Introduction to Stata

2017-04-26

Peter Hedström

metrika consulting

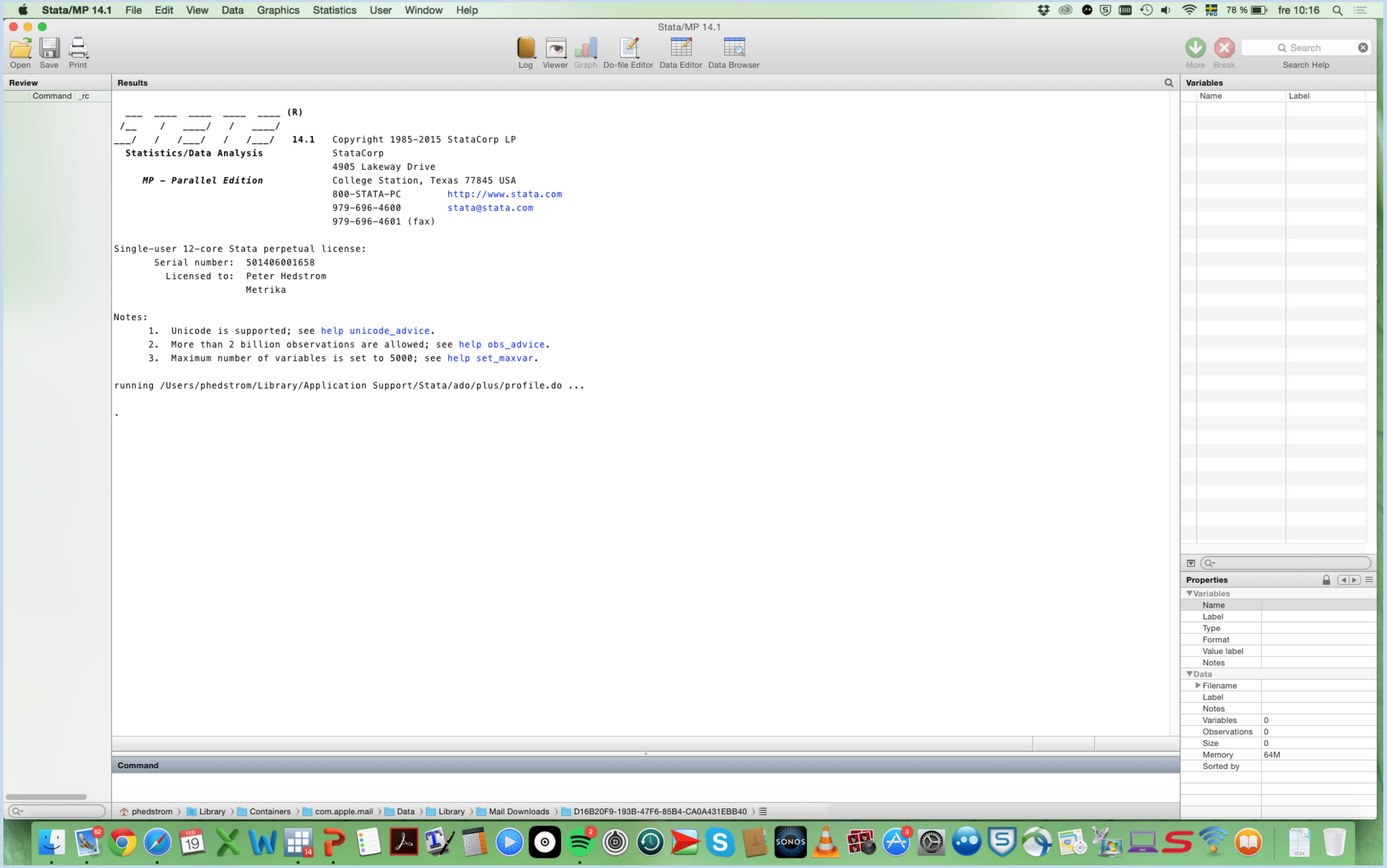
Course content

- How to start and end a Stata session
- Stata’s user interface
- Basic commands for:
  - Reading and saving data
  - Inspecting data
  - Modifying data
  - Handling data
  - Analyzing data

The most basic of the basic

- How to start a Stata session: Click on Stata icon.
- How to end a Stata session: Type **exit** or use the menu.
- How to use Stata’s help facilities: **help** and **search**.
- How to clear memory from all data: type **clear**

Stata's user interface



```
  ___  ____  ____  ____  ____ (R)
 /__    /   ____/   /   ____/
___/   /   /___/   /   /___/   14.1   Copyright 1985-2015 StataCorp LP
  Statistics/Data Analysis            StataCorp
                                      4905 Lakeway Drive
     MP - Parallel Edition            College Station, Texas 77845 USA
                                      800-STATA-PC        http://www.stata.com
                                      979-696-4600        stata@stata.com
                                      979-696-4601 (fax)

Single-user 12-core Stata perpetual license:
       Serial number:  501406001658
         Licensed to:  Peter Hedstrom
                       Metrika

Notes:
      1.  Unicode is supported; see help unicode_advice.
      2.  More than 2 billion observations are allowed; see help obs_advice.
      3.  Maximum number of variables is set to 5000; see help set_maxvar.

running /Users/phedstrom/Library/Application Support/Stata/ado/plus/profile.do ...

.
```

▼Variables	
Name	
Label	
Type	
Format	
Value label	
Notes	
▼Data	
▶Filename	
Label	
Notes	
Variables	0
Observations	0
Size	0
Memory	64M
Sorted by	

phedstrom › Library › Containers › com.apple.mail › Data › Library › Mail Downloads › D16B20F9-193B-47F6-85B4-CA0A431EBB40 ›

Stata's user interface

The user interface is divided into a set of widows:

- The **command window** is where all your commands are typed.
- The **review window** logs all commands (from the command window) as they are entered. Click on an old command, and it will appear again in the command window.
- The **variables window** lists all variables in the working file. Double-click on a variable, and it will appear in the command window.
- The **results window** displays results.
- The **properties window** describes various properties of the data set and each variable.

Stata’s user interface

In Stata you can either execute commands by using the menus or by typing the commands.

Experienced Stata users tend to type the commands because it is much faster once one knows the commands and experienced users tend to use so-called do-files which requires commands to be typed.

The menus are excellent for new users and for experienced users who need to fresh up their memory about a particular syntax. When you execute a command via the menu system the corresponding code also will be displayed.

Let us go through each of the top menu items to familiarize ourselves with them.

Stata’s user interface

There are three other important windows which can be accessed through icons in the toolbar:

- The **do-file editor**
- The **data browser**
- The **data editor**

A first brief exercise

- Start Stata
- Use “File – Open” to read into Stata the data file WVS.dta that was distributed ahead of class
- Now type these commands into the command window, one after the other:

```
summarize
tabulate v2
tab v2 v9, row
```

Data in Stata

Stata’s conceptualization of data is like a table:

- The columns are the variables and are named.
- The rows are the observations and are numbered.
- The data in the cells can be numeric or string/text.
- The missing values symbol in Stata is . and . is understood as positive infinity.

Let us take a look in the data editor.

Data types, numerical and strings

Numeric variables:

```
                                                          Closest to
Storage                                                   0 without
type                   Minimum                 Maximum    being 0     bytes
----------------------------------------------------------------------
byte                      -127                     100    +/-1           1
int                    -32,767                  32,740    +/-1           2
long            -2,147,483,647           2,147,483,620    +/-1           4
float   -1.70141173319*10^38  1.70141173319*10^38    +/-10^-38      4
double  -8.9884656743*10^307  8.9884656743*10^307    +/-10^-323     8
----------------------------------------------------------------------
```

Numbers are stored as byte, int, long, float, or double, with the default being float. byte, int, and long are said to be of integer type in that they can hold only integers.

Stata keeps data in memory, and if you have a shortage of RAM, you should record your data as parsimoniously as possible. If you have an integer variable, for example, it would be a waste to store it as a double.

The Stata command **compress** can be used for storing the data in the most economical fashion possible. You can use the **describe** command or look in the **Properties window** to see how each variable currently is stored.

Data types, numerical and strings

String or text variables:

```
String
storage        Maximum
type           length          Bytes
----------------------------------------
str1              1               1
str2              2               2
 ...              .               .
 ...              .               .
 ...              .               .
str2045         2045            2045
strL            2000000000      2000000000
----------------------------------------
```

Reading data into Stata

- Type data into Stata using the data editor.
- Copy and paste data into the data editor.
- Use the **use** command to read Stata .dta files into Stata.
- Use the **import** command to read text, Excel, and SAS files into Stata.
- Use the **Stat/Transfer** program to convert other types of files into .dta files.

Saving data

- Use the **save** command to save the data as a Stata .dta file.
- Use the **export** command to save the data into text, Excel, and SAS files as well as into text files.
- Save the data as a .dta file and use Stat/Transfer to convert the .dta file into some other format.

Basic syntax

Before learning how to inspect, modify, handle, and analyze data in Stata, it is useful to learn certain basics of Stata’s syntax.

Stata is case sensitive

- All Stata commands are lower-case.
- For example, **list** and **List** are understood differently by Stata.
- The former is a valid Stata command while the latter will give you an error message.
- Similarly, Stata is case sensitive. If you have a variable called `Income` in your dataset but try to refer to it as `income` you will get an error message.

Almost all Stata commands follow this structure:

Repeat a command for each subset of the data for which the values of the variables in the varlist are equal.

The list of variables.

Restrict the scope of the command to a specific observation range.

Additional command-specific options.

[by *varlist*:] ***command*** [*varlist*] [=*exp*] [if *exp*] [in *range*] [*weight*] [, *options*]

Specify the value to be assigned to a variable.

Restrict the scope of the command to those observations for which the value of the expression is true.

Indicate the weight to be attached to each observation.

by, **if** and **in**

- **by**, **if** and **in** are important in many different circumstances since they allow us to define the sub population for which a certain command is to be executed.
- Let us therefore go through each of these three qualifiers in some detail.

As can be seen from this slide, the **by** expression is used for separately executing a command for observations with identical values on the variables in the *varlist*:

Repeat a command for each subset of the data for which the values of the variables in the varlist are equal.

Restrict the scope of the command to a specific observation range.

The list of variables.

Additional command-specific options.

[by *varlist*:] ***command*** [*varlist*] [=*exp*] [if *exp*] [in *range*] [*weight*] [, *options*]

Specify the value to be assigned to a variable.

Indicate the weight to be attached to each observation.

Restrict the scope of the command to those observations for which the value of the expression is true.

The **by** qualifier

For example, if we have three variables called gender, education, and age, and we execute this command

by gender: tab education age

Stata will produce two cross tables between education and age, one for men and one for women.

If we instead had written

by education: tab gender age

we would have received one cross table for gender and age for each educational group.

The **by** qualifier

In order to use the **by** expression, the data first must be sorted on the variables included in the by-expression. For example:

sort v2
by v2: summarize

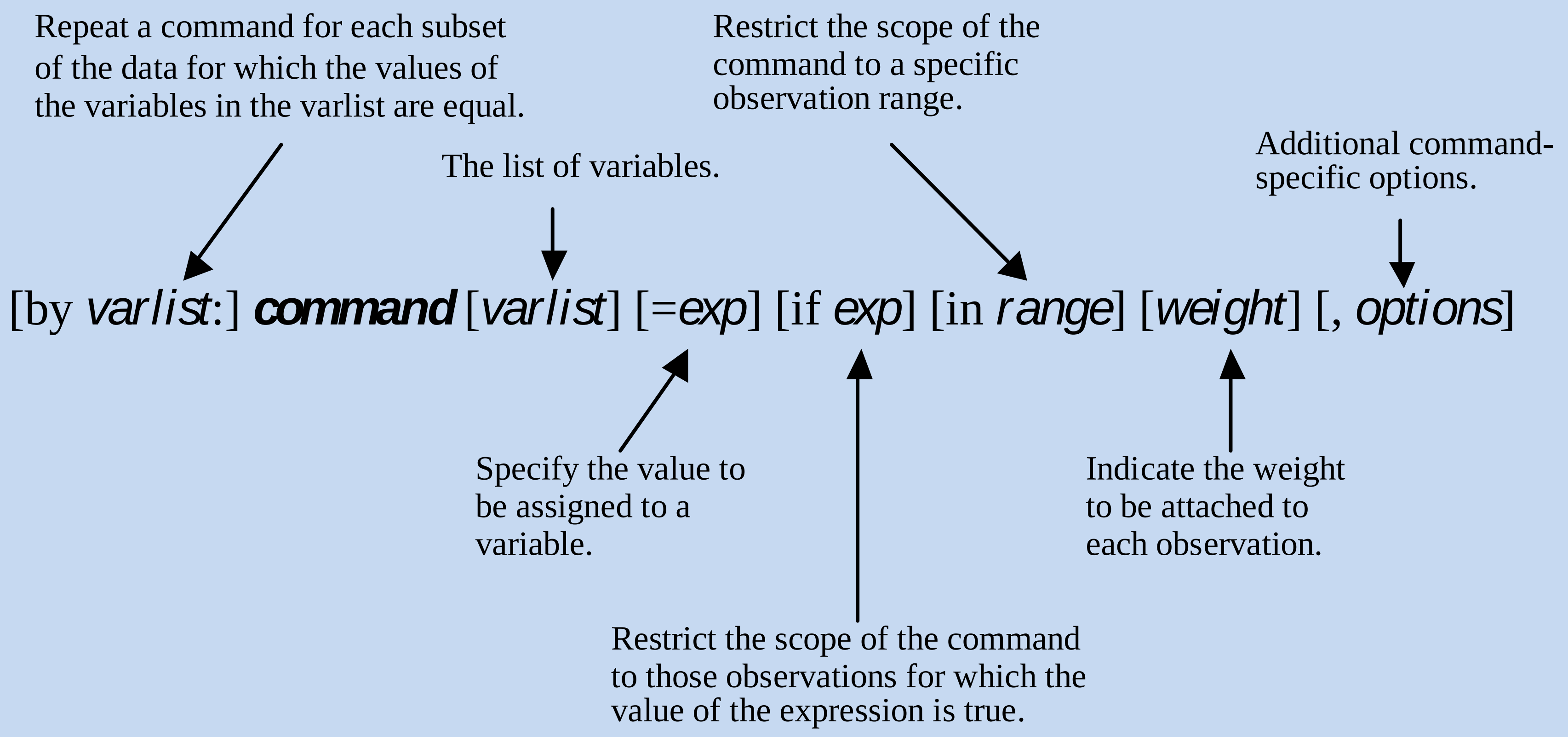
will produce summary statistics on each variable (number of observations, means, standard deviations, etc.) for groups of individuals with the same values on variable v2.

The sort and by commands can be replaced by the bysort command:

bysort v2 v10: summarize

will produce summary statistics on each variable for groups of individuals with the same unique combination of values on variables v2 and v10.

As could be seen from this slide, **if** and **in** expressions are used for restricting the observations that a command applies to:



Repeat a command for each subset of the data for which the values of the variables in the varlist are equal.

The list of variables.

Restrict the scope of the command to a specific observation range.

Additional command-specific options.

[by *varlist*:] ***command*** [*varlist*] [=*exp*] [if *exp*] [in *range*] [*weight*] [, *options*]

Specify the value to be assigned to a variable.

Restrict the scope of the command to those observations for which the value of the expression is true.

Indicate the weight to be attached to each observation.

Stata expressions

You usually use the **if** and **in** expressions in combination with these kinds of operators:

Operator	Meaning
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Identical with
!= or ~=	Different from
&	And
|	Or

The **in** qualifier

The **in** expression restricts the command to a subset of observations. For example:

- **list v6 in 10** will display the value on variable v6 for observation number 10.
- **list v6 in 1/10** will display the value on variable v6 for the first 10 observations.
- **list v6 in -4/l** will display the value on variable v6 for the last 4 observations.
- **list v6 in -5/-1** will display the value on variable v6 for the next to the last 4 observations

The **if** qualifier

- The **if** expression restricts the scope of a command to the subset of observations for which the value of the expression is true. For example:

- **list v6 if v2 == 1** will display the value on variable v6 for observations that have the value 1 on variable v2.

- **list v6 if v2 >= 5 & v2 < .** will display the value on variable v6 for observations with values on variable v2 that are greater than or equal to 5 and less than positive infinity, i.e., is not missing.

- **Nb!** The expression **==** means identical to and is different from the assignment operator **=**.

Exercise

- Now try this for yourself using the World Value Survey data.
- Execute the **summarize** command together with the **by** clause to examine how Swedes and Americans differ in how satisfied they are with their lives on average.
- Now do the same using the **if** clause instead of the **by** clause

Inspecting data

Once the data is read into Stata it is important to start with inspecting the variables to make sure that they look the way they should. Here are some useful commands:

- **describe**
- **codebook**
- **summarize**
- **histogram**
- **twoway**

See this web site for examples of Stata graphs:
http://www.ats.ucla.edu/STAT/stata/library/GraphExamples/default.htm

Exercise

Learning-by-doing is the best way to understand what these commands do. Try this:

```
sysuse auto, clear
describe
codebook
summarize
histogram mpg
scatter price weight
line price weight
line price weight, sort
twoway (scatter  price weight) (line price weight, sort)
twoway (scatter  price weight) (lfit  price weight)
twoway (scatter  price weight) (qfit  price weight)
twoway (scatter  price weight) (qfit  price weight) (lfit price weight)
```

Modify data

The following commands are the core commands for editing or modifying your data:

- **generate** are used for creating new variables (see also **egen** – extensions to generate).
- **replace** is used for making changes in existing variables.
- **recode** is a sometimes useful replacement for replace.
- **rename** is used for changing the name of a variable.
- **drop** is used for dropping observations or variables.
- **keep** is used for keeping certain observations or variables.

Examples of editing commands

```
generate male=v223
replace male=0 if v223==2
```

We could also have written like this

```
generate male=v223
recode male (2=0)
```

If we want to to drop all men from the data we could write like this:

```
drop if male==1
```

And if we wanted to drop the male variable, we could write like this:

```
drop male
```

Labels

Variable labels and value labels can be assigned with commands but it is easier to do using the menu system.

Variable labels:

Data – Variable Manager

Value labels:

Data – Data Utilities – Label Utilities – Manage Value Labels

Exercise

- Use auto.dta and change the variable label, value labels, and name of the variable foreign.
- Use the menu **Data – Variable Manager** to change the variable label from “Car type” to “Origin”.
- Use the menu **Data – Data Utilities – Label Utilities – Manage Value Labels** to change the value labels from “Domestic” and “Foreign” to “US made” and “Others”.
- Finally, use the **rename** command to change the name of the variable from foreign to origin.

Exercise

- Use the WVS data.
- Use the **generate** command to create a new variable called university which is equal to 1 for those who have had at least some university education (v226 is equal to 8 or 9) and zero for all the others.
- Use the **summarize** command together with appropriate **by** commands to examine how the proportion with university education varies between men and women in Sweden and the US.

Stata expressions and functions

When creating new variables or modifying existing ones we often need to use various sorts of algebraic expressions.

Such expressions are specified in a natural way using standard rules, and you may use parentheses freely to force a different order of evaluation.

For example:

generate new = myv+2/oth

is interpreted as

generate new = myv+(2/oth)

If what you wanted was (myv+2)/oth, you would have to tell that to Stata by typing

generate new = (myv+2)/oth

In addition, Stata expressions often use various functions, and Stata includes a wide array of different types of functions:

```
+-----------------------------------------------------------------+
| Type of function                      | See help                |
|---------------------------------------+-------------------------|
| Mathematical functions                | math functions          |
|                                       |                         |
| Probability distributions and         |                         |
|   density functions                   | density functions       |
|                                       |                         |
| Random-number functions               | random-number functions |
|                                       |                         |
| String functions                      | string functions        |
|                                       |                         |
| Programming functions                 | programming functions   |
|                                       |                         |
| Date and time functions               | datetime_functions      |
|                                       |                         |
| Selecting time spans                  | time-series functions   |
|                                       |                         |
| Matrix functions                      | matrix functions        |
|                                       |                         |
+-----------------------------------------------------------------+
```

Stata expressions

More examples:

```
generate agediff=abs(age-avage)
generate lninc=ln(income)
gen eqinc=faminc/sqrt(famsize)
generate age2=age*age
generate edageint=education*age
```

To easily generate dummy variables:

```
tabulate v226 if v226<99, generate(ed)
```

Exercise

- Use the WVS data

- Use the type of command described on the previous slide, i.e.,

  tabulate v226 if v226<99, generate(ed)

  to create a set of dummy variables indicating how important religion is in their lives (v9)

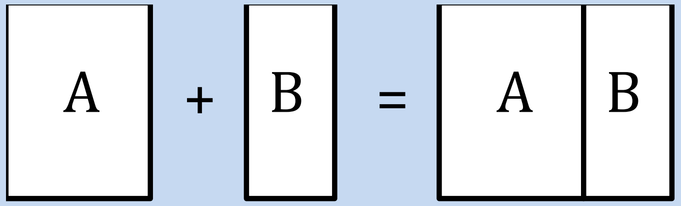
- Inspect the new variables in some way you deem appropriate to make sure that the dummy variables were correctly created.

Data management

The following commands are crucial for handling your data and getting it in a format suitable for analysis:

- **merge**
- **append**
- **reshape**

Use the **merge** command to add variables from another dataset



One-to-one merge on specified key variables:

merge 1:1 varlist using filename

Many-to-one merge on specified key variables:

merge m:1 varlist using filename

Nb! Both datasets must be sorted on varlist

Exercise (1)

Use the data editor to create a data set that looks like this (pid= personal ID, sid= school ID):

```
+-----------+
| pid   sid |
|-----------|
|   1     1 |
|   2     1 |
|   3     2 |
|   4     2 |
|   5     3 |
+-----------+
```

Sort the data on pid and save it as a.dta. Now create a new data set that looks like this, sort it on pid, and save it as b.dta (GPAp = the person's grade point average) :

```
+----------------+
| pid    GPAp |
|----------------|
|   1        9    |
|   2       12    |
|   3       11    |
|   4       24    |
|   5       13    |
+----------------+
```

Finally, create a school data set called c.dta with one observation per school and two variables, sid and GPAs (= average GPA in the school). School 1 should have the value 14 on GPAs, School 2 the value 10, and School 3 the value 15. Sort the data on sid before saving.

Exercise (2)

With these three toy data sets we can see how the one-to-one and many-to-one merges work.

Try this:

```
use a
merge 1:1 pid using b
list
```

Now try this:

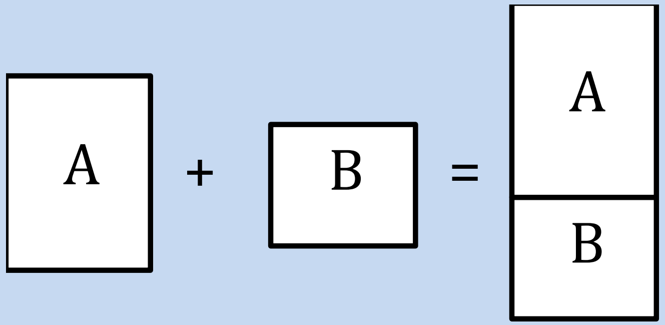
```
sort sid
merge m:1 sid using c
list
```

Use the **append** command to add observations from another dataset.

The command

append using filename

appends Stata-format datasets stored on disk to the end of the dataset in memory.



Exercise

Now assume that we have received more data on other individuals attending other schools. Use the data editor to create a data set that looks like this (pid= personal ID, sid= school ID):

```
 +-----------+
 | pid   sid |
 |-----------|
 |   1    4  |
 |   2    4  |
 |   3    5  |
 |   4    5  |
 |   5    6  |
+-----------+
```

Save it as d.dta.

Now try this:

append using a

list

Observe that pid no longer uniquely identifies the individuals. Create a new variable called ID that is unique for each individual and drop the old pid-variable from the dataset.

Real world exercise

- Use the WVS data
- Save the Swedish and the US observations in two different data sets. Call them sweden.dta and usa.dta.
- Now use the append command to reunite the two datasets into one again.

The **reshape** command is extremely useful when working with panel data

Data in long format: **Data in wide format:**

```
+--------------------+          +-------------------------+
| ID  Year  Income |          | ID  IncomeY1 IncomeY2 |
+--------------------+          |-------------------------|
| 1     1      4.1  |          | 1       4.1       4.5   |
| 1     2      4.5  |  <-->    | 2       3.3       3.0   |
| 2     1      3.3  |          +-------------------------+
| 2     2      3.0  |
+--------------------+
```

For panel data analyses where we observe the individuals at several different points in time we usually want the data in long format but many datasets are in the wide format.

The **reshape** command is extremely useful when working with panel data

Data in long format:

```
+------------+
| i  j  stub |
|------------|
| 1  1   4.1 |
| 1  2   4.5 |
| 2  1   3.3 |
| 2  2   3.0 |
+------------+
```

```
  reshape
<---------->
```

Data in wide format:

```
+-----------------+
| i  stub1 stub2 |
|-----------------|
| 1    4.1   4.5 |
| 2    3.3   3.0 |
+-----------------+
```

For panel data analyses where we observe the individuals at several different points in time we usually want the data in long format but many datasets are in the wide format. i = individual, j = time, and “stub” is part of the variable name (e.g. income).

Syntax of the **reshape** command

Convert data from wide form to long form:

reshape long stubname, i(varname) j(varname)

Convert data from long form to wide form:

reshape wide stubname, i(varname) j(varname)

i(varname)	varname is the ID variable
j(varname)	long->wide: varname, existing variable
	wide->long: varname, new variable

Exercise

Use the data editor to create a dataset that looks like this:

id	sex	inc10	inc11	inc12
1	0	5000	5500	6000
2	1	2000	2200	3300
3	0	3000	2000	1000
4	1	7000	7500	8000

How do we change it from its current wide format into long format with the reshape command?

Exercise

Answer:

reshape long inc, i(id) j(year)

And if we want it back in wide format:

reshape wide inc, i(id) j(year)

Data analys

Stata has a huge number of commands useful for analyzing your data (see the sum menus under Statistics).

These are some of the most commonly used commands:

- **tabulate**
- **correlate**
- **regress**
- **logit/logistic**

Tabulate and correlate variables in Stata

Try this:

```
sysuse auto, clear
correlate price-foreign
tabulate foreign
tabulate foreign, sum(price)
generate expensive=0
replace expensive=1 if price>6340 &price<.
tabulate foreign expensive
tabulate foreign expensive, row
tabulate foreign expensive, column
tabulate foreign expensive, cell
tabulate foreign expensive, chi2
```

Regression analysis in Stata

- Model: $Y_i = a + b_1X1_i + b_2X2_i + b_3X3_i + e_i$
- **regress y x1 x2** regresses y on x1 and x2.
- **predict yhat** generates a new variable, here called yhat, equal to the predicted values from the regression.
- **predict e, resid** generates a new variable, here called e, equal to the residuals from the regression.

Regression examples

Try this:

```
sysuse auto, clear
regress price length
regress price weight
twoway (scatter price  length) (lfit price length)
twoway (scatter price  weight) (lfit price weight)
twoway (scatter price  weight) (qfit price weight)
twoway (scatter price length) (qfit price length)
```

Regression analysis in Stata

- **test x1** tests the null hypothesis that the coefficient for x1 is equal to zero.
- **test x1=x2** tests the null hypothesis that the coefficient for x1 and x2 are the same.
- **test x1 x2** tests the null hypothesis that the coefficient for x1 and x2 both are equal to zero.

Regression example

After having estimated the model with

```
sysuse auto, clear
regress price weight length
```

Perform the following tests:

```
test weight
test weight=3
test weight=length
test weight length
```

Exercise

- Use the WVS data.
- Use income (v236) as the dependent variable and pretend that it is a proper continuous variable.
- Use the following variables as independent variables: sex (v223), age (v225), education (v226).
- Recode the variables into dummy variables if you think it is needed.
- Estimate the regression model separately for Swedes and Americans using the regress command combined with **by** or **if**.

Logistic regression analysis in Stata

- Ordinary regression analysis assumes that the dependent variable is continuous.

- Many outcomes analyzed by social scientists and public health researchers tend to be discrete rather than continuous, however. We analyze how various factors influence individuals' choices or various events taking place in the lives of individuals, for example.

- In such situations, logistic regression analysis is a useful alternative to ordinary regression analysis, and conceptually logistic regression analysis is very similar to ordinary linear regression analysis.

- Model: $\ln(p_i/1-p_i) = a + b_1X1_i + b_2X2_i + b_3X3_i$

Logistic regression in Stata

Two different commands in Stata can be used for estimating these models:

logit yvar xvar

logistic yvar xvar

The first reports coefficients as default while the second reports odds-ratios as default.

But since both commands can display the results in either way, the commands are very similar.

Personally I prefer **logit** because I want to start with examining the regression coefficients, but that is just a personal habit and taste.

Exercise

Try the following:

```
webuse nhanes2f
logit highbp height weight age female
logit, or
```

Test if the size of the age and the weight coefficients are significantly different from one another.

Margins

- The **margins** command and the associate **marginsplot** command are extremely useful for understanding and communicating the results of regression-like analyses to others.
- Results from ordinary linear regression analyses usually is not problematic in that respect, unless it contains polynomial terms or interaction effects.
- But logistic regression analyses always are difficult to interpret because normally we are not interested in knowing how the logit changes with unit-changes in the independent variables but in the *probabilities*, and the probabilities are not additively and linearly related to the independent variables.

Margins

- We use the **margins** command for arriving at "adjusted predictions" of a model and to examine "marginal effects".
- In order to use the **margins** command we have to learn about so-called *factor variable notation*.
- Assume that we have four variables y, X1, x2 and x3.
- y is the dependent variable and we believe that the relationship between X1 and y is curvilinear so we want to include the square of X1, and that there is an interaction effect between x2 and x3.

Margins

- We thus want to estimate the parameters of a model that looks like this:

  $y = a + b_1*X1 + b_2*X1^2 + b_3*X2 + b_4*X3 + b_5*(X2*X3)$

- Normally we would do that as follows in Stata:

```
gen X12 = X1*X1
gen X23 = X2*X3
regress y X1 X2 X3 X12 X23
```

- But when we use factor-variable notation we do not have to create any new variables but instead we write like this:

```
regress y c.X1 c.X1#c.X1 i.X2##c.x3
```

- The i. and c. notation tells Stata which variables are categorical indicator variables and which are continuous.

Margins

In general, the notation implies the following:

- A#A: A^2
- A#A#A: A^3
- A#A#A#A: A^4
- A##B: A B AB
- A##B##C: A B C AB AC BC ABC

where A, B, and C are variable names.

Often this is a more economical way of expressing our selves, but the main reason for using these types of expressions is that Stata then knows which variables are interrelated, and this is essential for **margins**.

Exercise

- Now try this:

```
webuse nhanes2f, clear
keep if !missing(diabetes, black, female, age)
logit diabetes black female age
```

- It is easy to see what sign the effect of a variable has from this table but not the exact form of the relationship. Adjusted predictions simplifies matters:

```
margins, at(age=(20 30 40 50 60 70)) atmeans
marginsplot
```

Margins

If we include polynomials or interactions in the model, we have to use the factor variable notation.

Try this:

```
logit diabetes i.black i.female c.age c.age#c.age i.black#c.age
margins, at(age=(20 30 40 50 60 70)) atmeans
marginsplot

margins black#female, at(age=(20 30 40 50 60 70)) atmeans
marginsplot
```

Margins

One can define and measure marginal effects in many different ways.

Assume that we want to know the marginal effect of being black on the probability of having diabetes, following Williams (2012) we would calculate the average marginal effect as follows:

- Start with the first person in your dataset. Treat that person as though he or she were white, regardless of what the person's race actually is. Leave all other independent variable values as is. Compute the probability that this person (if he or she were white) would have diabetes.
- Now do the same thing but this time treating the person as though he or she were black.
- The difference in the two probabilities just computed is the marginal effect for that case.
- Repeat the process for every case in the sample.
- Compute the average of all the marginal effects you have computed. This gives you the average marginal effect (AME) for being black.

Margins

- In Stata, the margins command makes it easy to calculate the AME.

```
logit diabetes i.black i.female c.age c.age#c.age i.black#c.age
margins, dydx(black)
margins, dydx(black female)
```

- But the effect may vary with age and to examine whether that is the case, we can do like this:

```
margins, dydx(black female) at(age=(20 30 40 50 60 70))
marginsplot
```

And now it is time for you to raise questions about things that I did not explain well enough or things that I did not cover at all.