Introduction to Stata - Session 2

Siv-Elisabeth Skjelbred

ECON 3150/4150, UiO

January 26, 2016

Before we start

- Download auto.dta, auto.csv from course home page and save to your stata course folder.
- Open Stata: Either through kiosk.uio.no (using Internet Explorer) or directly from the computer
- Change your working directory to your stata course folder

Outline of this session

1. Log and do-files
2. Preparing data
3. Data types - numeric or string
4. Naming, labeling and formatting variables
5. Adding and manipulating variables

Numeric data types

As explained last time Stata stores numbers in different formats such as byte, int and float where float is the default data type and has about 7 digits of accuracy. To save memory you should store the data with as low accuracy as necessary. The command -compress- does the work for you. Using auto.dta

```
. compress
  mpg was  int now  byte
  rep78 was  int now  byte
  trunk was  int now  byte
  turn was  int now  byte
  make was  str18 now  str17
  (370 bytes saved)
```

Do-file - make

- In the result window you see the commands for what you have done in a session.
- To store your commands in a do-file mark the desired ones, right click and press "Send to do-file editor".
- Save the do-file for later use.
- You can run the entire do-file again by writing do 'do-file name'

Do-file - comments

To make it easier to understand what you have done include comments in your do-file.

- Begin the line with a * and stata ignores the full line.
- Place the comment in /* */ delimiters. Allows comments over multiple lines.
- Place the comment after two forward slashes: // . Everything after the line is considered a comment
- Place the comment after three forward slashes which allows your command to run over multiple lines with comments on each line.

Note: * is the only one that works as input in the command window, the others only work in a do-file.

Exercise

Make a do-file from this session inserting comments to explain where you think it is necessary.
All commands can be run from the do-file rather than from the comment window.

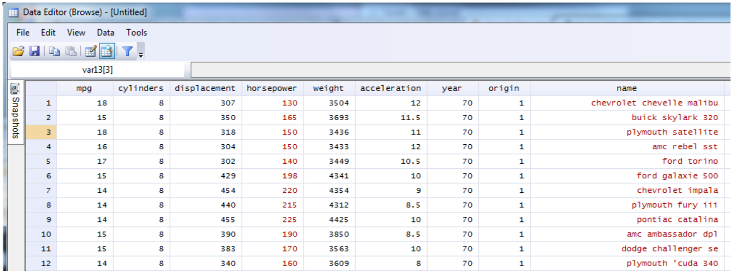
Log-file

Stata can record your work in a log-file which contains what you type and what Stata produce in response in a smcl-file. smcl files needs to be opened in the viewer-window.

- The command -log using *filename*- specifies that (from then on) what you type and what Stata produces in response is stored in a log file.
- -, replace- tells Stata to replace the existing file if a file already exists with the same name.
- -, append- tells Stata to append the new log onto the existing log-file.
- -log off- tells Stata to take a break from logging.
- -log on- tells Stata to start logging again after the break.
- -log close- tells Stata to stop logging and close the log.

Preparing data

The data editor for the auto.csv file looks like this:



	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
1	18	8	307	130	3504	12	70	1	chevrolet chevelle malibu
2	15	8	350	165	3693	11.5	70	1	buick skylark 320
3	18	8	318	150	3436	11	70	1	plymouth satellite
4	16	8	304	150	3433	12	70	1	amc rebel sst
5	17	8	302	140	3449	10.5	70	1	ford torino
6	15	8	429	198	4341	10	70	1	ford galaxie 500
7	14	8	454	220	4354	9	70	1	chevrolet impala
8	14	8	440	215	4312	8.5	70	1	plymouth fury iii
9	14	8	455	225	4425	10	70	1	pontiac catalina
10	15	8	390	190	3850	8.5	70	1	amc ambassador dpl
11	15	8	383	170	3563	10	70	1	dodge challenger se
12	14	8	340	160	3609	8	70	1	plymouth 'cuda 340

- Black text means number.
- Red text means string.

Horsepower is stored as a string (text), while we know it is a number. ? indicates missing, while stata considers . to be the symbol for missing.

Strings

Strings are good for ID's, but most of the time we do not want our data as strings. We cannot do our calculations with string variables and they take a lot of memory. Quick fix:

-gen namenewvariable = real(namestringvariable) -

Problem: variables that includes comma f.ex "130,00" is registered as a missing variable.
Alternative:

```
. destring horsepower, dpcomma replace ignore("?")
horsepower: characters  ? removed; replaced as int
(5 missing values generated)
```

Part of string

A string can contain multiple parts of information.

- The variable name in auto.csv both gives make and model.
- To extract part of the string you can use the following commands:
    - word(variablename,wordnumber) - is a function that gives a specific word number from the given variable.
    - substr(varname,n1,n2) - to take the string from the n1'st letter to the n2'st letter.

Categorical string values

- Text in string value can put observations into categories: gender, car brand, country.
- The command -encode varname, gen(nameofnewvar)- preserves the information in the data as value labels.

Encoded variables

The variable foreign seems to have the values "Foreign and Domestic" however if asking for description we see that it is stored as byte.

```
. list foreign in 52/54

     +----------+
     |  foreign |
     |----------|
 52. | Domestic |
 53. |  Foreign |
 54. |  Foreign |
     +----------+

. desc foreign

                 storage   display    value
variable name     type     format     label      variable label
-----------------------------------------------------------------
foreign           byte     %8.0g      origin        Car type
```

Encoded variables

```
. sum price if foreign=="Foreign"
type mismatch
r(109);

. sum price if foreign==Foreign
Foreign not found
r(111);

. sum price if foreign==1

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       price |         22    6384.682    2621.915       3748      12990
```

Encoded variables

```
. tab foreign

   Car type │      Freq.     Percent        Cum.
────────────┼───────────────────────────────────
   Domestic │         52       70.27       70.27
    Foreign │         22       29.73      100.00
────────────┼───────────────────────────────────
      Total │         74      100.00

. tab foreign, nolabel

   Car type │      Freq.     Percent        Cum.
────────────┼───────────────────────────────────
          0 │         52       70.27       70.27
          1 │         22       29.73      100.00
────────────┼───────────────────────────────────
      Total │         74      100.00

.
```

Stata relies on formats when displaying the data:

```
. list price in 1/2

     +-------+
     | price |
     |-------|
  1. | 4,099 |
  2. | 4,749 |
     +-------+

. format price %8.2f

. list price in 1/2

     +---------+
     |   price |
     |---------|
  1. | 4099.00 |
  2. | 4749.00 |
     +---------+
```

where the .2 specifies that we want "dot" to be the comma separator and we want two decimal. You can add "c" after the f if you want to separate thousands with a comma.

Value labels

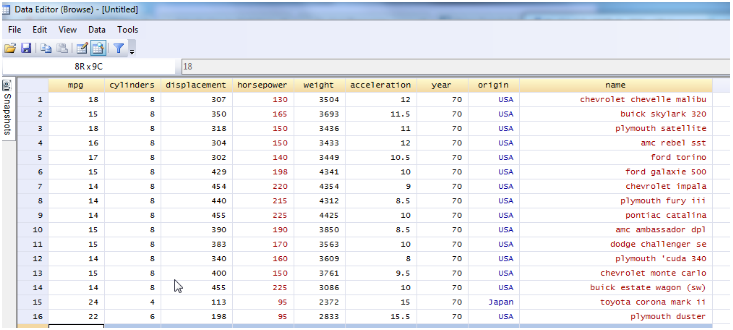
- The variable "origin" has value 1, 2 or 3.
- Information about the data set: 1=USA 2=Europa and 3=Japan.
- Can this information be included directly?

Attaching a variable (and value) label consist of two steps:

1. Define a mapping from values to labels: a value label.
2. Associate the value label (mapping) to the variable.

Example value label

Use the commands: - label define *labelname* 1 "USA" 2 "Europe" 3 "Japan" -
- label values origin *labelname* -
After:



	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
1	18	8	307	130	3504	12	70	USA	chevrolet chevelle malibu
2	15	8	350	165	3693	11.5	70	USA	buick skylark 320
3	18	8	318	150	3436	11	70	USA	plymouth satellite
4	16	8	304	150	3433	12	70	USA	amc rebel sst
5	17	8	302	140	3449	10.5	70	USA	ford torino
6	15	8	429	198	4341	10	70	USA	ford galaxie 500
7	14	8	454	220	4354	9	70	USA	chevrolet impala
8	14	8	440	215	4312	8.5	70	USA	plymouth fury iii
9	14	8	455	225	4425	10	70	USA	pontiac catalina
10	15	8	390	190	3850	8.5	70	USA	amc ambassador dpl
11	15	8	383	170	3563	10	70	USA	dodge challenger se
12	14	8	340	160	3609	8	70	USA	plymouth 'cuda 340
13	15	8	400	150	3761	9.5	70	USA	chevrolet monte carlo
14	14	8	455	225	3086	10	70	USA	buick estate wagon (sw)
15	24	4	113	95	2372	15	70	Japan	toyota corona mark ii
16	22	6	198	95	2833	15.5	70	USA	plymouth duster

Variable labels

It is also useful to document your data by attaching labels to variables.

- - label var varname "label text" -

Adding variables

Two methods that work with different set of functions:

- Simple transformation of other variables use - generate-. The values of the variable are specified by $=$ exp.
  EX: generate $price2 = price^2$
- -egen- works for functions that work across all observations. F.ex:
  - by foreign: egen maxprice $=$ max(price)
  - egen meanyear $=$ rowmean(year*)

Do it yourself

Use auto.dta:

- Generate a new variable with only the first word from the variable make. (i.e extract only the manufacturer)
- Use encode to create a new variable *manuf*
- Label the variable rep78 with 1 "Poor" 2 "Fair" 3 "average" 4 "Good" 5 "Excellent".
- Label the variable heavy with "=1 if car is heavier than 4000 lbs"
- Label the variable make with "Make of car"

Drop variables

Encode requires that you always make a new variable. You can either drop the extra variables or keep the desired ones.

- drop varname1 varname2 ...
- keep varname1 varname2 ..

Variable naming

Smart naming of your variables help you use the command line efficiently.
Choose your variables names such that you:

- Minimize typing.
    - no uppercase (Ex female rather than Female)
    - no underscore (year98 rather than year_98)
- Can effectively use wildcards
    - regyr1, regyr2

  Use variable labels to document

Functions

When generating variables you can use functions and expressions.
Mathematical functions:

- Example take the logaritm: gen lninc = ln(income)
- abs(), round(), sqrt() (for aboslute number, rouding and square root)

Random numbers:

- runiform() Return uniformly distributed random variates on the interval [0,1)
- rnormal() returns standard normal random variates (i.e with mean 0 and standard deviation 1)

Probability distribution

- normal(), ttail(), invttial() and many more

Missing variables

Note: Missing variables are stored as "." Stata deals with missing variables in different ways depending on the command:

- - generate - Stata treats a missing value as the largest possible value (e.g positive infinity) thus they are included when you use -generate heavy if weight *geq* 4000. Alternatives:
  - gen heavy=0
    replace heavy=1 if weight$\geq$4000 & weight!=.
  - generate byte heavy2 $=$ weight $\geq$ 4000 if weight $<$.
- -Summarize - use all the available data.
- - Tabulate - by default missing values are excluded and percentages are based on the number of non missing values. Can include them by adding ", missing" to the command.
- - correlate - by default correlations are computed based on the number of pairs with non-missing data.
- - regress - if any of the variables listed after the regress command are missing, the observations missing that value(s) are excluded from the analysis.

Sorting

- sort arranges the observations of the current data into ascending order based on the values of the variables in varlist.
- There is no limit to the number of variables in the varlist
- Missing numeric values are interpret as being larger than any other number.
- If you want to use by varname: command, you need to first sort by that variable.

Exercises

- Make a table of price and weight by whether the car is foreign or not.
- Define this as the label for rep78: "This is the frequency of repair record on a 1-5 scale, 1=Poor, 5=excellent".
- Generate a cross tabulation of repair and foreign status.
- Generate a cross tabulation of repair and foregin status with the cell frequency.
- Correlate mpg and weight
- Correlate mpg and weight separately by foreign status and test significance of correlation.

- tabulate the variable manuf
- Delete the variables make2 and manuf by using the command -drop-
- rename the variable manuf make by using -rename oldvarname newvarname
- Drop observations with missing information on repair record of 78 by using drop if varname $>=$. (. to stata is stored as a large number)

What you should have learned...

- Read in data in non-Stata formats
- Add and change variables (generate, replace)
- Be aware of the type of your variables
- Label your variables (label ...)
- Convert string to numeric and vice versa (destring, real(), encode)