Econometrics - Advanced Methods (for M.Sc.)

Helmut Farbmacher

Department of Economics
University of Munich

Winter Semester 2018/2019

Introduction to Mata

- The following slides are mainly based on the tutorial "An Introduction to Mata" by the University of Wisconsin's Social Science Computing Cooperative (SSCC).
- Link: `https://ssc.wisc.edu/sscc/pubs/4-26.htm`
- For more information use the documentation by typing `help mata` within Stata.

Introduction to Mata

- Mata is matrix language built into Stata, similar in many ways to R, Matlab or GAUSS.
- Mata is not a replacement for Stata, nor is it indented to be a stand-alone statistical package.

Mata Basics

- To **start** Mata, first start Stata and then type

  ```
  mata
  ```

  Stata will then switch to "Mata mode".

- Where you normally see a period you'll now see a colon.

- To **get out** of Mata mode, type

  ```
  end
  ```

Mata Basics

- While Stata is organized around commands, Mata is organized around **statements**. For example, you can simply type

  2+2

  and Mata will return

  4

Mata Basics

- Storing results in a variable is just as easy:

  ```
  x=2+2
  ```

- Note that there was no output this time. If you want to see the value of x, simply type:

  ```
  x
  ```

  and you'll get 4 back.

Matrix Operators

- The comma is defined as the **column join** operator.
- That means

  ```
  1,2
  ```

  is interpeted as a matrix with one row and two columns. The output will be:

```
      1   2
    +---------+
  1 |  1   2  |
    +---------+
```

Matrix Operators

- The backslash (\ not /, which is division) is the **row join** operator. Thus

  ```
  1\2
  ```

  is a matrix with two rows and one column:

```
     1
  +-----+
1 |  1  |
2 |  2  |
  +-----+
```

Matrix Operators

- The `..` operator creates a series starting from the number on the left up to the number on the right and makes them into a row vector.

  ```
  1..3
  ```

- The `::` operator does the same but puts them into a column vector.

  ```
  1::3
  ```

Matrix Operators

- The standard **arithmetic** operators recognize when one or both of their arguments is a matrix, and act accordingly.

```
x=(1,2)\(3,4)
y=(1,2)\(3,4)
x*y
```

- However, there are also "colon" operators which work element by element.

```
x:*y
```

Matrix Operators

- **Logical** operators such as greater than, less than, and equal to are also defined in both matrix and colon versions.
- Note that the **equals** operator is ==, as opposed to = for assignment.
- The **transpose** operator is the right single quote (').
- It works on just one object, the one to its left, and returns its transpose.

```
x'
```

Matrix Operators

- **Subscripts**: To reference a particular element of a matrix, put the row and column number in square brackets after the matrix name.

```
z=x[1,1]
z
x[1,1]=5
x
```

- "Missing" for the row or column number means all the rows or columns.

```
x[.,1]
```

Matrix Operators

- **Range subscripts** have two elements, separated by a backslash: The row and column of
  - the upper left corner of the desired range
  - the lower right corner of the desired range

- Example for selecting rows 4-7 and columns 3-8:

```
x[|4,3\7,8|]
```

- Example for selecting columns 3 to 5 with all rows:

```
x[|1,3\.,5|]
```

Getting your Data from Stata to Mata

- Within Stata:
  `putmata` exports the contents of Stata variables to Mata vectors and matrices.

- Within Mata:
  See next slides.

Getting your Data from Stata to Mata

st_data

- To make copies of your data in Mata from Stata.
- Takes two arguments: observation number and variable number.
- The row number can be missing, in which case all observations are returned.
- The column number can also be missing (all variables) or a vector listing the variables desired.

Getting your Data from Stata to Mata

- `st_data` can also take a third argument: a selection variable
- If it is specified then only observations where that variable is not equal to zero are returned.

```
st_data(.,("price","rep78"), "foreign")
```

Getting your Data from Mata to Stata

- Within Stata:
  `getmata` imports the contents of Mata vectors and matrices to Stata variables.

- Within Mata:
  `st_addvar, st_store` rather complicated way; better to use `getmata` in Stata.

Matrix Functions

- The `I` function will return an **identity matrix** of size equal to the argument it was given.

  ```
  I(3)
  ```

- The `J` function creates a **matrix of constants**. It takes three arguments: the number of rows, the number of columns, and what to put in the matrix.

  ```
  J(3,3,0)
  J(2,3,"a")
  ```

Matrix Functions

- The `e` function creates **unit vectors**: row vectors with a one in one column and zeroes everywhere else. It takes two arguments: the location of the one, and the size of the row vector.

  ```
  e(1,3)
  ```

- The `runiform` function returns a matrix filled with **random** numbers distributed **uniform(0,1)**. The size is specified in the same way as with the `J` function.

  ```
  runiform(2,3)
  ```

Matrix Functions

- The `rnormal` function returns a matrix filled with **Gaussian normal random** numbers. It takes four arguments: the number of rows, the number of columns, the mean, and the standard deviation.

  ```
  rnormal(2,3,0,1)
  ```

Matrix Functions

- **Sizes of Matrices:** `rows(x)`, `cols(x)` and `length(x)` return the number of rows, the number of columns, and the total number of elements in that matrix (rows · columns), respectively.

- Also helpful is `mata describe [x]`, which gives further information about the defined matrices apart from the size.

Matrix Functions

- `diagonal(x)` returns the principal diagonal of a matrix as a row vector (can be used to obtain standard errors).
- `trace(x)` returns the sum of the diagonal elements.
- `invsym(x)` inverts symmetric matrices.

Descriptive statistics

- `sum(x)` adds up the whole matrix, returning a single number
- `rowsum(x)` adds up each row, returning a column vector.
- `colsum(x)` adds up each column, returning a row vector.
- `max(x)` and `min(x)` simply return the largest or smallest element of the matrix.

Descriptive statistics

- `mean(x)` returns a row vector containing the column means of the matrix.
- If you need row means, you'll need to construct them by yourself:

  `rowsum(x)/cols(x)`

- `variance(x)` returns the variance matrix of your matrix and `correlation(x)` returns the correlation matrix.

Loops

- `for` loops have three components: an initialization step, a rule for when the loop should end, and some sort of progression towards that ending.

- Example:

```
for(x=0; x<=5; x++) {
x
}
```

- Note that `x++` is shorthand for `x=x+1`.

Loops

- A common use of `for` loops is to loop over the elements of a matrix:

```
m=J(5,5,0)
for(i=1; i<=rows(m); i++) {
for(j=1; j<=cols(m); j++) {
m[i,j]=i+(j-1)*cols(m)
}
}
m
```

Loops

- `while` loops are an alternative.
- The difference is that the statements will be executed over and over as long as the condition is true.
- Thus your first concern should be to make sure that at some point the condition will become false, or the loop will run forever
- Example:

```
x=0
while (x<=5) {
x
x++
}
```

Example: Linear Regression

- Use the automobile data set (`. sysuse auto`).
- Regress `price` on `mpg`, `rep78`, `foreign` and `weight` in Stata.
- One complication is that `rep78` is missing for some observations.
- Stata programs take care of this. Mata does not.
- Create a new variable `touse` which is one if all the variables are non-missing and zero otherwise:

```
gen touse=(price!=. & mpg!=. & rep78!=. & weight!=. & foreign!=.)
```

Example: Linear Regression

- Now go into Mata by typing `mata`.
- Get the data from Stata to Mata and use `touse` as the selection variable

```
x=st_data(.,("mpg","rep78","weight","foreign"), "touse")
y=st_data(.,("price"), "touse")
```

- To include a constant term in the regression you need to add a column of ones to `x`. Use the `J` function to create it and add it to `x`:

```
x=x,J(rows(x),1,1)
```

Example: Linear Regression

- Now you can estimate $\widehat{\beta}$ as

```
b=invsym(x'x)*x'y
```

- To compute their standard errors, start by finding the residuals

```
uhat=y-x*b
```

- and use them to estimate the model variance

```
sigma_hat=(uhat'uhat)/(rows(x)-cols(x))
```

Example: Linear Regression

- Then the variance-covariance matrix of the estimates is

```
vhat=sigma_hat*invsym(x'x)
```

- The standard errors for each element of $\widehat{\beta}$ can be extracted using the `diagonal` function, along with `sqrt`:

```
se=sqrt(diagonal(vhat))
```

- The t-statistic is each element of $\widehat{\beta}$ divided by its standard error.

Example: Linear Regression

- This is an element by element division so you need to use the "colon" operator:

```
t=b:/se
```

- Finding the p-values requires the `ttail` function, which takes as arguments the degrees of freedom and the t-statistic and returns the probability. It is a single-tailed test however, so you need to multiply by two:

```
p=2*ttail(rows(x)-cols(x),abs(t))
```

Example: Linear Regression

- We can construct 95%-confidence intervals with

```
ci_lower=b:-invt(rows(x)-cols(x),0.975)*se
ci_upper=b:+invt(rows(x)-cols(x),0.975)*se
```

- Of course, in large samples we could simple use the normal approximation

```
ci_lower=b:-invnormal(0.975)*se
ci_upper=b:+invnormal(0.975)*se
```

Example: Linear Regression

- Put your results together in a readable form:

  ```
  b,se,t,p,ci_lower,ci_upper
  ```

- Now exit Mata, and check your results against

  ```
  regress price mpg rep78 weight foreign
  ```

  They should be identical.