JEM174 Stata for Economists
Institute of Economic Studies
Charles University in Prague

Radek Janhuba
janhuba@gmail.com
Spring 2018

Contents

2 Getting Data In and Out, Inspecting and Describing the Data

This part of the course will first start with some very preliminary data commands, followed by a topic on how to get data from various formats into Stata. After that, we will look into how to explore the data little deeper.

Note that from now on, unless otherwise specified, we will assume that all the files we want to use are located in the working directory which is set by `cd`.

2.1 The Very Basics of Working with Data

2.1.1 Saving a Dataset

Let's start with saving a local copy of the *auto* example dataset. The command to save a dataset is `save`:

```
.  sysuse auto
(1978 Automobile Data)

. save myauto, replace
file myauto.dta saved
```

Note that the option `replace` is only needed in case the file already exists, however, Stata returns an error if the file exists and the option is not specified, while it only shows a warning if it is the other way around:

```
.  save myauto
file myauto.dta already exists
r(602);

.  save myauto, replace
(note: file myauto.dta not found)
file myauto.dta saved
```

2.1.2 Listing Observations

If you want to list (a subset of) observations on the screen, you can use the command `list`. You can use this with connection to `[if]`, `[in]` or for some subset of variables.

```
.list // lists the whole data set (output omitted as it spans several pages)
. list in 1/3 // only first 3 observations, but all variables - also confusing

     +-------------------------------------------------------------------------------
----------------------------+
     | make          price   mpg   rep78   headroom   trunk   weight   length   turn   disp
     |-------------------------------------------------------------------------------
----------------------------|
  1. | AMC Concord   4,099    22       3        2.5      11    2,930      186     40
121       3.58   Domestic |
  2. | AMC Pacer     4,749    17       3        3.0      11    3,350      173     40
258       2.53   Domestic |
  3. | AMC Spirit    3,799    22       .        3.0      12    2,640      168     35
121       3.08   Domestic |
     +-------------------------------------------------------------------------------
----------------------------+

. list make price mpg if mpg > 30 // List only specified vars for cars with mpg>30

     +------------------------------+
     | make             price   mpg |
     |------------------------------|
```

```
 43. | Plym. Champ      4,425    34 |
 57. | Datsun 210       4,589    35 |
 66. | Subaru           3,798    35 |
 68. | Toyota Corolla   3,748    31 |
 71. | VW Diesel        5,397    41 |
     +-------------------------------+
```

```
. list make price mpg if mpg > 30, mean(price) // adds mean of the subset
```

```
      +---------------------------------+
      | make               price   mpg |
      |---------------------------------|
  43. | Plym. Champ        4,425    34 |
  57. | Datsun 210         4,589    35 |
  66. | Subaru             3,798    35 |
  68. | Toyota Corolla     3,748    31 |
  71. | VW Diesel          5,397    41 |
      |---------------------------------|
Mean |                  4,391.4       |
      +---------------------------------+
```

Note that if you have a large dataset, you will often get a virtually unreadable output.

2.1.3 Brief Description of the Dataset

Whenever we want to have a quick look at what we have in Stata, we can use the command `describe`. For now, note how many observations and variables you have and which these are (including their labels in the last column). We will return to the remaining information in a short time.

```
.  describe

Contains data from myauto.dta
  obs:            74                          1978 Automobile Data
 vars:            12                          9 Feb 2018 13:43
 size:         3,182                          (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
make            str18   %-18s                 Make and Model
price           int     %8.0gc                Price
mpg             int     %8.0g                 Mileage (mpg)
rep78           int     %8.0g                 Repair Record 1978
headroom        float   %6.1f                 Headroom (in.)
trunk           int     %8.0g                 Trunk space (cu. ft.)
weight          int     %8.0gc                Weight (lbs.)
length          int     %8.0g                 Length (in.)
turn            int     %8.0g                 Turn Circle (ft.)
displacement    int     %8.0g                 Displacement (cu. in.)
gear_ratio      float   %6.2f                 Gear Ratio
foreign         byte    %8.0g      origin     Car type
```

```
-------------------------------------------------------------------------------
Sorted by: foreign
```

2.1.4 Opening a Stata Format Dataset

Naturally, you are likely to use the Stata datasets (files in `*.dta` format) most often. These are opened using the `use` command:

```
.  use myauto, clear
(1978 Automobile Data)

.describe // output omitted - identical as above
.  use make price mpg using myauto, clear // load only make, price, mpg
(1978 Automobile Data)

. describe

Contains data from myauto.dta
  obs:            74                          1978 Automobile Data
 vars:             3                          9 Feb 2018 13:43
 size:         1,628                          (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
make            str18   %-18s                 Make and Model
price           int     %8.0gc                Price
mpg             int     %8.0g                 Mileage (mpg)
-------------------------------------------------------------------------------
Sorted by:

. use if foreign==1 using "myauto", clear // load only foreign cars
(1978 Automobile Data)

. describe // the dataset includes all variables

Contains data from myauto.dta
  obs:            22                          1978 Automobile Data
 vars:            12                          9 Feb 2018 13:43
 size:           946                          (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
make            str18   %-18s                 Make and Model
price           int     %8.0gc                Price
mpg             int     %8.0g                 Mileage (mpg)
rep78           int     %8.0g                 Repair Record 1978
headroom        float   %6.1f                 Headroom (in.)
trunk           int     %8.0g                 Trunk space (cu. ft.)
weight          int     %8.0gc                Weight (lbs.)
```

```
length          int     %8.0g                 Length (in.)
turn            int     %8.0g                 Turn Circle (ft.)
displacement    int     %8.0g                 Displacement (cu. in.)
gear_ratio      float   %6.2f                 Gear Ratio
foreign         byte    %8.0g      origin     Car type
------------------------------------------------------------------------------------------
Sorted by: foreign

. list make price mpg foreign in 1/3 // only foreign cars

     +-----------------------------------+
     | make        price   mpg   foreign |
     |-----------------------------------|
  1. | Audi 5000   9,690    17   Foreign |
  2. | Audi Fox    6,295    23   Foreign |
  3. | BMW 320i    9,735    25   Foreign |
     +-----------------------------------+
```

2.1.5 Reducing Dataset to a Subset

The above syntax of `use` lets us to load a subset of a dataset into the memory, but does not tell us anything about how to restrict a dataset which is currently loaded. This is controlled by commands `keep` and `drop`. While the first one tells Stata what to keep in memory and drops the rest, the second does exactly the opposite. Both commands can either drop some observations or variables:

```
.  sysuse auto, clear
(1978 Automobile Data)

. sum price foreign // We can see descs of price and percentage of foreign cars

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       price |         74    6165.257    2949.496       3291      15906
     foreign |         74    .2972973    .4601885          0          1

. keep if price<10000 // drops observations (namely "expensive" cars)
(10 observations deleted)

. keep make price foreign // drops variables: only three remain

. sum price foreign // Confirmed - max. price is under 10,000

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       price |         64    5158.641    1412.852       3291       9735
     foreign |         64       .3125    .4671766          0          1

. drop if foreign // drops foreign cars
(20 observations deleted)

. sum price foreign // Mean of foreign is 0 - only domestic cars remain
```

```
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       price |         44    4878.977    1109.672       3291       8814
     foreign |         44           0           0          0          0

. drop foreign // drops the (redundant) foreign variable
```

Watch out! Keep and drop are not reversible. Essentially there are only two ways to get back the original data: either to `save` and then `use`, or use `preserve` and then `restore` (these will save an "image" of the dataset in the current form into a memory and then recall it).

```
.  sysuse auto, clear
(1978 Automobile Data)

. preserve // saves an image of the dataset

. drop if foreign // drops all foreign cars
(22 observations deleted)

. sum foreign // only domestic cars remain

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
     foreign |         52           0           0          0          0

. restore // reloads the image

. sum foreign //the foreign cars are back

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
     foreign |         74    .2972973    .4601885          0          1
```

2.1.6 Sorting the Data

In case you would like to see the data sorted in some order, you can use the command `sort`:

```
.  sysuse auto, clear
(1978 Automobile Data)

. sort make

. list make mpg price in 1/3

     +---------------------------+
     | make          mpg   price |
     |---------------------------|
  1. | AMC Concord    22   4,099 |
  2. | AMC Pacer      17   4,749 |
  3. | AMC Spirit     22   3,799 |
```

```
     +--------------------------+

. sort mpg price

. list make mpg price in 1/3

     +----------------------------------+
     | make                mpg    price |
     |----------------------------------|
  1. | Linc. Continental    12   11,497 |
  2. | Linc. Mark V         12   13,594 |
  3. | Merc. Cougar         14    5,379 |
     +----------------------------------+
```

Note that the `sort` command is only able to sort in ascending order. If you want to do a descending sort, you can use the command `gsort` with a negative sign next to the variable you want to sort in descending order:

```
.  gsort mpg -price

. list make mpg price in 1/3

     +----------------------------------+
     | make                mpg    price |
     |----------------------------------|
  1. | Linc. Mark V         12   13,594 |
  2. | Linc. Continental    12   11,497 |
  3. | Cad. Eldorado        14   14,500 |
     +----------------------------------+
```

2.1.7 Working with Subgroups

Sometimes you want to execute the same command for a subgroup of observations. With many different commands, you can use the `by` and `bysort` prefixes. The difference with these is that `by` only works in case that the data is sorted by the grouping variable:[1]

```
.  by foreign: sum price // This line creates an error "not sorted"
not sorted
r(5);

. bysort foreign: sum price // this line works

-------------------------------------------------------------------------------------------
-> foreign = Domestic

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       price |         52    6072.423    3097.104       3291      15906
```

[1] To be more precise, `bysort` is just `by` with the `sort` option. In other words, they are identical, except `bysort` first sorts the data, while `by` does not.

```
-------------------------------------------------------------------------------
-> foreign = Foreign

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

2.2 Importing Data into Stata

Many times Stata format of data will not be accessible. The following explains some of the options for importing the data from various file types. We will cover two main options, namely importing from excel and from files with a specified delimiter. See `help import` for an overview of other available methods.

2.2.1 Excel Files

Up until Stata 12, the only way to import an excel file was to save it as a csv file and then import it (see below). Since then, Stata has its own way of importing excel files. Suppose now we want to import the data coming from the paper LaLonde (1986), which we have obtained in an excel file:[2]

```
.  import excel nsw.xlsx, firstrow clear

. describe

Contains data
  obs:           722
 vars:            10
 size:        18,050
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
treated         byte    %10.0g                treated
age             byte    %10.0g                age
age2            int     %10.0g                age2
educ            byte    %10.0g                educ
black           byte    %10.0g                black
hispanic        byte    %10.0g                hispanic
married         byte    %10.0g                married
nodegree        byte    %10.0g                nodegree
re75            double  %10.0g                re75
re78            double  %10.0g                re78
-------------------------------------------------------------------------------
Sorted by:
     Note: Dataset has changed since last saved.
```

[2]Notice that the command specifies the file extension as well. In case it is not specified, Stata looks for the older (`*.xls`) excel type file before looking for the newer (`*.xlsx`) file.

The option `firstrow` tells Stata to use the contents of the first line as variable names. Without it, variables would be called *A* to *J*. Note that analogically to `use`, we need `clear` to specify it's OK to replace the data in memory.

Sometimes the excel files will not be as easy looking as the one above. Take the file *NUC02* containing information about the GDP of the Czech Republic. If we write simply

```
. import excel NUC02.xls, clear
```

we get a mess:

```
. list in 4/7
```

```
     +-------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
  4. | A |                              B |                                C |          D |
        E |                                              F |                       G |
     |   | current prices (CZK million) |                                  |            |
     |-------------------------------------------------------------------------------
----------------+-------------------------------------------------------------------|
     |                               H |                     I |
     |                                 |                       |
     |-------------------------------------------------------------------------------
-----------------------------------------------------------------------------------|
     |                                                  M             |
     |                                                                |
     +-------------------------------------------------------------------------------
----------------------------------------------------------------------------------+

     +-------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
  5. | A |                              B |                                C |          D |
        E |                                              F |                       G |
     |   |                                 |                                  |            |
     |-------------------------------------------------------------------------------
----------------+-------------------------------------------------------------------|
     |                               H |                     I |
     |                                 |                       |
     |-------------------------------------------------------------------------------
-----------------------------------------------------------------------------------|
     |                                                  M             |
     |                                                                |
     +-------------------------------------------------------------------------------
----------------------------------------------------------------------------------+

     +-------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
  6. | A |                              B |                                C |          D |
        E |                                                 F |                    G |
     |   |                                   | Final consumption expenditure | Households | Gene
serving households | Gross capital formation |
     |-------------------------------------------------------------------------------
```

```
---------------+-------------------------------------------------------------------------|
     |                              H |                         I |
     | Gross fixed capital formation | Changes in inventories | Acquisitions less disposals
 of valuables | Gross domestic final expenditure | Exports of goods and services[1] |
     |-------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------|
     |                                              M              |
     |            Imports of goods and services[2]                 |            External balanc
     +---------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------+


     +---------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------+
  7. | A |                              B |                              C |          D |
       E |                                                F |                       G |
     |   |                           2004 |                         2134201 |    1481541 |
     |-------------------------------------------------------------------------------------
---------------+-------------------------------------------------------------------------|
     |                              H |                         I |
     |                         860928 |                     34363 |
     |-------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------|
     |                                              M              |
     |                                         1742088             |
     +---------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------+
```

Luckily, Stata has options that allow us to read only a part of the file:

```
.  import excel NUC02.xls, sheet("DATA") cellrange(B6:O17) firstrow clear

. describe

Contains data
  obs:            11
 vars:            14
 size:           660
-------------------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------------------
B               str4    %9s
Finalconsumpt~e long    %10.0g               Final consumption expenditure
Households      long    %10.0g               Households
Generalgovern~t long    %10.0g               General government
Nonprofitinst~g int     %10.0g               Nonprofit institutions serving households
Grosscapitalf~n long    %10.0g               Gross capital formation
Grossfixedcap~n long    %10.0g               Gross fixed capital formation
Changesininve~s long    %10.0g               Changes in inventories
Acquisitionsl~f int     %10.0g               Acquisitions less disposals of valuables
Grossdomestic~e long    %10.0g               Gross domestic final expenditure
```

```
Exportsofgood~1 str7    %9s                 Exports of goods and services[1]
Importsofgood~2 str7    %9s                 Imports of goods and services[2]
Externalbalan~e str6    %9s                 External balance of goods and services
Grossdomestic~t long    %10.0g              Gross domestic product
-------------------------------------------------------------------------------
Sorted by:
     Note: Dataset has changed since last saved.
```

Note that the naming of the variables is messy. In order to rename the variables, you can use e.g. the following code:

```
.  ren B year

. ren Finalconsumptionexpenditure cons_exp

. ren Households hholds

. ren Generalgovernment govt

. ren Nonprofitinstitutionsserving nonprofit

. ren Grosscapitalformation cap_form

. ren Grossfixedcapitalformation fixedcap_form

. ren Changesininventories inventory_change

. ren Acquisitionslessdisposalsof valuables_change

. ren Grossdomesticfinalexpenditure gdp_expenditure

. ren Exportsofgoodsandservices1 exports

. ren Importsofgoodsandservices2 imports

. ren Externalbalanceofgoodsandse ext_balance

. ren Grossdomesticproduct gdp

. describe

Contains data
  obs:            11
 vars:            14
 size:           660
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
year            str4    %9s
cons_exp        long    %10.0g              Final consumption expenditure
```

```
hholds          long    %10.0g                Households
govt            long    %10.0g                General government
nonprofit       int     %10.0g                Nonprofit institutions serving households
cap_form        long    %10.0g                Gross capital formation
fixedcap_form   long    %10.0g                Gross fixed capital formation
inventory_cha~e long    %10.0g                Changes in inventories
valuables_cha~e int     %10.0g                Acquisitions less disposals of valuables
gdp_expenditure long    %10.0g                Gross domestic final expenditure
exports         str7    %9s                   Exports of goods and services[1]
imports         str7    %9s                   Imports of goods and services[2]
ext_balance     str6    %9s                   External balance of goods and services
gdp             long    %10.0g                Gross domestic product
------------------------------------------------------------------------------------------
Sorted by:
     Note: Dataset has changed since last saved.
```

Note that the code may seem drastically messy, however, it does not take much time to create a "helper" in e.g. Excel to produce it. Step-by-step instructions on how to copy the variable names into Excel are included in the Appendix at the end of this file.

Let's now save the dataset to a Stata format as we will need it later in this part of the course:

```
.  save NUC02, replace
file NUC02.dta saved
```

Before we continue, note that if you do not like the messy variable names, you can also go without the `firstrow` option and have the variables named *A,B,...*, however, this way has a substantial disadvantage in sense that Stata will not get the variable labels from the excel file:

```
.  import excel NUC02.xls, sheet("DATA") cellrange(B7:O17) clear

. describe

Contains data
  obs:            11
 vars:            14
 size:           660
------------------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------------------
B               str4    %9s
C               long    %10.0g
D               long    %10.0g
E               long    %10.0g
F               int     %10.0g
G               long    %10.0g
H               long    %10.0g
I               long    %10.0g
J               int     %10.0g
K               long    %10.0g
```

```
L               str7    %9s
M               str7    %9s
N               str6    %9s
O               long    %10.0g
-------------------------------------------------------------------------------------------
Sorted by:
     Note: Dataset has changed since last saved.
```

2.2.2 Delimited (text, csv) Files

You can import the comma-separated files or raw ASCII data files (in case they include a delimiter) using the `import delimited` command. Let's first try to import an example dataset coming from Mitchell (2010):

```
.  import delimited dentists2.txt, clear
(4 vars, 5 obs)

. list

     +-----------------------------------------------+
     |              name   years   fulltime   recom |
     |-----------------------------------------------|
  1. | Y. Don Uflossmore    7.25          0       1 |
  2. |    Olive Tu'Drill   10.25          1       1 |
  3. | Isaac O'Yerbreath   32.75          1       1 |
  4. |      Ruth Canaale      22          1       1 |
  5. |        Mike Avity     8.5          0       0 |
     +-----------------------------------------------+
```

In this case, observations were delimited by a *TAB*. We can see that Stata understood this automatically. In some cases, it is not so easy - take a different form of the file about dentists:

```
.  import delimited dentists4.txt, clear
(1 var, 5 obs)

. list

     +------------------------------+
     |                           v1 |
     |------------------------------|
  1. |  Y. Don Uflossmore":7.25:0:1 |
  2. |    Olive Tu'Drill":10.25:1:1 |
  3. | Isaac O'Yerbreath":32.75:1:1 |
  4. |         Ruth Canaale":22:1:1 |
  5. |          Mike Avity":8.5:0:0 |
     +------------------------------+
```

We can see that the data is delimited by a colon, so we need to specify it:

```
. import delimited dentists4.txt, delimiter(":") clear
(4 vars, 5 obs)

. list

     +-----------------------------------+
     |                v1      v2   v3   v4 |
     |-----------------------------------|
  1. | Y. Don Uflossmore    7.25    0    1 |
  2. |    Olive Tu'Drill   10.25    1    1 |
  3. | Isaac O'Yerbreath   32.75    1    1 |
  4. |      Ruth Canaale      22    1    1 |
  5. |        Mike Avity     8.5    0    0 |
     +-----------------------------------+
```

Note that in case of `import delimited`, Stata names the new variables automatically as *v1* to *vX*. In order to rename them, we can use the similar way as in the previous section, or, in case of several variables only, we can use the second syntax of the `rename` command:

```
. ren (v1 v2 v3 v4) (name years fulltime recom)

. list in 1

     +-----------------------------------------------+
     |              name   years   fulltime   recom |
     |-----------------------------------------------|
  1. | Y. Don Uflossmore    7.25          0       1 |
     +-----------------------------------------------+
```

Alternatively, if you know beforehand how you want to name all these variables, you can specify it already in the import command:

```
. import delimited name years fulltime recom using dentists4.txt, ///
                                delimiter(":") clear

(4 vars, 5 obs)

. list in 1

     +-----------------------------------------------+
     |              name   years   fulltime   recom |
     |-----------------------------------------------|
  1. | Y. Don Uflossmore    7.25          0       1 |
     +-----------------------------------------------+
```

Note that you can also use this command to load the csv-files generated by spreadsheets such as MS Excel:

```
. import delimited nsw.csv, varnames(1) clear
(10 vars, 722 obs)
```

```
. describe

Contains data
  obs:           722
 vars:            10
 size:        12,274
-------------------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------------------
treated         byte    %8.0g
age             byte    %8.0g
age2            int     %8.0g
educ            byte    %8.0g
black           byte    %8.0g
hispanic        byte    %8.0g
married         byte    %8.0g
nodegree        byte    %8.0g
re75            float   %9.0g
re78            float   %9.0g
-------------------------------------------------------------------------------------------
Sorted by:
     Note: Dataset has changed since last saved.
```

Notes:

- The option `varnames(1)` tells Stata where the variable names are.
- In this case we do not actually need to specify `varnames(1)` as Stata understands it automatically.
- The `import delimited` is a relatively new command that replaced command `insheet`. The old command still works, but will not be updated.

2.3 Variables in Stata

Each variable has the following four characteristics that are associated with the variable rather than with the actual observations:

1. storage type
2. format
3. variable label (if any)
4. value label (if any)

In principle, there are two classes of variables, strings and numeric. The easiest explanation is that strings store text and numeric variables store numbers. However, keep in mind that numbers can be also stored as strings (we will discuss that later).

2.3.1 Storage Types

The following summary shows you all types of possible variables in Stata.

```
Strings:
str1 to str244 (depending on how long is the longest entry)

Numeric:
type         Minimum               Maximum    Bytes
-------------------------------------------------------------
byte            -127                   100       1
int          -32,767                32,740       2
long  -2,147,483,647        2,147,483,620       4
float   -1.70141173319*10^38  1.70141173319*10^36       4
double  -8.9884656743*10^307  8.9884656743*10^307       8
-------------------------------------------------------------
```

In principle, strings are "easy" to understand - the variable type is (or at least should be) the length of the longest observation.

Numeric variables come in five different types and are distinct in terms of how much disk/memory space they need to allocate and whether or not they are able to hold decimal values.

Let's now introduce one of very useful commands. If you are not sure whether your variables are stored in the correct types, you can type `compress`

```
.  sysuse auto, clear
(1978 Automobile Data)

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

and all variables that can be reduced to a less-demanding type without losing information will be transformed. In cases where the dataset is stored efficiently, Stata simply reports "0 bytes saved":

```
.  compress
  (0 bytes saved)
```

Note that if you work with small datasets, you probably do not need to worry about trying to reduce the sample size by keeping the efficient variable types. Nevertheless, it is a good practice which can save you substantial space while working with large datasets.

2.3.2 Formats

The following is a list of (maybe all) formats you can specify for a variable. If it is too complicated, do not worry, it is not crucial. However, keep in mind that there are a lot of ways how you can format your variables. Take e.g. dates, which are stored as numbers in Stata. If you leave general number, you will get confused. However, you can specify a "date format" (either general or user specified) and you will not get lost in it (we will go more through these when we use dates later on).

```
%fmt description                                  example
------------------------------------------------------------------------------
Right-justified formats
%#.#ggeneral numeric format                       %9.0g
%#.#ffixed numeric format                         %9.2f
%#.#eexponential numeric format                   %10.7e
%d   default numeric elapsed date format          %d
%d...user-specified elapsed date format           %dM/D/Y
%#s  string format                                %15s
Right-justified, comma formats
%#.#gc       general numeric format               %9.0gc
%#.#fc       fixed numeric format                 %9.2fc
Leading-zero formats
%0#.#f       fixed numeric format                 %09.2f
%0#s string format                                %015s
Left-justified formats
%-#.#g       general numeric format               %-9.0g
%-#.#f       fixed numeric format                 %-9.2f
%-#.#e       exponential numeric format           %-10.7e
%-d  default numeric elapsed date format          %-d
%-d...       user-specified elapsed date format      %-dM/D/Y
%-#s string format                                %-15s
Left-justified, comma formats
%-#.#gc      general numeric format               %-9.0gc
%-#.#fc      fixed numeric format                 %-9.2fc
Centered formats
%~#s string format (special)                      %~15s
------------------------------------------------------------------------------
```

Let's have a look at which variables have which properties and what does it mean:

```
. sysuse auto, clear
(1978 Automobile Data)

. describe

Contains data from C:\Programs\Stata14\ado\base/a/auto.dta
  obs:            74                          1978 Automobile Data
 vars:            12                          13 Apr 2014 17:45
 size:         3,182                          (_dta has notes)
------------------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------------------
make            str18   %-18s                 Make and Model
price           int     %8.0gc                Price
mpg             int     %8.0g                 Mileage (mpg)
rep78           int     %8.0g                 Repair Record 1978
headroom        float   %6.1f                 Headroom (in.)
trunk           int     %8.0g                 Trunk space (cu. ft.)
weight          int     %8.0gc                Weight (lbs.)
```

```
length          int     %8.0g                 Length (in.)
turn            int     %8.0g                 Turn Circle (ft.)
displacement    int     %8.0g                 Displacement (cu. in.)
gear_ratio      float   %6.2f                 Gear Ratio
foreign         byte    %8.0g      origin     Car type
------------------------------------------------------------------------------------------
Sorted by: foreign
```

We can see that we have one string and the rest are numeric variables in various formats. We can also see that foreign, the only byte, has a value label. Let's list the first five observations so that we can compare different formats:

```
. list make price mpg headroom gear_ratio foreign in 1/5

     +-------------------------------------------------------------------+
     | make            price   mpg   headroom   gear_r~o    foreign |
     |-------------------------------------------------------------------|
  1. | AMC Concord     4,099    22        2.5       3.58   Domestic |
  2. | AMC Pacer       4,749    17        3.0       2.53   Domestic |
  3. | AMC Spirit      3,799    22        3.0       3.08   Domestic |
  4. | Buick Century   4,816    20        4.5       2.93   Domestic |
  5. | Buick Electra   7,827    15        4.0       2.41   Domestic |
     +-------------------------------------------------------------------+
```

From the output, we can identify that `c` means comma-separated thousands, `.X` means X decimal places, etc.

2.3.3 Variable Labels

A variable label is a description of the variable, such as "Household income". You can find them in the last column of the output from `describe`.

If you want to add or change the variable label, you can use the command `label variable` or its shorter abbreviation `la var`

```
. label variable price "Price in 1978 US Dollars"

. la var mpg "Miles per gallon"

. describe price mpg // variable labels are changed

              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------------------
price           int     %8.0gc                Price in 1978 US Dollars
mpg             int     %8.0g                 Miles per gallon
```

2.3.4 Value Labels

Value labels assign some verbal descriptions to individual values, such as e.g.

```
1=University degree
0=No university degree
```

In order to change value labels of a variable, one must first define the value label and then attach it to the variable in question.

Let's now briefly return to the NSW dataset and value label one of the variables:

```
.  import excel nsw.xlsx, firstrow clear // get the data

. describe // we can see that we do have variable labels, but not value labels.

Contains data
  obs:           722
 vars:            10
 size:        18,050
-------------------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------------------
treated         byte    %10.0g                treated
age             byte    %10.0g                age
age2            int     %10.0g                age2
educ            byte    %10.0g                educ
black           byte    %10.0g                black
hispanic        byte    %10.0g                hispanic
married         byte    %10.0g                married
nodegree        byte    %10.0g                nodegree
re75            double  %10.0g                re75
re78            double  %10.0g                re78
-------------------------------------------------------------------------------------------
Sorted by:
     Note: Dataset has changed since last saved.

. label define married 0 "Not Married" 1 "Married" // define a label

. label values married married // attach the defined label to the variable

. describe // we can see that it is attached now

Contains data
  obs:           722
 vars:            10
 size:        18,050
-------------------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------------------
treated         byte    %10.0g                treated
age             byte    %10.0g                age
age2            int     %10.0g                age2
educ            byte    %10.0g                educ
```

```
black           byte    %10.0g                black
hispanic        byte    %10.0g                hispanic
married         byte    %11.0g     married    married
nodegree        byte    %10.0g                nodegree
re75            double  %10.0g                re75
re78            double  %10.0g                re78
------------------------------------------------------------------------------------------
Sorted by:
     Note: Dataset has changed since last saved.

. list age married nodegree in 1/3 // the label shows up in the data

     +------------------------------+
     | age       married   nodegree |
     |------------------------------|
  1. |  24   Not Married          0 |
  2. |  34   Not Married          0 |
  3. |  18   Not Married          1 |
     +------------------------------+
```

Notes:

- When you open the data browser, you will discover that value-labeled variables are shown with blue color - they appear as strings, but are actually numeric.
- In the example above, I chose to name the value label as the variable in question - this is often the case, but it is not a requirement. If you use a different name, the syntax goes the following: `label values` *`varlist labelname`*.
- Although keeping the value labels in order requires some more technicalities, the general rule of thumb (assuming one wants to keep good programming practices) is that any time a variable is categorical, it should be numeric and value labeled.

2.3.5 Identifying Lists of Variables

Generally, a list of variable names is called a *varlist* in Stata. In order to automatically list all variables on the screen, we can utilize a very useful command `ds`:

```
. ds
treated   age       age2      educ      black     hispanic  married   nodegree  re75      r
```

The command `ds` has some useful options, which can occasionaly be used to obtain the list of variables matching some pattern or having some characteristics. As an example, the following command shows a list of all variables with names that do not contain the letter "a". See `help ds` for more information and examples.

```
. ds *a*, not
educ      nodegree  re75      re78
```

2.4 Cleaning the Data

In a typical research project, once you have the data imported and labeled, it is time to clean them.[3]

2.4.1 Missing Values

Part of your work on a project will always inevitably involve dealing with missing values. Things to remember in this area are:

- Stata always has to have "something" in each observation. If some variable is not available for a particular observation, it fills it with "missing".
- By default, missing values are represented by a dot for numeric variables and by an "empty" for a string (""). Note that this is different from a string that contains a space (" ").
- Missing values for numeric variables are treated as large numbers.
- There is a Stata function `missing()` which can tell whether a particular observation is missing (we will use it later in the course).
- In case you wish to do so, you can specify up to 27 missing categories by using the missing values ".a" to ".z" besides simple "." (sometimes, especially in surveys, it is important to e.g. distinguish between "refused", "I don't know" and "not applicable" categories)

2.4.2 Utilizing/Creating Value Labeled Datasets

Many times you will import the data and discover that one of the categorical variables is a string, rather than a numerical. This causes several issues - firstly, many of Stata's procedures cannot run based on string variables, secondly, such storage may increase the data size substantially.

Let's now open a web-based example dataset *hbp2*:[4]

```
.  webuse hbp2, clear

. describe

Contains data from http://www.stata-press.com/data/r14/hbp2.dta
  obs:         1,130
 vars:             7                          3 Mar 2014 06:47
 size:        24,860
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
id              str10   %10s                  Record identification number
city            byte    %8.0g
year            int     %8.0g
age_grp         byte    %8.0g      agefmt
race            byte    %8.0g      racefmt
hbp             byte    %8.0g      yn         high blood pressure
sex             str6    %9s
-------------------------------------------------------------------------------
Sorted by:
```

[3] Actually, from some point of view, it makes sense to label the dataset *after* cleaning - the choice is always on the individual researcher.

[4] Note that `webuse` is in a sense really similar to `sysuse`.

The quick glance at the data reveals that *sex* is a string variable. Let's now look at how it's coded:

```
. list sex in 1/5

     +--------+
     |    sex |
     |--------|
  1. | female |
  2. |        |
  3. |   male |
  4. |   male |
  5. | female |
     +--------+
```

In order to create a value-labeled variable containing the same information, we can use the command `encode`:

```
. encode sex, gen(gender) // new variable is created
```

If we now repeat the listing command, there seems to be no difference:[5]

```
. list sex gender in 1/5

     +-----------------+
     |    sex   gender |
     |-----------------|
  1. | female   female |
  2. |               . |
  3. |   male     male |
  4. |   male     male |
  5. | female   female |
     +-----------------+
```

If we, however, tell Stata to ignore labels while listing, we see that the new variable is in fact numeric:

```
. list sex gender in 1/5, nolabel

     +-----------------+
     |    sex   gender |
     |-----------------|
  1. | female        1 |
  2. |               . |
  3. |   male        2 |
  4. |   male        2 |
  5. | female        1 |
     +-----------------+
```

[5] In fact, there is a little difference in the appearance of the missing value.

Moreover, we can see that this simple `encode` reduces the dataset size by almost 23%:

```
.  drop sex

. compress
  variable gender was long now byte
  (3,390 bytes saved)

. describe

Contains data from http://www.stata-press.com/data/r14/hbp2.dta
  obs:         1,130
 vars:             7                          3 Mar 2014 06:47
 size:        19,210
-------------------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------------------
id              str10   %10s                  Record identification number
city            byte    %8.0g
year            int     %8.0g
age_grp         byte    %8.0g      agefmt
race            byte    %8.0g      racefmt
hbp             byte    %8.0g      yn         high blood pressure
gender          byte    %8.0g      gender
-------------------------------------------------------------------------------------------
Sorted by:
     Note: Dataset has changed since last saved.
```

Suppose now that you wanted to go the other way around, namely create a string variable out of a value-labeled one. This can be done by an associated command `decode`:

```
.  decode gender, gen(sex) // creates a variable identical to the original one

. list sex gender in 1/5, nolabel

     +-----------------+
     |    sex   gender |
     |-----------------|
  1. | female        1 |
  2. |               . |
  3. |   male        2 |
  4. |   male        2 |
  5. | female        1 |
     +-----------------+
```

2.4.3 Getting from Strings to Numbers and the Other Way Around

Sometimes, especially after importing data, your variables will get imported as strings or numbers, while you wanted it to be of the other type.

Let's get back to the GDP dataset we earlier imported and saved into the Stata format:

```
. use NUC02, clear

. describe

Contains data from NUC02.dta
  obs:            11
 vars:            14                          9 Feb 2018 13:44
 size:           660
-------------------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------------------
year            str4    %9s
cons_exp        long    %10.0g               Final consumption expenditure
hholds          long    %10.0g               Households
govt            long    %10.0g               General government
nonprofit       int     %10.0g               Nonprofit institutions serving households
cap_form        long    %10.0g               Gross capital formation
fixedcap_form   long    %10.0g               Gross fixed capital formation
inventory_cha~e long    %10.0g               Changes in inventories
valuables_cha~e int     %10.0g               Acquisitions less disposals of valuables
gdp_expenditure long    %10.0g               Gross domestic final expenditure
exports         str7    %9s                  Exports of goods and services[1]
imports         str7    %9s                  Imports of goods and services[2]
ext_balance     str6    %9s                  External balance of goods and services
gdp             long    %10.0g               Gross domestic product
-------------------------------------------------------------------------------------------
Sorted by:
```

We can see that some variables (namely *year, exports, imports*, and *ext_balance*) are strings, but we would expect them to be numeric. Let's now apply the command that will solve the situation:

```
. destring, replace
year: all characters numeric; replaced as int
cons_exp already numeric; no replace
hholds already numeric; no replace
govt already numeric; no replace
nonprofit already numeric; no replace
cap_form already numeric; no replace
fixedcap_form already numeric; no replace
inventory_change already numeric; no replace
valuables_change already numeric; no replace
gdp_expenditure already numeric; no replace
exports: all characters numeric; replaced as long
(1 missing value generated)
imports: all characters numeric; replaced as long
(1 missing value generated)
ext_balance: all characters numeric; replaced as long
(1 missing value generated)
gdp already numeric; no replace
```

```
. describe

Contains data from NUC02.dta
  obs:            11
 vars:            14                          9 Feb 2018 13:44
 size:           550
-------------------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------------------
year            int     %10.0g
cons_exp        long    %10.0g               Final consumption expenditure
hholds          long    %10.0g               Households
govt            long    %10.0g               General government
nonprofit       int     %10.0g               Nonprofit institutions serving households
cap_form        long    %10.0g               Gross capital formation
fixedcap_form   long    %10.0g               Gross fixed capital formation
inventory_cha~e long    %10.0g               Changes in inventories
valuables_cha~e int     %10.0g               Acquisitions less disposals of valuables
gdp_expenditure long    %10.0g               Gross domestic final expenditure
exports         long    %10.0g               Exports of goods and services[1]
imports         long    %10.0g               Imports of goods and services[2]
ext_balance     long    %10.0g               External balance of goods and services
gdp             long    %10.0g               Gross domestic product
-------------------------------------------------------------------------------------------
Sorted by:
     Note: Dataset has changed since last saved.
```

We can see that all variables are numeric now.

Most of the time, however, the situation will not be as easy as in the example above. The following points should help you make the procedure work as well as avoid crucial mistakes:

1. If you use `destring` without *varlist*, Stata looks for all variables. Use with *varlist* if you only want to destring some of them (typically there will be some variables that are supposed to be strings - e.g. *make* in the *auto* dataset)
2. In this case, `replace` means that Stata should not generate a new variable, but rather replace the existing one with the numerical one. An alternative is an option `generate(`*newvarlist*`)` which will keep old as well as new variables.
3. In case there would be a true string in the data, you need to specify the option `force` to delete the information contained in such string. As this leads to a direct loss of data, use `force` **with caution!**

The opposite of `destring` is `tostring`:

```
.  tostring year, gen(txtyear)
txtyear generated as str4
```

2.5 Inspecting the Data

We've already seen commands `describe`, `list` and `summarize`. In this section, we will cover some additional commands allowing us to look at what data is loaded in Stata.

```
. sysuse auto, clear
(1978 Automobile Data)
```

2.5.1 Quick Inspection of Data

In case we want to quickly check the basic properties of an unfamiliar data, we can use the command `inspect`. It shows us:

- Number of positive, negative and missing values
- Number of integers and nonintegers
- Number of unique values
- Number of missing
- It produces a small histogram

```
. inspect price rep78
```

```
price:  Price                                   Number of Observations
------------                            ---------------------------------------
                                              Total      Integers   Nonintegers
|  #                        Negative            -             -             -
|  #                        Zero                -             -             -
|  #                        Positive           74            74             -
|  #                                    -----------   -----------   -----------
|  #                        Total              74            74             -
|  #   #   .   .   .        Missing             -
+----------------------                 -----------
3291              15906                        74
  (74 unique values)

rep78:  Repair Record 1978                      Number of Observations
--------------------------              ---------------------------------------
                                              Total      Integers   Nonintegers
|          #                Negative            -             -             -
|          #                Zero                -             -             -
|          #                Positive           69            69             -
|          #   #                        -----------   -----------   -----------
|          #   #   #        Total              69            69             -
|  .   #   #   #   #        Missing             5
+----------------------                 -----------
1                     5                        74
   (5 unique values)
```

2.5.2 Quick Codebooks

A similar command to `describe` is `codebook`. In case of string variables, `codebook` shows us

- type including warning that it may be stored inefficiently (possible compress)
- how many unique values

- how many missing
- some examples
- warning that it contains blanks

```
. codebook make

-------------------------------------------------------------------------------------------
make                                                                          Make and Model
-------------------------------------------------------------------------------------------

                  type:  string (str18), but longest is str17

         unique values:  74                       missing "":  0/74

              examples:  "Cad. Deville"
                         "Dodge Magnum"
                         "Merc. XR-7"
                         "Pont. Catalina"

               warning:  variable has embedded blanks
```

In case of numeric variables, we obtain

- type
- units
- how many missing
- how many unique values
- range
- mean and standard deviation
- basic percentiles

```
. codebook price

-------------------------------------------------------------------------------------------
price                                                                                 Price
-------------------------------------------------------------------------------------------

                  type:  numeric (int)

                 range:  [3291,15906]                 units:  1
         unique values:  74                       missing .:  0/74

                  mean:   6165.26
             std. dev:    2949.5

           percentiles:        10%       25%       50%       75%       90%
                              3895      4195    5006.5      6342     11385
```

Note that if you simply write `codebook` without a *varlist*, you can get lost in the long list of information.

2.5.3 Simple Counts of Data

You may want to look at how many observations satisfy some criteria. You can do this with the `count` command together with `if`:

```
.  count // counts the full number of observations
  74

. count if foreign==0 & mpg>20 // only domestic and efficient cars
  19

. count if !foreign & mpg>=21 // same as above
  19
```

Note that the last two commands produce exactly the same output. This is because they are logically equivalent (in case *mpg* is an integer), just the syntax differs.

2.5.4 Summary Statistics

We already know that the command for displaying summary statistics is `sum`. Let's now look at one modification. Consider the difference between the following two commands:

```
.  sum price mpg

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       price |         74    6165.257    2949.496       3291      15906
         mpg |         74     21.2973    5.785503         12         41

. sum price mpg, detail

                            Price
-------------------------------------------------------------
      Percentiles      Smallest
 1%         3291           3291
 5%         3748           3299
10%         3895           3667       Obs                  74
25%         4195           3748       Sum of Wgt.          74

50%       5006.5                      Mean           6165.257
                        Largest       Std. Dev.      2949.496
75%         6342          13466
90%        11385          13594       Variance        8699526
95%        13466          14500       Skewness       1.653434
99%        15906          15906       Kurtosis       4.819188

                        Mileage (mpg)
-------------------------------------------------------------
      Percentiles      Smallest
```

```
 1%          12             12
 5%          14             12
10%          14             14       Obs                 74
25%          18             14       Sum of Wgt.         74

50%          20                      Mean           21.2973
                        Largest      Std. Dev.      5.785503
75%          25             34
90%          29             35       Variance       33.47205
95%          34             35       Skewness       .9487176
99%          41             41       Kurtosis       3.975005
```

As we can see, the `detail` option (which can be abbreviated as `det`) contains all of the default plus:

- 9 different percentile values (including the median at 50)
- 4 smallest and 4 highest values
- Variance, skewness, kurtosis
- sum of weights

2.5.5 Listing Category Frequencies

One of the best commands in Stata is `tabulate` (or `tab`). There are two basic ways how you can use it. The first is a one way tabulate, which shows you number of observations based on their values, their relative frequency and their cumulative distribution:

```
.  tabulate rep78

     Repair |
Record 1978 |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |          2        2.90        2.90
          2 |          8       11.59       14.49
          3 |         30       43.48       57.97
          4 |         18       26.09       84.06
          5 |         11       15.94      100.00
------------+-----------------------------------
      Total |         69      100.00

. tab foreign // notice that tab understands value labels.

   Car type |      Freq.     Percent        Cum.
------------+-----------------------------------
   Domestic |         52       70.27       70.27
    Foreign |         22       29.73      100.00
------------+-----------------------------------
      Total |         74      100.00
```

One of the extremely useful applications is to plot summary statistics of some variable based on categories of another variable:

```
. tab rep78, summarize(price)

     Repair |          Summary of Price
Record 1978 |        Mean   Std. Dev.       Freq.
------------+------------------------------------
          1 |     4,564.5   522.55191           2
          2 |   5,967.625   3,579.357           8
          3 |   6,429.233    3,525.14          30
          4 |     6,071.5   1,709.608          18
          5 |       5,913   2,615.763          11
------------+------------------------------------
      Total |   6,146.043    2,912.44          69
```

The second way how to use tab is to tabulate the data in two ways:

```
. tab rep78 foreign

    Repair |
    Record |       Car type
      1978 |  Domestic    Foreign |     Total
-----------+----------------------+----------
         1 |         2          0 |         2
         2 |         8          0 |         8
         3 |        27          3 |        30
         4 |         9          9 |        18
         5 |         2          9 |        11
-----------+----------------------+----------
     Total |        48         21 |        69
```

(note that the ordering of the two matters). We can also use the following options to report the relative frequencies:

```
row: row percentages
column: column percentages
cell: total percentages
```

When using these, it may be a good idea to also use the `nofreq` option, which suppresses the absolute frequencies. Notice the difference between the following outputs:

```
. tab rep78 foreign, cell // the results are not really easily readable

+-----------------+
| Key             |
|-----------------|
|    frequency    |
| cell percentage |
+-----------------+

    Repair |
    Record |       Car type
```

```
      1978 |  Domestic    Foreign |     Total
-----------+----------------------+----------
         1 |         2          0 |         2
           |      2.90       0.00 |      2.90
-----------+----------------------+----------
         2 |         8          0 |         8
           |     11.59       0.00 |     11.59
-----------+----------------------+----------
         3 |        27          3 |        30
           |     39.13       4.35 |     43.48
-----------+----------------------+----------
         4 |         9          9 |        18
           |     13.04      13.04 |     26.09
-----------+----------------------+----------
         5 |         2          9 |        11
           |      2.90      13.04 |     15.94
-----------+----------------------+----------
     Total |        48         21 |        69
           |     69.57      30.43 |    100.00

. tab rep78 foreign, cell nofreq // now we see total percentages

    Repair |
    Record |       Car type
      1978 |  Domestic    Foreign |     Total
-----------+----------------------+----------
         1 |      2.90       0.00 |      2.90
         2 |     11.59       0.00 |     11.59
         3 |     39.13       4.35 |     43.48
         4 |     13.04      13.04 |     26.09
         5 |      2.90      13.04 |     15.94
-----------+----------------------+----------
     Total |     69.57      30.43 |    100.00

. tab rep78 foreign, row nofreq // just row percentages

    Repair |
    Record |       Car type
      1978 |  Domestic    Foreign |     Total
-----------+----------------------+----------
         1 |    100.00       0.00 |    100.00
         2 |    100.00       0.00 |    100.00
         3 |     90.00      10.00 |    100.00
         4 |     50.00      50.00 |    100.00
         5 |     18.18      81.82 |    100.00
-----------+----------------------+----------
     Total |     69.57      30.43 |    100.00

. tab rep78 foreign, column nofreq // just column percentages

    Repair |
```

```
    Record |       Car type
      1978 |  Domestic    Foreign |     Total
-----------+----------------------+----------
         1 |      4.17       0.00 |      2.90
         2 |     16.67       0.00 |     11.59
         3 |     56.25      14.29 |     43.48
         4 |     18.75      42.86 |     26.09
         5 |      4.17      42.86 |     15.94
-----------+----------------------+----------
     Total |    100.00     100.00 |    100.00
```

Note that summary statistics can be reported in the two way table as well:

```
. tab rep78 foreign, sum(price) nofreq

                Means and Standard Deviations of Price

    Repair |
    Record |      Car type
      1978 |  Domestic    Foreign |     Total
-----------+----------------------+----------
         1 |   4,564.5          . |   4,564.5
           | 522.55191          . | 522.55191
-----------+----------------------+----------
         2 | 5,967.625          . | 5,967.625
           | 3,579.357          . | 3,579.357
-----------+----------------------+----------
         3 | 6,607.074  4,828.667 | 6,429.233
           | 3,661.267  1,285.613 |  3,525.14
-----------+----------------------+----------
         4 | 5,881.556  6,261.444 |   6,071.5
           | 1,592.019  1,896.092 | 1,709.608
-----------+----------------------+----------
         5 |   4,204.5  6,292.667 |     5,913
           | 311.83409  2,765.629 | 2,615.763
-----------+----------------------+----------
     Total |  6,179.25  6,070.143 | 6,146.043
           | 3,188.969  2,220.984 |  2,912.44
```

Besides of simple tabulating the data, `tab` can also calculate various statistics regarding the distribution of the tabulated data. Perhaps the most applicable is the calculation of Pearson's χ^2 statistic, with a null hypothesis that columns and rows are independent of each other:

```
. tab rep78 foreign, chi2

    Repair |
    Record |       Car type
      1978 |  Domestic    Foreign |     Total
-----------+----------------------+----------
         1 |         2          0 |         2
```

```
         2 |         8          0 |         8
         3 |        27          3 |        30
         4 |         9          9 |        18
         5 |         2          9 |        11
-----------+----------------------+----------
     Total |        48         21 |        69

          Pearson chi2(4) =  27.2640   Pr = 0.000
```

In this case, the test suggest that the rows and columns are statistically different from each other. We can also see which category of cars has the highest contribution to the overall χ^2 test statistic:

```
. tab rep78 foreign, chi2 cchi2 nofreq

    Repair |
    Record |       Car type
      1978 |  Domestic    Foreign |     Total
-----------+----------------------+----------
         1 |       0.3        0.6 |       0.9
         2 |       1.1        2.4 |       3.5
         3 |       1.8        4.1 |       5.9
         4 |       1.0        2.3 |       3.3
         5 |       4.2        9.5 |      13.7
-----------+----------------------+----------
     Total |       8.3       19.0 |      27.3

          Pearson chi2(4) =  27.2640   Pr = 0.000
```

2.5.6 Tabulating Multiple Summary Statistics

A command "merging" the `tab` and `sum` commands is `tabstat`. It lets you to report summary statistics of multiple numeric variables in one table, possibly separated by another categorical variable.

If we want only mean,

```
. tabstat price mpg weight length

   stats |     price       mpg    weight    length
---------+----------------------------------------
    mean |  6165.257   21.2973  3019.459  187.9324
--------------------------------------------------
```

If we want more statistics, we specify the `stat` option (see `help tabstat` for a list of possible statistics)

```
. tabstat price mpg weight length, stat(count mean sd min max) // Stats in cols

   stats |     price       mpg    weight    length
---------+----------------------------------------
       N |        74        74        74        74
    mean |  6165.257   21.2973  3019.459  187.9324
```

```
      sd |  2949.496  5.785503  777.1936  22.26634
     min |      3291        12      1760       142
     max |     15906        41      4840       233
----------------------------------------------------
```

If we want to have statistics in columns and variables in rows, we can specify `column` option with `statistics` inside it:

```
. tabstat price mpg weight length, stat(count mean sd min max) col(stat)

    variable |         N      mean        sd       min       max
-------------+--------------------------------------------------
       price |        74  6165.257  2949.496      3291     15906
         mpg |        74   21.2973  5.785503        12        41
      weight |        74  3019.459  777.1936      1760      4840
      length |        74  187.9324  22.26634       142       233
---------------------------------------------------------------
```

Both of these may also be combined with `by` prefix or option, however, watch out, things may get confusing! Also, note that `by` option is not the same as `by` (or `bysort`) prefix! Consider the difference between the two following commands:

```
. tabstat price mpg weight length, stat(count mean sd) by(rep78) col(stat)

Summary for variables: price mpg weight length
     by categories of: rep78 (Repair Record 1978)

   rep78 |         N      mean        sd
---------+----------------------------
       1 |         2    4564.5  522.5519
         |         2        21  4.242641
         |         2      3100   523.259
         |         2       189  12.72792
---------+----------------------------
       2 |         8  5967.625  3579.357
         |         8    19.125  3.758324
         |         8   3353.75  445.9961
         |         8   199.375  13.97894
---------+----------------------------
       3 |        30  6429.233   3525.14
         |        30  19.43333  4.141325
         |        30      3299  752.4184
         |        30       194  20.72313
---------+----------------------------
       4 |        18    6071.5  1709.608
         |        18  21.66667   4.93487
         |        18      2870  907.2842
         |        18  184.8333  28.65053
---------+----------------------------
       5 |        11      5913  2615.763
```

```
         |        11  27.36364  8.732385
         |        11  2322.727  410.5628
         |        11  170.1818  12.09808
---------+----------------------------
   Total |        69  6146.043   2912.44
         |        69  21.28986  5.866408
         |        69  3032.029  792.8515
         |        69  188.2899   22.7474
-------------------------------------

. bysort rep78: tabstat price mpg weight length, stat(count mean sd) col(stat)

-------------------------------------------------------------------------------------------
-> rep78 = 1

    variable |         N      mean        sd
-------------+----------------------------
       price |         2    4564.5  522.5519
         mpg |         2        21  4.242641
      weight |         2      3100   523.259
      length |         2       189  12.72792
-------------------------------------------

-------------------------------------------------------------------------------------------
-> rep78 = 2

    variable |         N      mean        sd
-------------+----------------------------
       price |         8  5967.625  3579.357
         mpg |         8    19.125  3.758324
      weight |         8   3353.75  445.9961
      length |         8   199.375  13.97894
-------------------------------------------

-------------------------------------------------------------------------------------------
-> rep78 = 3

    variable |         N      mean        sd
-------------+----------------------------
       price |        30  6429.233   3525.14
         mpg |        30  19.43333  4.141325
      weight |        30      3299  752.4184
      length |        30       194  20.72313
-------------------------------------------

-------------------------------------------------------------------------------------------
-> rep78 = 4

    variable |         N      mean        sd
-------------+----------------------------
       price |        18    6071.5  1709.608
```

```
         mpg |        18  21.66667   4.93487
      weight |        18      2870  907.2842
      length |        18  184.8333  28.65053
-------------------------------------------

-------------------------------------------------------------------------------------------
-> rep78 = 5

    variable |         N      mean        sd
-------------+--------------------------------
       price |        11      5913  2615.763
         mpg |        11  27.36364  8.732385
      weight |        11  2322.727  410.5628
      length |        11  170.1818  12.09808
-------------------------------------------

-------------------------------------------------------------------------------------------
-> rep78 = .

    variable |         N      mean        sd
-------------+--------------------------------
       price |         5    6430.4  3804.322
         mpg |         5      21.4   5.07937
      weight |         5      2846  544.7752
      length |         5       183  14.79865
-------------------------------------------
```

2.5.7 Correlations

In order to report correlations between variables, we can use the `corr` command:

```
.  corr price mpg
(obs=74)

             |    price      mpg
-------------+------------------
       price |   1.0000
         mpg |  -0.4686   1.0000

. corr price mpg weight length
(obs=74)

             |    price      mpg   weight   length
-------------+---------------------------------
       price |   1.0000
         mpg |  -0.4686   1.0000
      weight |   0.5386  -0.8072   1.0000
      length |   0.4318  -0.7958   0.9460   1.0000
```

2.5.8 Tests for Means

The clasical t-tests for means of variables in the dataset are performed using the command `ttest`. We can e.g. test that the mean price of a car in the dataset is $5,500

```
. ttest price = 5500

One-sample t test
------------------------------------------------------------------------------
Variable |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
---------+--------------------------------------------------------------------
   price |      74    6165.257    342.8719    2949.496    5481.914      6848.6
------------------------------------------------------------------------------
    mean = mean(price)                                            t =   1.9402
Ho: mean = 5500                                 degrees of freedom =       73

   Ha: mean < 5500             Ha: mean != 5500                 Ha: mean > 5500
 Pr(T < t) = 0.9719         Pr(|T| > |t|) = 0.0562          Pr(T > t) = 0.0281
```

or we can test that the mean of the price of domestic and foreign cars is the same:

```
. ttest price, by(foreign)

Two-sample t test with equal variances
------------------------------------------------------------------------------
   Group |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
---------+--------------------------------------------------------------------
Domestic |      52    6072.423    429.4911    3097.104    5210.184    6934.662
 Foreign |      22    6384.682    558.9942    2621.915     5222.19    7547.174
---------+--------------------------------------------------------------------
combined |      74    6165.257    342.8719    2949.496    5481.914      6848.6
---------+--------------------------------------------------------------------
    diff |           -312.2587    754.4488               -1816.225    1191.708
------------------------------------------------------------------------------
    diff = mean(Domestic) - mean(Foreign)                         t =  -0.4139
Ho: diff = 0                                    degrees of freedom =       72

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.3401         Pr(|T| > |t|) = 0.6802          Pr(T > t) = 0.6599
```

2.6 Getting Data out of Stata

Similarly as you would import the dataset, you can export it. You can also do so for only a subset of the data:

```
. export excel auto.xlsx, firstrow(variables) replace // without ext.: xls
file auto.xlsx saved

. export delimited make price mpg using auto.txt, replace // without ext.: csv
file auto.txt saved
```

References

LaLonde, R. (1986). Evaluating the Econometric Evaluations of Training Programs with Experimental Data. *American Economic Review*, *76*(4), 604–20.

Mitchell, M. N. (2010). *Data Management Using Stata: A Practical Handbook*. StataCorp LP.

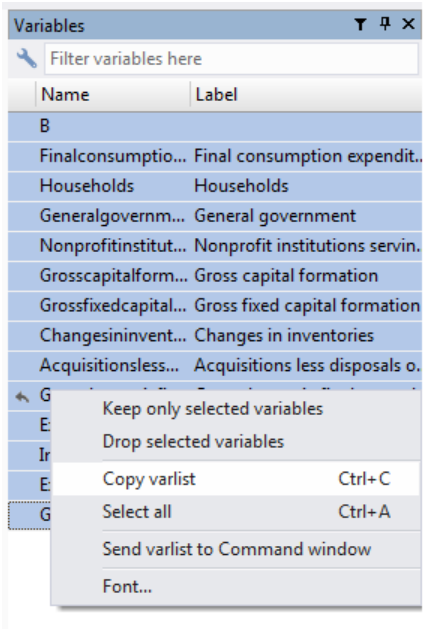
A Appendix: Copying *varlists* to Excel

This quick tutorial provides step-by-step instructions on how to get names of variables into Excel for creating manual code that needs to be applied on several variables. Note that these guidelines were prepared using Microsoft Excel 2010. There may be minor differences while using other versions of the software.

Step 1: Copy the *varlist* including all variables we need to rename

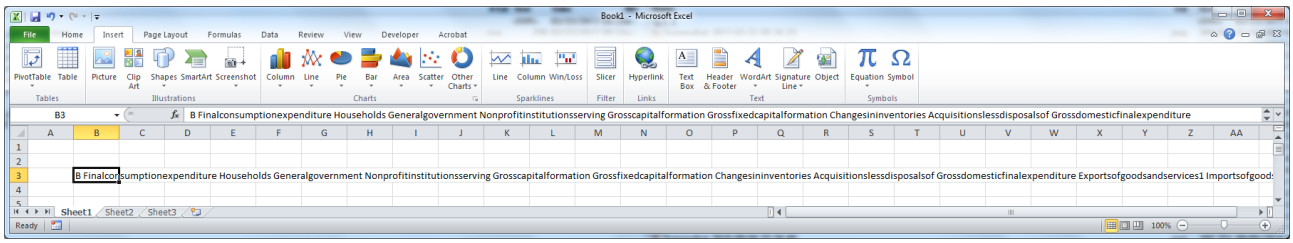
This will copy them as a Stata *varlist*, meaning they will be behind each other in one line. In this case, they will look the following:

```
B Finalconsumptionexpenditure Households Generalgovernment ...
```



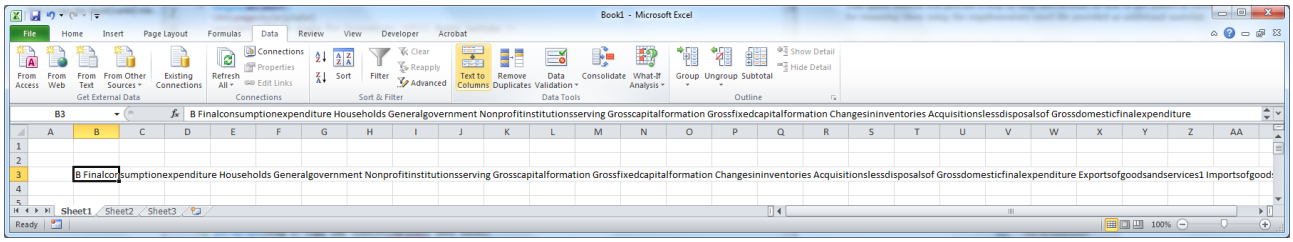
Step 2: Copy the *varlist* into Excel

This will copy it into one cell in Excel.



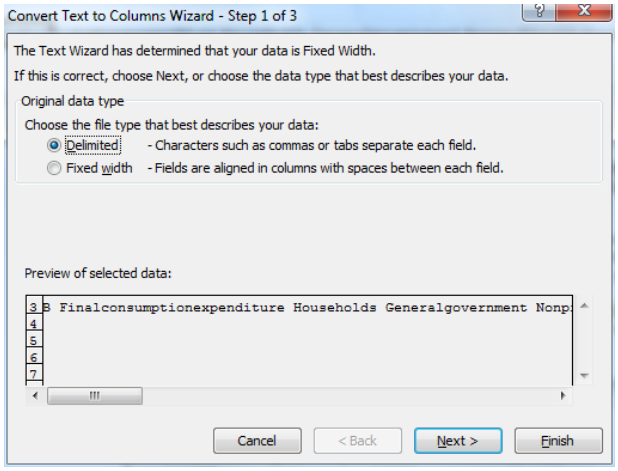
Step 3: Select the cell with the *varlist* and click on "Text to columns".

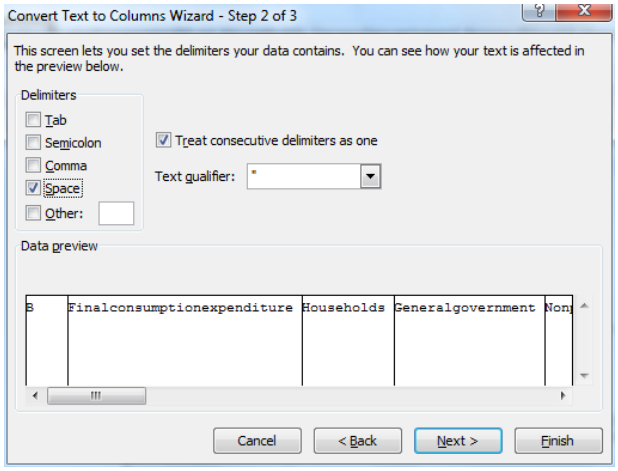
A new window should open.



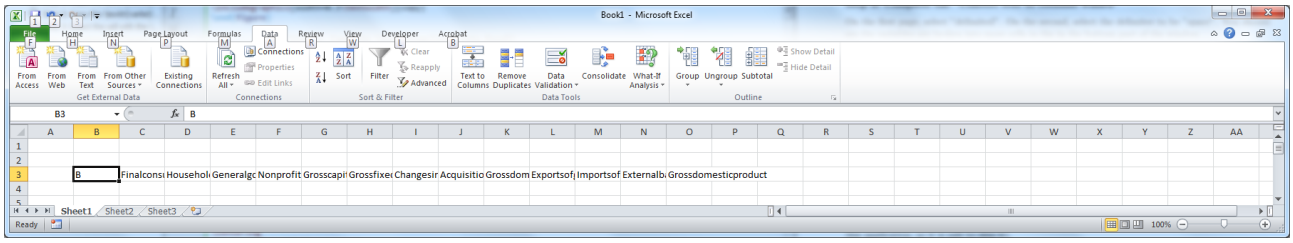
Step 4: Complete the "Convert text to columns wizard"

On the first page, select "delimited". On the second, select the delimiter to be "space". You should see the variable names broken into more cells in the in the bottom part of the window.



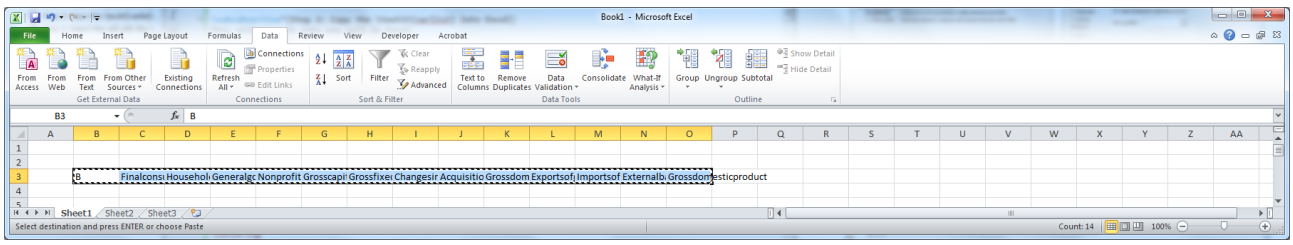


Click "Finish" in the bottom (note that there is also step 3, which is, however, unnecessary for this application, so it is safe to skip it). You should see that the varible names are now in separate cells.



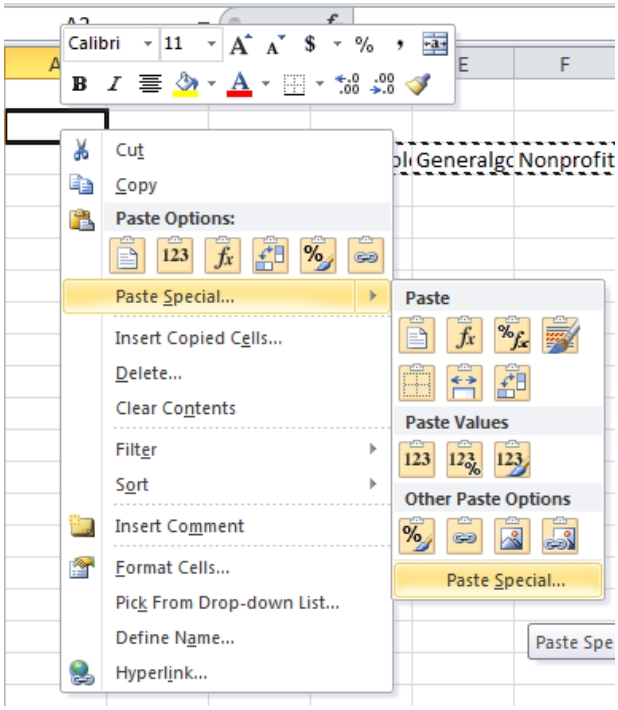
Step 5: Select all the cells with variable names and copy them

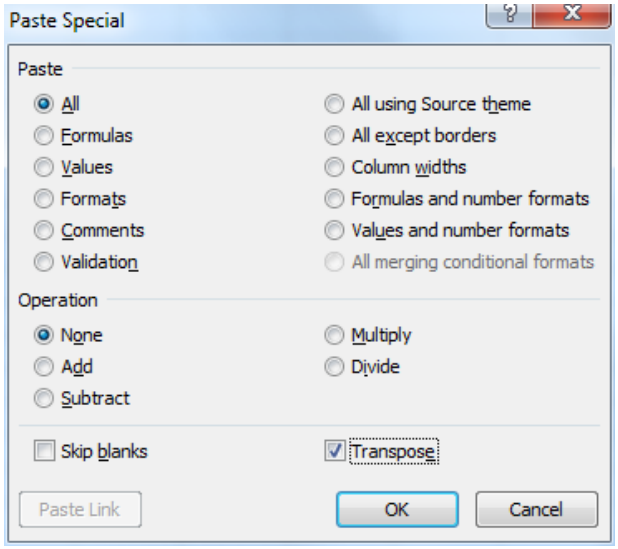
Note that it is not possible to "cut" them.



Step 6: Select the cell where you want the variables and paste their transposed copy

Right click on the cell and select "Paste Special". Do not look at the special shortcuts and select "Paste Special" in the bottom of the second selection box. In the dialog box that opened, select "Transpose" on the bottom right and click OK.





Step 7: Delete the content of the original cells.

We now have an excel file with the variable names in cells below each other.