Introduction to Stata - Session 2

Simon, Hong

ECON 3150/4150, UiO

January 15, 2018

Before we start

- Download auto.dta, auto.xlsx from course home page and move it 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. Data Types
2. Preparing Data
3. Data Types - Numeric or String
4. Naming, Labeling and Formatting Variables
5. Logical Expressions and Ranges
6. Adding and Manipulating Variables
7. Saving Modified Data
8. Sort, Tabulate, Correlate

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)
```

More on data types :Stata Data Types Documentation

Format

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.

Preparing data

Open auto.xlsx in Stata, copy paste in data editor or:

```
cd "PATH\statacourse"
import excel auto.xlsx , firstrow
```

	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.

Sometimes not everything goes smoothly. Horsepower should be numeric but is stored as a string (text). "?" indicates missing values in the original data, while stata recognize only "." to be the symbol for missing. So, it recognizes horsepower to be a string variable.

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 horsepower_num = real(horsepower)
```

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

```
destring horsepower, dpcomma replace ignore("?")
destring horsepower, dpcomma replace force
```

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.

```
gen maker_string = word(name, 1)
```

- 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.

```
encode maker_string, gen(maker)
```

Encoded variables

The new encoded variable `maker` seems to have the name of the brands as values, however if asking for description we see that it is stored as long (numeric value).

```
. list maker in 5/8
     +-----------+
     |     maker |
     |-----------|
  5. |      ford |
  6. |      ford |
  7. | chevrolet |
  8. |  plymouth |
     +-----------+
 . describe maker
              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------
maker           long    %13.0g     maker
```

Encoded variables

```
. sum mpg if maker==ford
ford not found
r(111);

. sum mpg if maker=="ford"
type mismatch
r(109);

. sum mpg if maker==2

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
         mpg |          7    26.71429    6.632855        20       36.4
```

Encoded variables

```
. tab maker in 5/8

       maker |      Freq.     Percent        Cum.
-------------+-----------------------------------
   chevrolet |          1       25.00       25.00
        ford |          2       50.00       75.00
    plymouth |          1       25.00      100.00
-------------+-----------------------------------
       Total |          4      100.00

. tab maker in 5/8 , nolabel

     maker |      Freq.     Percent        Cum.
-----------+-----------------------------------
         8 |          1       25.00       25.00
        14 |          2       50.00       75.00
        26 |          1       25.00      100.00
-----------+-----------------------------------
     Total |          4      100.00
```

Value labels

- The variable "origin" has value 1, 2 or 3.

```
. tab origin

    origin |      Freq.     Percent        Cum.
-----------+-----------------------------------
         1 |        248       62.47       62.47
         2 |         70       17.63       80.10
         3 |         79       19.90      100.00
-----------+-----------------------------------
     Total |        397      100.00
```

- 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:

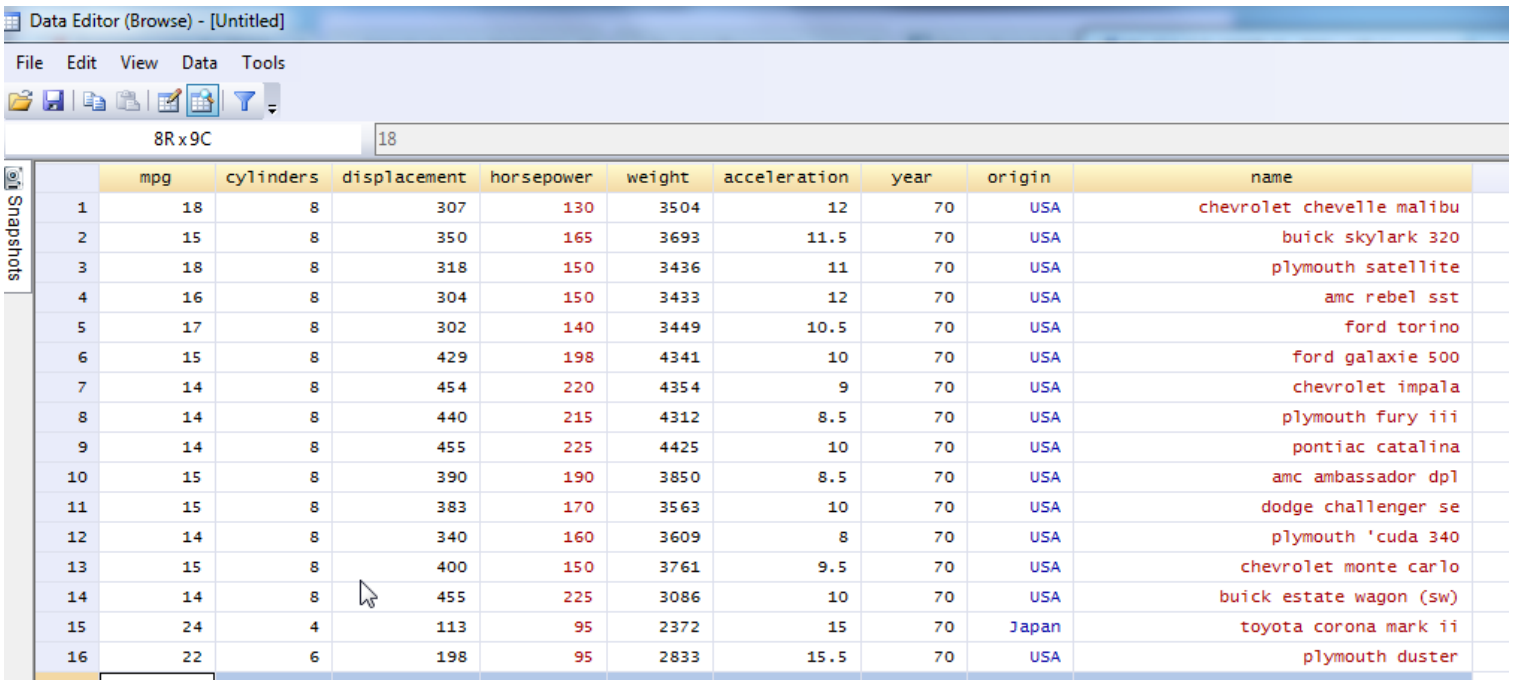
1. Define a mapping from values to labels: a value label.
2. Associate the value label (mapping) to the variable.

Example value label

Use the commands:

```
label define region 1 "USA" 2 "Europe" 3 "Japan"
label values origin region
```

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 origin "Region of Production"
```

```
. describe origin

              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------
origin          byte    %8.0g      region     Region of Production

. tab origin
  Region of |
 Production |      Freq.     Percent        Cum.
------------+-----------------------------------
        USA |        248       62.47       62.47
     Europe |         70       17.63       80.10
      Japan |         79       19.90      100.00
------------+-----------------------------------
      Total |        397      100.00
```

Logical Expressions

If you decide to use the optional `[if exp]` specification you must use a special syntax for logical expressions.

- == equals to
- != not equal to
- > larger than
- >= larger than or equal to
- & and
- | or

Ranges

- 1 gives first observation
- -1 gives the last observation
- 1/5 gives the first five observations.
- -5/-1 gives the last 5 observations

Exercise

- tabulate cylinders for cars produced in Europe
- tabulate cylinders for cars produced in Europe or in Japan
- tabulate cylinders for cars produced in Europe that weigh at least 4000 pounds.
- tabulate cylinders for toyota cars (we have created a string variable and a numeric variable with the name of producers)
- tabulate cylinders for first 10 observation.
- tabulate cylinders for last 10 observation.

Adding and changing variables

```
tab cylinders if origin == 2
tab cylinders if origin == 2 | origin == 3
tab cylinders if origin == 2 & weight >= 4000
tab cylinders if maker_string == "toyota"
//or
tab cylinders if maker == 31
tab cylinders in 1/10
tab cylinders in -10/-1
```

Adding and changing variables

Simple transformations of other variables in order to create a new one and modify it use the commands `generate` and `replace`

```
. sum weight

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
      weight |       397    2970.262    847.9041      1613       5140

. gen heavy = 0 if weight <=4000
(64 missing values generated)

. replace heavy = 1 if weight >4000
(64 real changes made)

// what is the data type ?
// do you need to worry about missings ?
// try : gen byte heavy2 = weight > 4000 if weight <.
```

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 ..
```

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 >= 4000`. Alternatives:

```
gen heavy=0
replace heavy=1 if weight>=4000 & weight!=.
//or
generate byte heavy2 = weight >= 4000 if weight != .
```

- `summarize` uses all the available data.

Do it yourself

Use auto.dta:

- Generate a new variable `manuf_s` with only the first word from the variable make. (i.e extract only the manufacturer)
- Use encode to create a new variable `manuf`
- drop the variables `make` and the unencoded manufacture variable
- Label the variable `rep78` with 1 "Poor" 2 "Fair" 3 "average" 4 "Good" 5 "Excellent".
- Generate a new variable `huge` that take values 1 if a car is longer than 200 inches and 0 otherwise (take care of possible missing values)
- Label the variable `manuf` with "Car Brand"
- Drop observations with missing information on repair record of 78 by using drop if varname >=. (. to stata is stored as a large number)

In case something went wrong

```
use "PATH/auto.dta", clear
gen manuf_s = word(make,1)
encode manuf_s, gen(manuf)
drop manuf_s make
label define quality 1 "Poor" 2 "Fair" 3 "Average" 4 "Good" 5 "Excellent"
label values rep78 quality
gen huge = 0 if length <= 200
replace huge = 1 if length > 200 & length != .
label var manuf "Car Brand"
drop if rep78  >= .
```

Variable naming

Smart naming of your new 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.

- natural logarithm: `gen lninc = ln(income)`

Available functions can be found typing `-help functions-`:

- mathematical functions: `abs()`, `round()`, `sqrt()`
- random numbers: `runiform()`, `rnormal()`
- prob distributions: `normal()`, `ttail()`, `invttail()`

and many more...

`egen` is a smart generate. It works for functions that work across all observations. F.ex:

- `by foreign : egen maxprice = max(price)`
- `egen meany = rowmean(y*)`

Saving the Modified Data

We have made a few modification to the raw Data (generated new variables, drop some variables, atacched labels to variables and variable values) in order to get the Data ready for further analysis. We want to save the new data, so that it is already ready for analysis next time, but:

- **Never overwrite the raw data!**
- Always keep the raw data as it was when downloaded.
- Save the modified data under a new name.

```
save auto_ready.dta
```

To avoid mistakes you can save the raw data under a new name as the first thing you do when load it.

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.

Tabulate

The command `tabulate` gives you the frequency, percent and cumulative percentage. missing values by default excluded. Can include them by adding `missing` as option to the command.

```
. tab rep78
Record 1978 |      Freq.     Percent        Cum.
------------+-----------------------------------
       Poor |          2        2.90        2.90
       Fair |          8       11.59       14.49
    Average |         30       43.48       57.97
       Good |         18       26.09       84.06
  Excellent |         11       15.94      100.00
------------+-----------------------------------
      Total |         69      100.00

. tab rep78, missing
Record 1978 |      Freq.     Percent        Cum.
------------+-----------------------------------
       Poor |          2        2.70        2.70
       Fair |          8       10.81       13.51
    Average |         30       40.54       54.05
       Good |         18       24.32       78.38
  Excellent |         11       14.86       93.24
          . |          5        6.76      100.00
------------+-----------------------------------
      Total |         74      100.00
```

Cross tabulation

Tabulate with two variables gives you the cross tabulation.

```
. tab rep78 foreign

    Repair |
    Record |       Car type
      1978 |  Domestic    Foreign |     Total
-----------+----------------------+----------
      Poor |         2          0 |         2
      Fair |         8          0 |         8
   Average |        27          3 |        30
      Good |         9          9 |        18
 Excellent |         2          9 |        11
-----------+----------------------+----------
     Total |        48         21 |        69
```

- Option `column` reports relative frequency within its column of each cell.
- Option `row` reports relative frequency within its row of each cell.

Correlate

`correlate` displays correlation matrix. Correlations are computed based on the number of pairs with non-missing data.

```
. correlate weight length
(obs=73)

             |   weight   length
-------------+------------------
      weight |   1.0000
      length |   0.9446   1.0000
```

option `covariance` displays covariances:

```
. correlate weight length, covariance
(obs=73)

             |   weight   length
-------------+------------------
      weight |   597175
      length |  16113.9  487.281
```

Exercise

- tabulate the variable make
- Generate a cross tabulation of rep78 and foreign status.
- Sort data by foreign status.
- tabulate rep78 by foreign status
- Generate a cross tabulation of rep78 and foreign status with the relative frequency of each cell within its column.
- Correlate mpg and weight
- Correlate mpg and weight separately by foreign status and test signifcance of correlation.
- rename the variable `make` to `make1` by using
  `rename oldvarname newvarname`
- generate a variable = to the mean value of price by foreign status

In case something went wrong

```
use "PATH/auto.dta", clear
tab make
tab rep78 foreign
sort foreign
by foreign: tab rep78
tab rep78 foreign, column
correlate mpg weight
by foreign: correlate mpg weight
rename make make1
by foreign: egen meanprice =  mean(price)
```

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)