'll forgo explaining until I get to STATA, since I think it does a little better.

```
> mydailydata <- read.table("C:/bpdret.csv",header=T,sep=",")
> armagarch <- garchFit(~arma(1,1)+garch(1,1),mydailydata$bpdret)
```

```
        Lots of Output that I've suppressed, and then

                             U          V      params includes
            mu     -1.642679e-01  0.1642679 -0.01641602     TRUE
            ar1    -1.000000e+00  1.0000000 -0.62103194     TRUE
            ma1    -1.000000e+00  1.0000000  0.64278556     TRUE
            omega   2.211298e-07 22.1129849  0.02211298     TRUE
            alpha1  1.000000e-08  1.0000000  0.10000000     TRUE
            gamma1 -1.000000e+00  1.0000000  0.10000000    FALSE
            beta1   1.000000e-08  1.0000000  0.80000000     TRUE
            delta   0.000000e+00  2.0000000  2.00000000    FALSE
            skew    1.000000e-01 10.0000000  1.00000000    FALSE
            shape   1.000000e+00 20.0000000  4.00000000    FALSE

mu            ar1           ma1           omega         alpha1        beta1
-0.00842      -0.37208      0.42763       0.01150       0.16002       0.79608

        More output related to time to convergence and then

mu            ar1           ma1           omega         alpha1        beta1
-0.00842      -0.37208      0.42763       0.01150       0.16002       0.79608

        Hessian Matrix:
                        mu        ar1         ma1        omega       alpha1
        mu     6956.340703  -29.29060    7.146397    -263.2579   -183.51830
        ar1     -29.290599 1767.20855 1803.103297     778.4359     55.09995
        ma1       7.146397 1803.10330 1852.242279     729.8793     54.46767
        omega  -263.257916  778.43594  729.879257 1322300.7769 107048.36951
        alpha1 -183.518302   55.09995   54.467671  107048.3695  16217.57723
        beta1   -27.476697  280.72289  287.251455  179515.6190  19948.07314
                      beta1
        mu        -27.47670
        ar1       280.72289
        ma1       287.25145
        omega  179515.61895
        alpha1  19948.07314
        beta1   29161.65200
```

```
> armagarchcoef <- armagarch@fit$coef

mu           ar1          ma1          omega        alpha1       beta1
-0.00842     -0.37208     0.42763      0.01150      0.16002      0.79608
```

As I said, I'm not sure how you get the standard errors for the coefficients. STATA doesn't give you the same thing for the ARMA terms, but the GARCH terms are the same. To do this in STATA, you'd have to run the following

```
clear
set mem 100m
insheet using "C:\bpdret.csv", comma
tsset obs, daily
arch bpdret, ar(1) ma(1) arch(1) garch(1)

       ARCH family regression -- ARMA disturbances

       Sample:  02jan1960 to 28may1965               Number of obs = 1974
                                                     Wald chi2(2) = 10.86
       Log likelihood = -1103.911                    Prob > chi2 = 0.0044

       bpdret Coef.         Std. Err.     z      P>z    [95% Conf.    Interval]
       _cons  -.006106      .0087586     -0.70  0.486   -.0232725     .0110605

       ARMA
       ar
       L1.    -.4098897     .3069457     -1.34  0.182  -1.011492      .1917128
       ma
       L1.     .4645401     .2992367      1.55  0.121   -.121953     1.051033

       ARCH
       arch
       L1.     .1602145     .0146999     10.90  0.000    .1314033     .1890258
       garch
       L1.     .7957788     .0176828     45.00  0.000    .7611211     .8304364
       _cons   .0115303     .0014334      8.04  0.000    .0087209     .0143397
```

How would I interpret this? Well, the constant part of the mean exchange rate return is –0.6%, which means that during the sample, the British pound was appreciating, although the result is not statistically significant; i.e., since pounds are in the numerator, one pound buys more Deutschmarks (or one Deutschmark buys fewer pounds). Also, there is evidence of negative serial correlation, since ρ = –0.41, implying that if it's up today, it's likely to be down tomorrow. The p-value is not too high, but leave it in since it's better to leave in an irrelevant variable than to omit a relevant variable (as before, omitted variable bias is worse than including an irrelevant variable). Finally, the MA term is 0.46, with a p-value that is lower, reflecting a higher significance level than the AR term. Now for the ARCH and GARCH terms. Well, you see the ARCH term is smaller in magnitude than the GARCH term, but both are statistically significant. Also, it is a good thing to check if the GARCH terms are not explosive, by adding up the two coefficients α and β. Just as the ρ <1 means that the series is not explosive, since the sum of the squares of the two ARCH and GARCH coefficients is $0.16^2 + 0.79^2 = 0.0256 + 0.6241 = 0.6497$, likewise, the volatility is not explosive.

Note, however, that there's a slight problem here with the dates. I'm not exactly sure when the Bollerslev-Ghysels (1996) data is from, but I'm pretty sure it's not from those dates. However, I also was not too careful about the dates in this context, since

I only needed to tell STATA to time-series set (`tsset`) the data, and working with daily dates can be more of a problem in STATA than in R, which I believe allows for irregularly spaced time-series, which is what you have because of weekends and holidays (intra-daily dates are even worse!).

GARCH with Ox©

Having just noted that R is not a well developed GARCH modelling environment, there are alternatives, including *Ox©*. There are two versions of *Ox©*, the very user-friendly and very expensive Professional version, and the very clumsy, but freely downloadable Console version. This section is intended to give you an understanding of how to start working with *Ox©*; it is not intended to teach you the "ins and outs" of GARCH modelling. To get access to the Console version, you may go to the following web-site http://www.doornik.com/download.html. You will see a link

Download Ox Console (all platforms)

When you left click on this button, you will be prompted for an e-mail address and additional information. If you agree to the terms of use (and you should), then you will receive an e-mail shortly thereafter, and and the very bottom of the message you will see the following link

http://www.doornik.com/download/oxmetrics5/c645br/oxcons.html

You will see two links (you can use either one) that allow you to download the Console version of *Ox©*. Once you download this, there are two additional packages you may wish to download. To get them, go back to

http://www.doornik.com/download.html

As you scroll down the page, the first package you may wish to use is the Arfima package, which allows you to estimate models when the underlying data is fractionally integrated, which relates to long range dependence, the last topic covered in this cookbook, high frequency data and the R/S statistic. The second is the G@RCH package, which allows you to estimate a whole range of univariate and multivariate GARCH models. When you download these zipped packages, the *Ox©* documentation asks you to make sure that you include them in the packages folder, such as, which might be found in a location such as

"C:\Program Files\OxMetrics5\Ox\packages"

To estimate a GARCH model in *Ox©* the first thing to do is to start with the working example. There is a working example in the G@RCH package examples folder, which might be found in a location such as

"C:\Program Files\OxMetrics5\Ox\packages\Garch42\Garch42\Examples"

[Note: the Garch42 in "packages\Garch42\Garch42\" appears twice because when I unzipped the package in the "packages" folder, it created a folder within "Garch42"

called “Garch42.” I left it as is, but you may eliminate the second folder after copying all contents to “packages\Garch42”]

The G@RCH 4.0 version has an example called “GarchEstim.ox.” To run it, the first thing to note is that you’ll have to change the location of the file “garch” in the first line of the codes from **#import <packages/Garch42/garch>** to **#import <packages/Garch42/Garch42/garch>** so that the correct address is called. Once you change the first line, you can run the model and it will give you results for the Nasdaq for a simple GARCH model with a constant conditional mean for the first 2000 observations of the 1984-1999 sample:

```
********************
 ** SPECIFICATIONS **
 ********************
Dependent variable : Nasdaq
Mean Equation : ARMA (0, 0) model.
No regressor in the mean
Variance Equation : GARCH (1, 1) model.
 No regressor in the variance
The distribution is a Gauss distribution.

Strong convergence using numerical derivatives
Log-likelihood = -2192.99
Please wait : Computing the Std Errors ...

Robust Standard Errors (Sandwich formula)
            Coefficient Std.Error   t-value     t-prob
Cst(M)      0.078371    0.018303    4.282       0.0000
Cst(V)      0.038572    0.014600    2.642       0.0083
ARCH(Alpha1)0.193262    0.051737    3.735       0.0002
GARCH(Beta1)0.762596    0.057457    13.27       0.0000

No. Observations :      2000  No. Parameters  :         4
Mean (Y)         :   0.04326  Variance (Y)    :   0.83876
Skewness (Y)     :  -2.33675  Kurtosis (Y)    :  33.73842
Log Likelihood   : -2192.990  Alpha[1]+Beta[1]:   0.95586

The sample mean of squared residuals was used to start
recursion.
The positivity constraint for the GARCH (1,1) is observed.
This constraint is alpha[L]/[1 - beta(L)] >= 0.
The unconditional variance is 0.873822
The conditions are alpha[0] > 0, alpha[L] + beta[L] < 1 and
alpha[i] + beta[i] >= 0.
  => See Doornik & Ooms (2001) for more details.
The condition for existence of the fourth moment of the GARCH
is observed.
The constraint equals 0.988364 and should be < 1.
  => See Ling & McAleer (2001) for details.

Estimated Parameters Vector :
 0.078371; 0.038572; 0.193262; 0.762596

Elapsed Time : 0.594 seconds (or 0.0099 minutes).
```

Now, from this exercise you should be able to estimate the ARMA(1,1)-GARCH(1,1) model from that we tried earlier in R and STATA. Before doing that, note that the

only thing you would have to change to apply the model just estimated to the exchange rate data that you used earlier, is to change the data object,

```
garchobj.Load("/data/garch11x/garch11.xls");
```

as well as the sample from 2000 to 1974 in

```
garchobj.SetSelSample(-1, 1, 1974, 1);
```

To estimate the model with an ARMA(1,1) specification, you may run the following

```
#import <packages/Garch42/Garch42/garch>

main()
{
      decl garchobj;

      garchobj = new Garch();

//*** DATA ***//
      garchobj.Load("/data/garch11x/garch11.xls");
      garchobj.Info();

      garchobj.Select(Y_VAR, {"Y",0,0});
// TO INCLUDE A REGRESSOR IN THE VARIANCE INCLUDE THE FOLLOWING LINE
//    garchobj.Select(Z_VAR, {"NAME",0,0});           //

      garchobj.SetSelSample(-1, 1, 1974, 1);

//*** SPECIFICATIONS ***//
      garchobj.CSTS(1,1);     // cst in Mean (1 or 0), cst in
                              // Variance (1 or 0)
      garchobj.DISTRI(0);     // 0 for Gauss, 1 for Student, 2 for
                              // GED, 3 for Skewed-Student
      garchobj.ARMA_ORDERS(1,1);    // AR order (p), MA order (q).
      garchobj.ARFIMA(0);     // 1 if Arfima wanted, 0 otherwise
      garchobj.GARCH_ORDERS(1,1);   // p order, q order
      garchobj.ARCH_IN_MEAN(0);     // ARCH-in-mean: 1 or 2 to add
                                    // the variance or std. dev in
                                    //  the cond. mean

      garchobj.MODEL("GARCH");      //    0: RISKMETRICS  1:GARCH
                        //    2:EGARCH    3:GJR 4:APARCH
                        //    5:IGARCH     6:FIGARCH-BBM
                        //    7:FIGARCH-CHUNG     8:FIEGARCH
                        //    9:FIAPARCH-BBM
                        //    10: FIAPARCH-CHUNG      11: HYGARCH
      garchobj.TRUNC(1000);
            // Truncation order (only F.I. models with BBM method)

//*** TESTS & FORECASTS ***//
      garchobj.BOXPIERCE(<10;15;20>);
// Lags for the Box-Pierce Q-statistics, <> otherwise
      garchobj.ARCHLAGS(<2;5;10>);
// Lags for Engle's LM ARCH test, <> otherwise
      garchobj.NYBLOM(1);
// 1 to compute the Nyblom stability test, 0 otherwise
      garchobj.SBT(1);
// 1 to compute the Sign Bias test, 0 otherwise
```

```
      garchobj.PEARSON(<40;50;60>);

// Cells (<40;50;60>) for the adjusted Pearson Chi-square
// Goodness-of-fit test, <> otherwise //G@RCH1.12

      garchobj.RBD(<10;15;20>);

// Lags for the Residual-Based Diagnostic test of Tse, <> otherwise

      garchobj.FORECAST(0,15,0);    // Arg.1 : 1 to launch the
                                    // forecasting procedure, 0
                                    // otherwize
                                    // Arg.2 : Number of forecasts
                                    // Arg.3 : 1 to Print the
                                    // forecasts, 0 otherwise

//*** OUTPUT ***//
      garchobj.MLE(2);          // 0 : MLE (Second derivatives),
                                // 1 : MLE (OPG Matrix),
                                // 2 : QMLE
      garchobj.COVAR(0);        // if 1, prints variance-covariance
                                // matrix of the parameters.
      garchobj.ITER(0);         // Interval of iterations between
                                // printed intermediary results (if
                                // no intermediary results wanted,
                                // enter '0')
      garchobj.TESTS(0,0);      // Arg. 1 : 1 to run tests PRIOR to
                                // estimation, 0 otherwise
                                // Arg. 2 : 1 to run tests AFTER
                                // estimation, 0 otherwise
      garchobj.GRAPHS(0,0,"");// Arg.1 : if 1, displays graphics of
                                // the estimations (only when
                                // using GiveWin).
                                // Arg.2 : if 1, saves these graphics
                                // in a EPS file (OK with all Ox
                                // versions)
                                // Arg.3 : Name of the saved file.

      garchobj.FOREGRAPHS(0,0,"");

// Same as GRAPHS(p,s,n) but for the graphics of the forecasts.

//*** PARAMETERS ***//
      garchobj.BOUNDS(0);       // 1 if bounded parameters wanted,
                                // 0 otherwise
      garchobj.FIXPARAM(0,<0;0;0;0;1;0>);

// Arg.1 : 1 to fix some parameters to their starting values,
// 0 otherwize
// Arg.2 : 1 to fix (see garchobj.DoEstimation(<>)) and 0 to estimate
// the corresponding parameter

//*** ESTIMATION ***//

      garchobj.MAXSA(0,5,0.5,20,5,2,1);

// Arg.1 : 1 to use the MaxSA algorithm of Goffe, Ferrier and Rogers
// (1994)
//         and implemented in Ox by Charles Bos
// Arg.2 : dT=initial temperature
// Arg.3 : dRt=temperature reduction factor
```

```
// Arg.4 : iNS=number of cycles
// Arg.5 : iNT=Number of iterations before temperature reduction
// Arg.6 : vC=step length adjustment
// Arg.7 : vM=step length vector used in initial step

      garchobj.Initialization(<>);

// m_vPar = m_clevel | m_vbetam |  m_dARFI | m_vAR | m_vMA |
// m_calpha0 | m_vgammav | m_dD |  m_vbetav |
// m_valphav | m_vleverage | m_vtheta1 | m_vtheta2 | m_vpsy |
// m_ddelta | m_cA | m_cV | m_vHY | m_v_in_mean

      garchobj.PrintStartValues(0); // 1: Prints the S.V. in a table
                                    // form; 2: Individually;
                        //  3: in a Ox code to use in StartValues
      garchobj.PrintBounds(1);
      garchobj.DoEstimation(<>);
      garchobj.Output();
      garchobj.STORE(0,0,0,1,1,"01",0);
                        // Arg.1,2,3,4,5 : if 1 -> stored.
                        // (Res-SqRes-CondV-MeanFor-VarFor)
                        // Arg.6 : Suffix. The name of the saved
                        // series will be "Res_ARG6" (or
                        // "MeanFor_ARG6", ...).
                        // Arg.7 : if 0, saves as an Excel
                        // spreadsheet (.xls). If 1, saves as a
                        // GiveWin dataset (.in7)

      delete garchobj;
}
```

The codes look quite messy, but it will give you the following output.

```
*******************
 ** SPECIFICATIONS **
 *******************
Dependent variable : Y
Mean Equation : ARMA (1, 1) model.
No regressor in the mean
Variance Equation : GARCH (1, 1) model.
 No regressor in the variance
The distribution is a Gauss distribution.

Strong convergence using numerical derivatives
Log-likelihood = -1103.88
Please wait : Computing the Std Errors ...

 Robust Standard Errors (Sandwich formula)
                 Coefficient  Std.Error  t-value  t-prob
Cst(M)             -0.006128  0.0093623  -0.6546  0.5128
AR(1)              -0.371991    0.26599   -1.398  0.1621
MA(1)               0.427544    0.25927    1.649  0.0993
Cst(V)              0.011502  0.0063498    1.811  0.0702
ARCH(Alpha1)        0.160304   0.051525    3.111  0.0019
GARCH(Beta1)        0.795984   0.068860    11.56  0.0000

No. Observations :      1974  No. Parameters  :         6
Mean (Y)         :  -0.01643  Variance (Y)    :   0.22102
Skewness (Y)     :  -0.24951  Kurtosis (Y)    :   6.62765
Log Likelihood   : -1103.881  Alpha[1]+Beta[1]:   0.95629

The sample mean of squared residuals was used to start
recursion.
The positivity constraint for the GARCH (1,1) is observed.
This constraint is alpha[L]/[1 - beta(L)] >= 0.
The unconditional variance is 0.263129
The conditions are alpha[0] > 0, alpha[L] + beta[L] < 1 and
alpha[i] + beta[i] >= 0.
  => See Doornik & Ooms (2001) for more details.
The condition for existence of the fourth moment of the GARCH
is observed.
The constraint equals 0.965882 and should be < 1.
  => See Ling & McAleer (2001) for details.

Estimated Parameters Vector :
-0.006128;-0.371991; 0.427544; 0.011502; 0.160304; 0.795984

Elapsed Time : 1.187 seconds (or 0.0197833 minutes).
```

Higher Frequency Data Analysis: The Range Scale or R/S Statistic

I'll finish off with one more technique for time series analysis, essentially because it is so simple, yet it is quite useful. But for the simplicity of the Pfaff's R codes, I would have passed it over. Hurst (1951), who was looking at flooding in reservoirs, proposed a statistic to analyze the long-range dependence of the data. The idea is to compute the Hurst statistic. It has in the numerator the difference between the maximum and the minimum of the series in question. In the denominator, there is the standard deviation, as a measure of scale. If the Hurst statistics has the properties $0.5 < H \leq 1$, then the series is persistant. If $H = 0.5$, then there is no persistence, and finally, if the Hurst statistic is $0 \leq H < 0.5$, then the series is anti-persistent.

```
> timemean <- mean(mydailydata$bpdret)
> bpd.dm <- mydailydata$bpdret-timemean
> max.bpdret <- max(cumsum(bpd.dm))
> min.bpdret <- min(cumsum(bpd.dm))
> sd.bpdret <- sd(bpd.dm)
> RS <- (max.bpdret - min.bpdret)/sd.bpdret
> H <- log(RS)/log(length(bpd.dm))
       [1] 0.5620137
> d <- H - 0.5
       [1] 0.06201372
```

So what you see is that the BP-DM exchange rate is mildly persistant, since H is above 0.5. The `d` is the fractional difference parameter. This is Hurst's way to estimate it and there are other ways. If the `d` is greater than 0.5, then the series is explosive. If it's between 0 and 0.5, then the data is a long memory process, or persistant. If it's between –0.5 and 0, then it's intermediate memory process, or anti-persistant. At zero, the series exhibits no memory (think a random walk). The STATA codes below generate the same output.

STATA translation:

```
clear
set mem 100m
insheet using "C:\bpdret.csv", comma
tsset obs, daily
egen timemean = mean(bpdret)
gen bpddm = bpdret-timemean
gen cumsumbpddm=sum(bpddm)
egen maxbpdret = max(cumsumbpddm)
egen minbpdret = min(cumsumbpddm)
egen sdbpdret = sd(bpddm)
gen RS = (maxbpdret - minbpdret)/sdbpdret
gen sample = _N
gen H = log(RS)/log(sample)
gen d = H - 0.5
sum H d
```

```
    Variable Obs     Mean   Std. Dev.        Min          Max
    H        1974 .5620137     0            .5620137     .5620137
    d        1974 .0620137     0            .0620137     .0620137
```

Conclusions

Depending on your tastes (i.e., whether you like to know how to make the sausages, or you just like to eat them) you may find that R or STATA, or another program is more to your liking. You should have seen that for some things, like OLS, you get identical results, while for other things (anything related to Maximum Likelihood, like ordered logit, GARCH, random effects, because of the starting values) you can get different results. You should also have seen that neither program is best in all things. I still think R always has the greater potential, but that does not stop me from being interested in learning how to use STATA better. I still find R to be challenging to grasp (and at times it is even frustrating for a non-programmer like me), but still I often find that it is worth taking the challenge. In fact, writing this helped me figure out some of the mysteries in R, and to make the mental switch from S-Plus. So I thank the readers of this cookbook who were my inspiration on the demand side, while I again thank Grant Farnsworth, whose Rosetta-Stone "Econometrics with R" was my inspiration on the supply side, since it provided me with the technology to think about R in the language of econometrics rather than statistics.

References

Baltagi, Badi. (2001) *Econometric Analysis of Panel Data*. 2nd Ed. Wiley: Chichester, England.

Bollerslev, Tim and Eric Ghysels. (1996) "Periodic Autoregressive Conditional Heteroskedasticity." *Journal of Business and Economic Statistics* 14, 139-151.

Boot, J.C.G. and G.M. DeWit. (1960) "Investment Demand: An Empirical Contribution to the Aggregation Problem." *International Economic Review*, 3-30.

Brooks, Chris (2002) *Introductory Econometrics for Finance*. Cambridge University Press: Cambridge, UK.

Cameron, A. Colin and Pravin K. Trivedi. (1986) "Econometric Models Based on Count Data: Comparisons and Applications of Some Estimators and Tests." *Journal of Applied Econometrics* 1, 29-54.

Charnes, A., Cooper, W. and R. Ferguson. (1955) "Optimal Estimation of Executive Compensation by Linear Programming." *Management Science* 1, 138-151.

Cleveland, William S. (1993) *Visualizing Data*. Hobart Press: Summit, NJ.

Dewald, William G., Thursby, Jerry G., and Richard G. Anderson. (1986) "Replication in Empirical Economics: The Journal of Money, Credit and Banking Project." The American Economic Review 76, 587-603.

Efron, B. and Tibshirani, R. (1993) *An Introduction to the Bootstrap*. Chapman and Hall: New York, NY.

Fama, Eugene. (1965) "The Behavior of Stock Market Prices." *Journal of Business* 38, 34-105.

Farnsworth, Grant (2006) Econometrics with R. Available from: http://cran.r-project.org/doc/contrib/Farnsworth-EconometricsInR.pdf.

Fisher, R. A. (1958) *Statistical Methods for Research Workers*. 13th Edition. Hafner: New York.

Fu, Vincent. (1998) "Estimating Generalized Ordered Logit Models." *Stata Technical Bulletin* 8, 160-164.

Hurst, H. (1951) "Long Term Storage Capacity of Reservoirs." Transactions of the American Society of Civil Engineers 116, 770-799.

Johansen, and Juselius. (1991) "Maximum Likelihood Estimation and Inference on Cointegration - With Applications to the Demand for Money." *Oxford Bulletin of Economics and Statistics* 52, 169-210.

Koenker, Roger. (2006) "Quantile Regression in R: A Vignette." Version 4.05. http://www.r-project.org

Koenker, Roger and Gilbert Bassett, Jr. (1978) "Regression Quantiles." *Econometrica* 46, 33-50.

----------, ----------. (1982) "Robust Tests for Heteroscedasticity Based on Regression Quantiles." *Econometrica* 50, 43-62.

Levy, David M. (1992) *The Economic Ideas of Ordinary People*. Routledge: London.

----------. (2001) *How the Dismal Science Got Its Name*. University of Michigan Press: Ann Arbor, MI.

Long, J. (1997) *Regression Models for Categorical and Limited Dependent Variables*. Sage Publications: Thousand Oaks, CA.

Mankiw, N. Gregory, David Romer, and David N. Weil. (1992) "A Contribution to the Empirics of Economic Growth." *Quarterly Journal of Economics* 107, 407-37.

Mathsoft. (1999) *S-PLUS 2000 Programmer's Guide*. Data Analysis Products Division, MathSoft: Seattle, WA.

Park, Hun Myoung. (2005) "Linear Regression Models for Panel Data Using SAS, STATA, LIMDEP, and SPSS." Available from: http://www.indiana.edu/~statmath/stat/all/panel/panel.pdf

Peart, Sandra and David Levy. (2005) *The Vanity of the Philosopher*. University of Michigan Press: Ann Arbor, MI.

Pfaff, Bernhard. (2006) *Analysis of Integrated and Cointegrated Time Series with R*. Springer: New York).

Smith, Adam. (1981) *An Inquiry into the Nature and Causes of the Wealth of Nations*. Liberty Fund: Indianapolis, IN.

Summer, Robert, and Alan Heston. (1988) "A New Set of International Comparisons of Real Product and Price Level Estimates for 130 Countries, 1950-85." Review of Income and Wealth, 34, 1-26.

Temple, Jonathan. (1998) "Robustness Tests of the Augmented Solow Model." *Journal of Applied Econometrics* 13, 361-375.

Williams, Richard. (2006) "Generalized Ordered Logit/ Partial Proportional Odds Models for Ordinal Dependent Variables." *The Stata Journal* 6, 58-82.

Zellner, Arnold. (1961) "Econometric Estimation with Temporally Dependent Disturbance Terms." International Economic Review 2, 164-178.

Appendix 1: A More Concrete Expression of the Relationship between the OLS and Median Regression

There are several ways to understand the relationship between OLS and the median regression. One way is to think about how OLS is usually defined as the solution to the following minimization problem

$$\min_{\{\beta_0,\beta_1,\ldots\beta_n\}} \sum_{t=1}^{T} \varepsilon_t^2$$

or

$$\min_{\{\beta_0,\beta_1,\ldots\beta_n\}} \sum_{t=1}^{T} \left(y_t - \beta_0 - \beta_1 x_{1t} - \beta_2 x_{2t} - \cdots - \beta_n x_{nt}\right)^2$$

where ε_t is the error term, y_t is the "dependent" variable, β_0 is the intercept, and β_i are the slope coefficients, representing the "marginal impact" of a the ith "independent variable", x_{it}. Without really changing the problem, the parentheses can be replaced with the absolute value operator

$$\min_{\{\beta_0,\beta_1,\ldots\beta_n\}} \sum_{t=1}^{T} \left|\varepsilon_t\right|^2$$

or

$$\min_{\{\beta_0,\beta_1,\ldots\beta_n\}} \sum_{t=1}^{T} \left|y_t - \beta_0 - \beta_1 x_{1t} - \beta_2 x_{2t} - \cdots - \beta_n x_{nt}\right|^2$$

Written this way, it is clear that OLS is a special example of what's called an L^p estimator, in this case with p = 2 because the residuals are squared. The least absolute deviations (LAD) or least absolute errors (LAE) regression, which may also simply be called the median regression, is the analogous L^1 estimator, represented as the sum of absolute errors

$$\min_{\{\beta_0,\beta_1,\ldots\beta_n\}} \sum_{t=1}^{T} \left|\varepsilon_t\right|$$

or

$$\min_{\{\beta_0,\beta_1,\ldots\beta_n\}} \sum_{t=1}^{T} \left|y_t - \beta_0 - \beta_1 x_{1t} - \beta_2 x_{2t} - \cdots - \beta_n x_{nt}\right|$$

Unlike the sum of squares problem, from which the normal equations can be solved using calculus, the sum of absolute errors problem cannot. In fact, it was not until Charnes, Cooper and Ferguson (1955) figured out how to apply piece-wise linear programming in the 1950's that the median regression could be implemented.

Appendix 2: The Median Regression as a Special Case of Quantile Regressions

So far, we have seen the relationship between the mean (OLS) and median (LAD) regressions. Remember that the median is also the 50th percentile, among other things. Therefore, another way to think about it is that it is only one case of many possible quantile regressions. To understand how piece-wise linear programming fits in here, consider rewriting the problem for the median regression as follows

$$\min_{\{\beta_0,\beta_1,\ldots\beta_n\}} 0.5\varepsilon^+ + 0.5\varepsilon^-$$
$$s.t. \quad y_t = \beta_0 + \beta_1 x_{1t} + \beta_2 x_{2t} + \cdots + \beta_n x_{nt} + \varepsilon^+ - \varepsilon^-$$

where all notation is the same, except that ε^+ now represents any residual that is above the median, and ε^- is any residual below the median. Unlike OLS, the median estimator actually goes through the $n + 1$ data points, since it represents an actual observation. So, if you have 10 right-hand-side variables (including the intercept), you will know that the median estimator passes through ten data points.

So, what does that mathematical notation above mean? In words, it says, first you get fitted values that are subtracted from the dependent variable to give you residuals, second, you rank/sort the residuals, and third, if you fit a line through the tau-percentile residual, in this case the 50th percentile, that represents the tau-percentile regression line, in this case the median. From this we see that you pick any percentile, tau, and you will get the corresponding quantile regression line as the solution to the following problem

$$\min_{\{\beta_0,\beta_1,\ldots\beta_n\}} \tau\varepsilon^+ + (1-\tau)\varepsilon^-$$
$$s.t. \quad y_t = \beta_0 + \beta_1 x_{1t} + \beta_2 x_{2t} + \cdots + \beta_n x_{nt} + \varepsilon^+ - \varepsilon^-$$

Appendix 3: Ordered and Generalized Ordered Logit

Ordered logit is a Maximum Likelihood Estimator (MLE), which can be used to estimate the likelihood that the household will make a certain number of visits to the doctor, and can be expressed in this case as follows

$$L = p(v = 0|\boldsymbol{X}, b, c_1, c_2) \cdot p(v = 1|\boldsymbol{X}, b, c_1, c_2) \cdot p(v = 2 \mid \boldsymbol{X}, b, c_1, c_2) \cdot p(v = 3 \text{ or more} \mid \boldsymbol{X}, b, c_1, c_2)$$

$$= F(c_1 - \boldsymbol{X}b) \cdot [F(c_2 - \boldsymbol{X}b) - F(c_1 - \boldsymbol{X}b)] \cdot [F(c_3 - \boldsymbol{X}b) - F(c_2 - \boldsymbol{X}b)] \cdot [1 - F(c_3 - \boldsymbol{X}b)]$$

where in the first line, p(v = j | ...) denotes "probability the number of visits *v* equals i given …," $\boldsymbol{X}$ is a matrix of exogenous factors that explains variation in visits to the doctor by the i^{th} person", *b* is an estimate of the sensitivity of an exogenous variable on the probability of visits *y* equal to j, and c_1 , c_2, and c_3 are the thresholds at which the number of visits *y* takes on a new higher value, as depicted in the graph above. In the second line, F(·) refers to the cumulative distribution function (cdf), the first term on the right hand side is the cdf up to the first cut point, the second and third terms, in brackets, are the cdf between the first and second cut points, and second and third cut points, respectively, and the fourth term, also in brackets, is the cdf between the highest cut point and one. Taking logs of both sides yields

$$\ln L(b, c_1, c_2 \mid y, \boldsymbol{X}) = \ln[F(c_1 - \boldsymbol{X}b)] + \ln[F(c_2 - \boldsymbol{X}b) - F(c_1 - \boldsymbol{X}b)] + \ln[F(c_3 - \boldsymbol{X}b) - F(c_2 - \boldsymbol{X}b)] + \ln[1 - F(c_3 - \boldsymbol{X}b)]$$

An optimal *b* is computed using a maximum likelihood estimator (MLE). After computing the MLE coefficients, an intuitive way to interpret the statistical output is to consider the odds ratio as a measure of the marginal effect of an increase in the independent variable on the probability of observing a specific number of visits to the doctor. First, consider the odds making "3" visits as opposed to making more than "3" visits, which is the same as "4" here. This can expressed as follows

$$\begin{aligned} \text{Odds} &= p(\text{visits} \le 3 \mid \boldsymbol{X})/[1 - p(\text{visits} \le 3 \mid \boldsymbol{X})] \\ &= p(\text{visits} \le 3 \mid \boldsymbol{X})/p(\text{visits} > 3 \mid \boldsymbol{X}) \\ &= e^{c_3 - Xb} \end{aligned}$$

By considering a change in one of the variables in $\boldsymbol{X}$, it is possible compute the so-called odds ratio, the ratio of the likelihood of that the person makes a specific number of visits to the doctor at the new $\boldsymbol{X}$ values relative to the old $\boldsymbol{X}$ values, or

$$\begin{aligned} \text{Odds ratio} &= \frac{p(\text{visits} \le 3 \mid \boldsymbol{X}_{\text{new}})/p(\text{visits} > 3 \mid \boldsymbol{X}_{\text{new}})}{p(\text{visits} \le 3 \mid \boldsymbol{X}_{\text{old}})/p(\text{visits} > 3 \mid \boldsymbol{X}_{\text{old}})} \\ &= \frac{e^{c_3 - X_{new}b}}{e^{c_3 - X_{old}b}} \\ &= e^{[X_{old} - X_{new}]b} \end{aligned}$$

According to Fu (http://www.bol.ucla.edu/~vfu/gologitfaq.html), there is apparently no formal derivation of the Generalized Ordered Logit estimator. He lists eight references pointing to the possibility of this estimator. Essentially Fu's gologit relaxes the assumption that the *b*'s are identical across thresholds. In the context of predicting the number of visits to the doctor, the log likelihood function becomes

$$L = p(v = 0|\, \boldsymbol{X}, b_1, b_2, b_3, c_1, c_2, c_3) \cdot p(v = 1|\, \boldsymbol{X}, b_1, b_2, b_3, c_1, c_2, c_3) \cdot p(v = 2 \mid \boldsymbol{X}, b_1, b_2, b_3, c_1, c_2, c_3) \cdot p(v = 3 \text{ or more} \mid \boldsymbol{X}, b_1, b_2, b_3, c_1, c_2, c_3)$$

$$= F(c_1 - \boldsymbol{X} b_1) \cdot [F(c_2 - \boldsymbol{X} b_2) - F(c_1 - \boldsymbol{X} b_1)] \cdot [F(c_3 - \boldsymbol{X} b_3) - F(c_2 - \boldsymbol{X} b_2)] \cdot [1 - F(c_3 - \boldsymbol{X} b_3)]$$

where p(y = i | ...) again denotes "probability the number of visits to the doctor, *y* equals j given …," $\boldsymbol{X}$ is again the matrix of exogenous factors used to explain the number of visits to the doctor by the individual, b_j is an estimate of the sensitivity of an exogenous variable on the probability of event *y* at threshold j, and c_1, and c_2 are the thresholds at which *y* takes on a new higher value, as depicted in the graph above. Taking logs of both sides yields the following assumed log-likelihood function

$$\ln L(b_1, b_2, c_1, c_2 \mid y, \boldsymbol{X}) = \ln[F(c_1 - \boldsymbol{X} b_1)] + \ln[F(c_2 - \boldsymbol{X} b_2) - F(c_1 - \boldsymbol{X} b_1)] + \ln[F(c_3 - \boldsymbol{X} b_3) - F(c_2 - \boldsymbol{X} b_2)] + \ln[1 - F(c_3 - \boldsymbol{X} b_3)]$$

The generalized odds ratios now appear as follows

$$e^{X_{old} b_i - X_{new} b_i}$$

Appendix 4: Penn World Table Country Data Grades

country	grade	country	grade	country	grade
Algeria	D	israel	B	taiwan	D
Angola	D	italy	A	tanzania	C
argentina	B	jamaica	C	thailand	C
australia	A	japan	A	togo	D
austria	A	jordan	C	trinidad	C
bahrain	C	Kenya	C	tunisia	C
bangladesh	C	korea	B	turkey	C
barbados	C	kuwait	C	uganda	D
belgium	A	Lesotho	D	uae	D
Benin	C	Liberia	D	uk	A
bolivia	C	luxembourg	A	uruguay	B
Botswana	C	Madagascar	C	usa	A
brazil	C	Malawi	C	venezeula	C
BurkinaFaso	C	malaysia	C	yemen	D
Burundi	C	Mali	C	zambia	C
Cameroon	C	malta	D	zimbabwe	C
canada	A	Mauritania	C		
CentralAfr.Rep.	D	Mauritius	C		
Chad	D	mexico	C		
chile	B	Morocco	C		
colombia	C	Mozambique	D		
zaire	D	myanmar	D		
Congo	C	nepal	C		
costarica	C	netherlands	A		
cyprus	D	newzealand	B		
denmark	A	nicaragua	C		
dominicanrep	C	Niger	D		
ecuador	C	Nigeria	C		
Egypt	C	norway	A		
elsalvador	C	oman	C		
Ethiopia	C	pakistan	C		
fiji	C	panama	C		
finland	A	papuanewguinea	D		
france	A	paraguay	C		
Gabon	C	peru	C		
Gambia	C	philippines	C		
westgermany	B	Rwanda	C		
Ghana	C	Saudiarabia	D		
greece	B	Senegal	C		
guatemala	C	SierraLeone	C		
Guinea	C	singapore	B		
guyana	D	Somalia	D		
haiti	D	SAfrica	C		
honduras	C	spain	B		
HongKong	A	Srilanka	C		
iceland	B	sudan	D		
india	C	surinam	D		
indonesia	C	swaziland	C		
iran	C	sweden	A		
iraq	D	switzerland	A		
ireland	A	syria	C		

Appendix 5: R and STATA Codes if You'd Like to Cut To the Chase

R Code:

```
# Codes to load in basic data and get summary statistics
mydata <- read.table("C:/temple.csv",header=T,sep=",")
summary(mydata$DY)
summary(mydata)
sd(mydata$DY, na.rm = T)
median(mydata$DY, na.rm = T)
mad(mydata$DY, na.rm = T)
quantile(mydata$DY,probs=seq(0,1,0.25),na.rm=T)

# Codes to estimate regressions and generate regression summaries and
# replicate MRW
myreg <- lm(DY~LGDP60+LNGD+LINV,data=mydata)
summary(myreg)
myaugreg <- lm(DY~LGDP60+LNGD+LINV+LSCH,data=mydata)
summary(myaugreg)
myregnsam <- lm(DY~LGDP60+LNGD+LINV,data=subset(mydata,mydata$NSAM == 1))
summary(myregnsam)
myregisam <- lm(DY~LGDP60+LNGD+LINV,data=subset(mydata,mydata$ISAM == 1))
summary(myregisam)
myregosam <- lm(DY~LGDP60+LNGD+LINV,data=subset(mydata,mydata$OSAM == 1))
summary(myregosam)

# Codes to generate histogram with normal density plot of the residuals
# of myaugreg
augsolres <- myaugreg$residuals
hist(augsolres,br=20,freq = FALSE)
new.x <- seq(min(augsolres), max(augsolres), length = length(augsolres))
new.dens <- dnorm(new.x, mean = mean(augsolres), sd = sd(augsolres))
lines(new.x, new.dens)

# Codes to generate quantile-quantile normal plots for a random sample
set.seed(91)
rs <- rnorm(15)
cbind(sort(rs))
plot(qnorm(ppoints(15),mean=mean(rs),sd=sd(rs)),sort(rs))
abline(0,1)
qqnorm(rs)
abline(mean(rs),sd(rs))
qqnorm(rs)
qqline(rs)

# Codes to generate quantile-quantile normal plots of the residuals
# of myaugreg
qqnorm(augsolres)
qqline(augsolres)

# Codes to generate quantile-quantile normal plots of the residuals
# of myaugreg with more detail
qqnorm(augsolres,main="Normal Q-Q Plot",xlab="Normal
Quantiles",ylab="Residual Quantiles")
qqline(augsolres)
text(-1.5,-0.8,"Fat-Tailed")
text(-2,-0.3,"Light-Tailed")
text(2,0.3,"Light-Tailed")
text(1.7,0.85,"Fat-Tailed")

# Codes to generate basic scatterplot matrix with lattice package
mydata <- read.table("C:/temple.csv",header=T,sep=",")
mymatrix <-
cbind(mydata$DY,mydata$LGDP60,mydata$LNGD,mydata$LINV,mydata$LSCH)
splom(~mymatrix, varnames = c("DY","LGDP60","LNGD", "LINV","LSCH"))
```

```
# Codes to generate fancier scatterplot matrix
solowmatrix <- cbind(mydata$DY,mydata$LGDP60,mydata$LNGD,mydata$LINV)
splom( ~ solowmatrix, varnames = c("Per Capita\n\Real GDP\n\Growth","Log
of\n\Real Per\n\GDP in 1960","Log of\n\(n+g+d)","Log\n\Investment\n\Share"),
panel = function(x, y)
{
i <- c(2,3,5,6,8,9,12,14:24,27:33,35,37,38,40,41,42,46,47,49,58,97)
j <- c(1,4,7,10,11,13,25,26,34,36,39,43:45,48,50:57,59:96,98:121)
panel.splom(x[i],y[i], pch = 16, cex = 0.7, col = 1)
panel.splom(x[j],y[j], pch = 16, cex = 0.4, col = 2)
},sub=list("Points In Black: Angola, Benin, Burkina Faso, CAR, Chad,
Ethiopia, Gambia, Ghana, Ivory Coast, Kenya, Lesotho, Liberia, Madagascar,
Malawi, Mali, Mauritania,
Mozambique, Niger, Nigeria, Rwanda, Senegal, Sierra Leone, Somalia, Sudan,
Tanzania, Togo, Uganda, Zaire, Zambia, Bangladesh, Burma, India, Nepal,
Haiti",cex=0.5))

# Codes to generate Spread-Location Plot for residuals against fitted values
# to visualize heteroskedasticity using the lattice package
xyplot(sqrt(abs(residuals(myaugreg))) ~ fitted.values(myaugreg), panel =
function(x, y)
{panel.xyplot(x, y, pch = 16, cex = 0.7, col = 1)
panel.loess(x, y, span = 1, degree = 2, family = 'symmetric')
}, aspect = 1,xlab="Fitted Values of Regression",ylab="Residuals of
Regression")

# Codes to install quantile regressions package and estimate the median
# 25th and 75th quantile regressions
myqreg <- rq(DY~LGDP60+LNGD+LINV,data=mydata)
summary.rq(myqreg)
myaugqreg <- rq(DY~LGDP60+LNGD+LINV+LSCH,data=mydata)
summary.rq(myaugqreg)
myaugq25reg <- rq(DY~LGDP60+LNGD+LINV+LSCH,tau=0.25,data=mydata)
summary.rq(myaugq25reg)
myaugq75reg <- rq(DY~LGDP60+LNGD+LINV+LSCH,tau=0.75,data=mydata)
summary.rq(myaugq75reg)

# Codes for non-parametric bootstrapping of regression coefficients
mydata <- read.table("C:/temple.csv",header=T,sep=",")
mod1 <- lm(DY~LGDP60+LNGD+LINV,data=mydata)
mod1coefs <- coef(mod1)
fit <- fitted(mod1)
e <- residuals(mod1)
X <- model.matrix(mod1)
mod2 <- NULL
for (i in 1:1000)
{
       s <- sample(length(X[,1]),replace=T)
       y <- fit + e[s]
mod2 <- rbind(mod2, lm(y[s]~-1+X[s,])$coef)
}
cov(mod2)
se <- sqrt(diag(cov(mod2)))

OLS <- c(mod1coefs[1],mod1coefs[2],mod1coefs[3],mod1coefs[4])
BootStrap <- c(mean(mod2[,1]),mean(mod2[,2]),mean(mod2[,3]), mean(mod2[,4]))
Bias <- c(mean(mod2[,1])-mod1coefs[1],mean(mod2[,2])-mod1coefs[2],
   mean(mod2[,3])-mod1coefs[3],mean(mod2[,4])-mod1coefs[4])
table <- cbind(OLS,BootStrap,Bias)
coefsum <- coef(summary(mod1))
OLSSE <- c(coefsum[1,2],coefsum[2,2],coefsum[3,2],coefsum[4,2])
tableSE <- data.frame(OLSSE,BootSE=se,row.names=c("Intercept","LGDP60",
"LINV","LSCH"))
```

```
# Codes for parametric bootstrapping of regression coefficients to
# demonstrate the efficiency of the median regression relative to OLS when
# the errors are drawn from a t-distribution with three degrees of freedom
Rw <- 0.04287973
sigmaw <- 0.1245768
sample <- 250
n.trial <- 10000
alphaols <- rep(0,n.trial)
betaols <- rep(0,n.trial)
alphalad <- rep(0,n.trial)
betalad <- rep(0,n.trial)
set.seed(200)
a <- 0
b <- 2
for (trial in 1:n.trial) {
       trial.mktret <- Rw+sigmaw*rnorm(sample,mean=0,sd=1)
       trial.res <- rt(sample,df=3)
       trial.indret <- (a + b*trial.mktret + trial.res)
       trial.ols <- lm(trial.indret~trial.mktret)
       tmpols <- coef(trial.ols)
       alphaols[trial] <- tmpols[1]
       betaols[trial] <- tmpols[2]
       trial.lad <- rq(trial.indret~trial.mktret)
       tmplad <- coef(trial.lad)
       alphalad[trial] <- tmplad[1]
       betalad[trial] <- tmplad[2]
}

# Codes to generate table of average coefficients for OLS and LAD
c("mean ols alpha"=mean(alphaols),"mean lad alpha"=mean(alphalad),"mean ols
beta"=mean(betaols),"mean lad beta"=mean(betalad))

# Codes to generate table of coefficient bias for OLS and LAD
c("ols alpha bias"=(mean(alphaols)-a),"lad alpha bias"=(mean(alphalad)-
a),"ols beta bias"=(mean(betaols)-b),"lad beta bias"=(mean(betalad)-b))

# Codes to generate table of efficiency of OLS and LAD
c("stdev ols alpha"=sd(alphaols),"stdev lad alpha"=sd(alphalad),"stdev ols
beta"=sd(betaols),"stdev lad beta"=sd(betalad))

# Codes to merge data
myrandom <- data.frame("countryname"=mydata$country,"Random"=rnorm(121,0,1))
mynewdata <- merge(mydata,myrandom,by.x="country",by.y="countryname")

# Codes to load panel data and estimate fixed effects, and random effects
# regressions
mypanel <- read.table("C:/grunfeld.csv",header=T,sep=",")
xtfereg <- lm(I~factor(FIRM)+F+K,data=mypanel)
bfe <- coef(xtfereg)[11:12]
Vfe <- vcov(xtfereg)[11:12,11:12]
xtrereg <- lme(I~F+K,data=mypanel,random=~1| FIRM)

# Codes to apply the Hausman Specification test
bre <- fixef(xtrereg)[-1]
Vre <- summary(xtrereg)$varFix[-1,-1]          # or you could type
Vre <- vcov(xtrereg)[-1,-1]
bdifft <- bre-bfe
bdiff <- t(bre-bfe)
Vdiff <- (Vfe-Vre)
m <- bdiff%*%solve(Vdiff)%*%bdifft
pvalue <- 1 - pchisq(m,df=2)
Hausmantable <- c("Hausman"=m,"P-Value"=pvalue)
```

```
# Codes to estimate Seemingly Unrelated
mypanel <- read.table("C:/grunfeld.csv",header=T,sep=",")
model <- I ~ F + K
SUR <-
systemfitClassic("SUR",model,"FIRM","YEAR",data=mypanel,rcovformula=0)
summary(SUR)

# Codes to estimate logit
logitreg <- glm(AFRICA~LGDP60+LNGD+LINV+LSCH,data=mydata,
family=binomial(link="logit"))
logitsum <- summary(logitreg)
oddratios <- exp(coef(logitreg)[-1])
prob <- logitreg$fitted.values

# Codes to estimate ordered logit
mydata <- read.table("C:/temple.csv",header=T,sep=",")
gradedata <- read.table("C:/templegr.csv",header=T,sep=",")
mynewgradedata <- merge(mydata,gradedata,by.x="country", by.y="country")
mynewgradedata$grade <-
ordered(mynewgradedata$grade,levels=c("D","C","B","A"))
ologitreg <- polr(factor(graden)~LGDP60+LNGD+LSCH,data=
mynewgradedata,method="logistic")
summary(ologitreg)
ologitoddratios <- exp(coef(ologitreg))

# Unit Root Tests:  It looks Messy, but it's not too involved
johansen <- read.table("C:/johansen.csv",header=T,sep=",")
lrm1 <- ts(johansen$lrm1, start=c(1958,2),end=c(1984,3),frequency=4)
lrm1df.ct <- ur.df(lrm1, type = "trend", lags = 6,selectlags = "BIC")
summary(lrm1df.ct)
lrm1df.c <- ur.df(lrm1, type = "drift", lags = 6,selectlags = "BIC")
cbind(t(lrm1df.c@teststat),lrm1df.c@cval)
lrm1df.nc <- ur.df(lrm1, type = "none", lags = 6,selectlags = "BIC")
cbind(t(lrm1df.nc@teststat),lrm1df.nc@cval)
gm <- diff(lrm1)
gm.ct <- ur.df(gm,type = "trend", lags = 6,selectlags = "BIC")
cbind(t(gm.ct@teststat),gm.ct@cval)
gm.c <- ur.df(gm,type = "drift", lags = 6,selectlags = "BIC")
cbind(t(gm.c@teststat),gm.c@cval)
gm.nc <- ur.df(gm,type = "none", lags = 6,selectlags = "BIC")
cbind(t(gm.nc@teststat),gm.nc@cval)

lny <- ts(johansen$lny, start=c(1958,2),end=c(1984,3),frequency=4)
lnydf.ct <- ur.df(lny,type = "trend", lags = 6,selectlags = "BIC")
cbind(t(lnydf.ct@teststat),lnydf.ct@cval)
lnydf.c <- ur.df(lny,type = "drift", lags = 6,selectlags = "BIC")
cbind(t(lnydf.c@teststat),lnydf.c@cval)
lnydf.nc <- ur.df(lny,type = "none", lags = 6,selectlags = "BIC")
cbind(t(lnydf.nc@teststat),lnydf.nc@cval)
gy <- diff(lny)
gy.ct <- ur.df(gy,type = "trend", lags = 6,selectlags = "BIC")
cbind(t(gy.ct@teststat),gy.ct@cval)
gy.c <- ur.df(gy,type = "drift", lags = 6,selectlags = "BIC")
cbind(t(gy.c@teststat), gy.c@cval)
gy.nc <- ur.df(gy,type = "none", lags = 6,selectlags = "BIC")
cbind(t(gy.nc@teststat),gy.nc@cval)

lnmr <- ts(johansen$lnmr, start=c(1958,2),end=c(1984,3),frequency=4)
lnmrdf.ct <- ur.df(lnmr,type = "trend", lags = 6,selectlags = "BIC")
cbind(t(lnmrdf.ct@teststat),lnmrdf.ct@cval)
lnmrdf.c <- ur.df(lnmr,type = "drift", lags = 6,selectlags = "BIC")
cbind(t(lnmrdf.c@teststat),lnmrdf.c@cval)
lnmrdf.nc <- ur.df(lnmr,type = "none", lags = 6,selectlags = "BIC")
cbind(t(lnmrdf.nc@teststat),lnmrdf.nc@cval)
gr <- diff(lnmr)
gr.ct <- ur.df(gr,type = "trend", lags = 6,selectlags = "BIC")
cbind(t(gr.ct@teststat),gr.ct@cval)
gr.c <- ur.df(gr,type = "drift", lags = 6,selectlags = "BIC")
```

```
cbind(t(gr.c@teststat),gr.c@cval)
gr.nc <- ur.df(gr,type = "none", lags = 6,selectlags = "BIC")
cbind(t(gr.nc@teststat),gr.nc@cval)

difp <- ts(johansen$difp, start=c(1958,2),end=c(1984,3),frequency=4)
difpdf.ct <- ur.df(difp,type = "trend", lags = 6,selectlags = "BIC")
cbind(t(difpdf.ct@teststat),difpdf.ct@cval)
difpdf.c <- ur.df(difp,type = "drift", lags = 6,selectlags = "BIC")
cbind(t(difpdf.c@teststat),difpdf.c@cval)
difpdf.nc <- ur.df(difp,type = "none", lags = 6,selectlags = "BIC")
cbind(t(difpdf.nc@teststat),difpdf.nc@cval)
diffdifp <- diff(difp)
difpdfdiff.ct <- ur.df(diffdifp,type = "trend", lags = 6,selectlags = "BIC")
cbind(t(difpdfdiff.ct@teststat),difpdfdiff.ct@cval)
difpdfdiff.c <- ur.df(diffdifp,type = "drift", lags = 6,selectlags = "BIC")
cbind(t(difpdfdiff.c@teststat),difpdfdiff.c@cval)
difpdfdiff.nc <- ur.df(diffdifp,type = "none", lags = 6,selectlags = "BIC")
cbind(t(difpdfdiff.nc@teststat),difpdfdiff.nc@cval)

# Cointegrating Rank Test
finnish <- data.frame(gm,gy,gr,diffdifp)
H1 <- ca.jo(finnish,type='trace',K=4)
cbind(H1@cval,cbind(H1@teststat))

# Create a new dataset so that all variables have the same starting date
sample <- length(johansen$date)
finnish <- data.frame(date=johansen$date[2:sample],gm,gy,gr,diffdifp)

# Create a objects in R's memory to store the forthcoming regression output
window <- 40
subsample <- sample-window
bgy <- numeric(subsample)
bgy.ci <- numeric(subsample)
bgr <- numeric(subsample)
bgr.ci <- numeric(subsample)
bddifp <- numeric(subsample)
bddifp.ci <- numeric(subsample)
bcons <- numeric(subsample)
bcons.ci <- numeric(subsample)

# Estimate rolling regressions
for (i in 1:subsample) {
       model <- lm(gm~gy+gr+diffdifp,data=finnish,subset=i:(i+window-1))
       modelsum <- summary.lm(model)
       coefs <- coefficients(model)
       bgy[i] <- coefs[2]
       bgy.ci[i] <- 1.96*modelsum$coefficients[2,2]
       bgr[i] <- coefs[3]
       bgr.ci[i] <- 1.96*modelsum$coefficients[3,2]
       bddifp[i] <- coefs[4]
       bddifp.ci[i] <- 1.96*modelsum$coefficients[4,2]
       bcons[i] <- coefs[1]
       bcons.ci[i] <- 1.96*modelsum$coefficients[1,2]
}

# Create time series objects to be plotted shortly from the regression
# coefficients and the coefficient standard errors

bgy <- ts(bgy, start=c(1968,2),end=c(1984,3),frequency=4)
bgr <- ts(bgr, start=c(1968,2),end=c(1984,3),frequency=4)
bddifp <- ts(bddifp, start=c(1968,2),end=c(1984,3),frequency=4)
bcons <- ts(bcons, start=c(1968,2),end=c(1984,3),frequency=4)
bgyplusci <- ts(bgy+bgy.ci, start=c(1968,2),end=c(1984,3),frequency=4)
bgyminusci <- ts(bgy-bgy.ci, start=c(1968,2),end=c(1984,3),frequency=4)
bgrplusci <- ts(bgr+bgr.ci, start=c(1968,2),end=c(1984,3),frequency=4)
bgrminusci <- ts(bgr-bgr.ci, start=c(1968,2),end=c(1984,3),frequency=4)
bddifpplusci<-ts(bddifp+bddifp.ci,start=c(1968,2),end=c(1984,3),frequency=4)
bddifpminusci<-ts(bddifp-ddifp.ci,start=c(1968,2),end=c(1984,3),frequency=4)
```

```
bconplusci <- ts(bcons+bcons.ci, start=c(1968,2),end=c(1984,3),frequency=4)
bconminusci <- ts(bcons-bcons.ci, start=c(1968,2),end=c(1984,3),frequency=4)

# Use the par(mfrow … command to create a multi-panel plot, in this case
# with a 2 row, 2 column layout, and then plot each of the series together
# with the standard errors

par(mfrow=c(2,2))
plot(bgy,type="n",ylim=c(0,2))
abline(0,0,lty=8)
lines(bgy)
lines(bgyplusci,lty=2)
lines(bgyminusci,lty=2)
plot(bgr,type="n",ylim=c(-0.5,1.5))
abline(0,0,lty=8)
lines(bgr)
lines(bgrplusci,lty=2)
lines(bgrminusci,lty=2)
plot(bddifp,type="n",ylim=c(-2.2,3.2))
abline(0,0,lty=8)
lines(bddifp)
lines(bddifpplusci,lty=2)
lines(bddifpminusci,lty=2)
plot(bcons,type="n",ylim=c(-0.04,0.04))
abline(0,0,lty=8)
lines(bcons)
lines(bconplusci,lty=2)
lines(bconminusci,lty=2)

# Codes to install fSeries packages to estimate ARMA-GARCH model
mydailydata <- read.table("C:/bpdret.csv",header=T,sep=",")
armagarch <- garchFit(~arma(1,1)+garch(1,1),mydailydata$bpdret)
armagarchcoef <- armagarch@fit$coef

# Codes to estimate Hurst Statistic
timemean <- mean(mydailydata$bpdret)
bpd.dm <- mydailydata$bpdret-timemean
max.bpdret <- max(cumsum(bpd.dm))
min.bpdret <- min(cumsum(bpd.dm))
sd.bpdret <- sd(bpd.dm)
RS <- (max.bpdret - min.bpdret)/sd.bpdret
H <- log(RS)/log(length(bpd.dm))
d <- H - 0.5
```

STATA translation:

```
* Codes for basic data entry and summary statistics
clear
set mem 100m
insheet using "C:\temple.csv", comma
describe
summarize dy
regress dy lgdp60 lngd linv
tabstat dy, statistics( min p25 p50 p75 p90 max ) columns(variables)

* Codes to estimate OLS regressions and replicate MRW results
regress dy lgdp60 lngd linv
drop lsch
generate lsch=ln(school/100)
regress dy lgdp60 lngd linv lsch
regress dy lgdp60 lngd linv if nsam==1
regress dy lgdp60 lngd linv if isam==1
regress dy lgdp60 lngd linv if osam==1

* Codes to estimate augmented Solow Model and to generate model residuals
reg dy lgdp60 lngd linv lsch
predict augsolres, res

* Codes to generate histogram with normal density plot of the residuals
histogram augsolres, bin(20) normal

* Codes to generate quantile-quantile normal density plot of the residuals
qnorm augsolres

* Codes to generate quantile-quantile normal density plot of the residuals
* with more detail
qnorm augsolres, ytitle("Residual Quantiles") xtitle("Normal Quantiles")
title("Normal Q-Q Plot")

* Codes to generate scatterplot matrix
drop lsch
generate LSCH=ln(school/100)
graph matrix dy lgdp60 lngd linv LSCH

* Codes to estimate median, 25th and 75th quantile regressions
qreg dy lgdp60 lngd linv
qreg dy lgdp60 lngd linv lsch
qreg dy lgdp60 lngd linv lsch, q(0.25)
qreg dy lgdp60 lngd linv lsch, q(0.75)

* Codes for non-parametric bootstrap of regression coefficients
clear
set mem 100m
insheet using "C:\temple.csv", comma
drop lsch
generate lsch=ln(school/100)
regress dy lgdp60 lngd linv lsch
* STATA 7.0 command for non-parametric bootstrap of regression coefficients
bs "regress dy lgdp60 lngd linv" "_b[_c] _b[lgdp60] _b[lngd] _b[linv]",
reps(1000)
* STATA 9.0 command for non-parametric bootstrap of regression coefficients
bootstrap _b, reps(1000) bca: regress dy lgdp60 lngd linv estat bootstrap,
all
```

```
* Codes to load panel data and estimate fixed effects, random effects
* regressions and to apply the Hausman Specification test
clear
insheet using "c:\grunfeld.csv", comma
xi: reg i f k i.firm
tab firm, gen(d)
reg i d2 d3 d4 d5 d6 d7 d8 d9 d10 f k
tsset firm year, yearly
xtreg i f k, fe
est store fixed
xtreg i f k, re
hausman fixed

* Codes to estimate Seemingly Unrelated
clear
insheet using "C:\grunfeld.csv", comma
reshape wide i f k, i(year) j(firm)
sureg (i1 f1 k1) (i2 f2 k2) (i3 f3 k3) (i4 f4 k4) (i5 f5 k5) (i6 f6 k6) (i7
f7 k7) (i8 f8 k8) (i9 f9 k9) (i10 f10 k10)
sureg (i1 f1 k1) (i2 f2 k2) (i3 f3 k3) (i4 f4 k4) (i5 f5 k5) (i6 f6 k6) (i7
f7 k7) (i8 f8 k8) (i9 f9 k9) (i10 f10 k10), dfk corr

* Codes to estimate logit
clear
set mem 100m
insheet using "C:\temple.csv", comma
drop lsch
generate lsch=ln(school/100)
logit africa lgdp60 lngd linv lsch
glm africa lgdp60 lngd linv lsch, family(binomial) link(logit)
logit africa lgdp60 lngd linv lsch, or
predict prob, pr

* Codes to estimate ordered logit and generalized ordered logit
clear
set mem 100m
insheet using "C:\templegr.csv", comma
sort country
save "C:\gradedata.dta", replace
clear
insheet using "C:\temple.csv", comma
sort country
save "C:\mydata.dta", replace
sort country
merge country using "C:\gradedata.dta"
tab _merge
drop if _merge==1
drop _merge
sort country
tab grade, gen(d)
gen gradea = d1*4
gen gradeb = d2*3
gen gradec = d3*2
gen graded = d4
egen graden = rsum(gradea gradeb gradec graded)
drop d1-graded
drop lsch
generate lsch=ln(school/100)
save "C:\mymergegradedata.dta", replace
ologit graden lgdp60 lngd lsch

gologit graden lgdp60 lngd lsch, or
```

```
* Unit Root Tests:  It looks Messy, but it's not too involved
clear
set mem 100m
insheet using "C:\johansen.csv", comma
display q(1958q2)
generate periods = _n
generate time = periods-8
format time %tq
tsset time
varsoc lrm1, maxlag(6) exog(time)
dfuller lrm1, lags(5) trend regress
dfuller lrm1, lags(4) trend regress
dfuller lrm1, lags(4) drift
dfuller lrm1, lags(4) noconstant

gen gm = lrm1-lrm1[_n-1]
varsoc gm, maxlag(6) exog(time)
dfuller mg, lags(4) trend
dfuller mg, lags(3) trend
dfuller mg, lags(3) drift
dfuller mg, lags(3) noconstant

varsoc lny, maxlag(6) exog(time)
dfuller lny, lags(5) trend
dfuller lny, lags(5) drift
dfuller lny, lags(5) noconstant

gen gy = lny-lny[_n-1]
varsoc gy, maxlag(6)  exog(time)
dfuller gy, lags(4) trend
dfuller gy, lags(4) drift
dfuller gy, lags(4) noconstant

varsoc lnmr, maxlag(6) exog(time)
dfuller lnmr, lags(1) trend
dfuller lnmr, lags(1) drift
dfuller lnmr, lags(1) noconstant

gen gr = lnmr-lnmr[_n-1]
varsoc gr, maxlag(6) exog(time)
dfuller gr, lags(5) trend
dfuller gr, lags(5) drift
dfuller gr, lags(5) noconstant

varsoc difp, maxlag(6) exog(time)
dfuller difp, lags(4) trend
dfuller difp, lags(4) drift
dfuller difp, lags(4) noconstant

gen diffdifp = difp-difp[_n-1]
varsoc diffdifp, maxlag(6) exog(time)
dfuller diffdifp, lags(4) trend
dfuller diffdifp, lags(4) drift
dfuller diffdifp, lags(4) noconstant

* Cointegrating Relationship Rank Test
varsoc gm gy gr diffdifp, maxlag(6) exog(time)
vecrank gm gy gr diffdifp, lags(4) levela
```

```
* Codes to generate data series for rolling regressions
clear
set mem 100m
insheet using "C:\johansen.csv", comma
generate periods = _n
generate time = periods-8
format time %tq
tsset time
drop if periods==1
gen gm = lrm1-lrm1[_n-1]
gen gy = lny-lny[_n-1]
gen gr = lnmr-lnmr[_n-1]
gen diffdifp = difp-difp[_n-1]

* Codes to estimate rolling regressions and generate two-standard error
* bands
rolling _b _se, window(40) start(1958q3) end(1984q3) clear: reg gm gy gr
diffdifp
gen gyseplus = _b_gy+1.96*_se_gy
gen gyseminus = _b_gy-1.96*_se_gy

* Codes to plot rolling regression and two-standard error bands for money
* against GDP
twoway (line _b_gy end) (line gyseplus end, lpattern(dot)) (line gyseminus
end, lpattern(dot)), ytitle(GDP Growth) xtitle("") legend(order(1
"Coefficient" 2 "+S.E.'s" 3 "-S.E.'s"))
graph save "C:\GDPGrowth.gph", replace

* Codes to plot rolling regression and two-standard error bands for money
* against money rates
gen grseplus = _b_gr+1.96*_se_gr
gen grseminus = _b_gr-1.96*_se_gr
twoway (line _b_gr end) (line grseplus end, lpattern(dot)) (line grseminus
end, lpattern(dot)), ytitle(Interest Rate Changes) xtitle("") legend(order(1
"Coefficient" 2 "+S.E.'s" 3 "-S.E.'s"))
graph save "C:\InterestRate.gph", replace

* Codes to plot rolling regression and two-standard error bands for money
* against inflationary acceleration
gen diffdifpseplus = _b_diffdifp+1.96*_se_diffdifp
gen diffdifpseminus = _b_diffdifp-1.96*_se_diffdifp
twoway (line _b_diffdifp end) (line diffdifpseplus end, lpattern(dot)) (line
diffdifpseminus end, lpattern(dot)), ytitle(Inflation Acceleration)
xtitle("") legend(order(1 "Coefficient" 2 "+S.E.'s" 3 "-S.E.'s"))
graph save "C:\Inflation.gph", replace

* Codes to plot rolling regression intercept and two-standard error bands
gen conseplus = _b_cons+1.96*_se_cons
gen conseminus = _b_cons-1.96*_se_cons
twoway (line _b_cons end) (line conseplus end, lpattern(dot)) (line
conseminus end, lpattern(dot)), ytitle(Constant (Avg. Money Growth))
xtitle("") legend(order(1 "Coefficient" 2 "+S.E.'s" 3 "-S.E.'s"))
graph save "C:\Constant.gph", replace

* Codes to combine plots of rolling regression coefficients
graph combine "C:\GDPGrowth.gph" "C:\InterestRate.gph" "C:\Inflation.gph"
"C:\Constant.gph"

* Codes to estimate ARMA-GARCH model
clear
set mem 100m
insheet using "C:\bpdret.csv", comma
tsset obs, daily
arch bpdret, ar(1) ma(1) arch(1) garch(1)
```

```
* Codes to estimate Hurst Statistic
clear
set mem 100m
insheet using "C:\bpdret.csv", comma
tsset obs, daily
egen timemean = mean(bpdret)
gen bpddm = bpdret-timemean
gen cumsumbpddm=sum(bpddm)
egen maxbpdret = max(cumsumbpddm)
egen minbpdret = min(cumsumbpddm)
egen sdbpdret = sd(bpddm)
gen RS = (maxbpdret - minbpdret)/sdbpdret
gen sample = _N
gen H = log(RS)/log(sample)
gen d = H - 0.5
sum H d
```