Data inspection

What's in store?

Loaded data (variables) may come in different **types** that are important when it comes to working with the data.

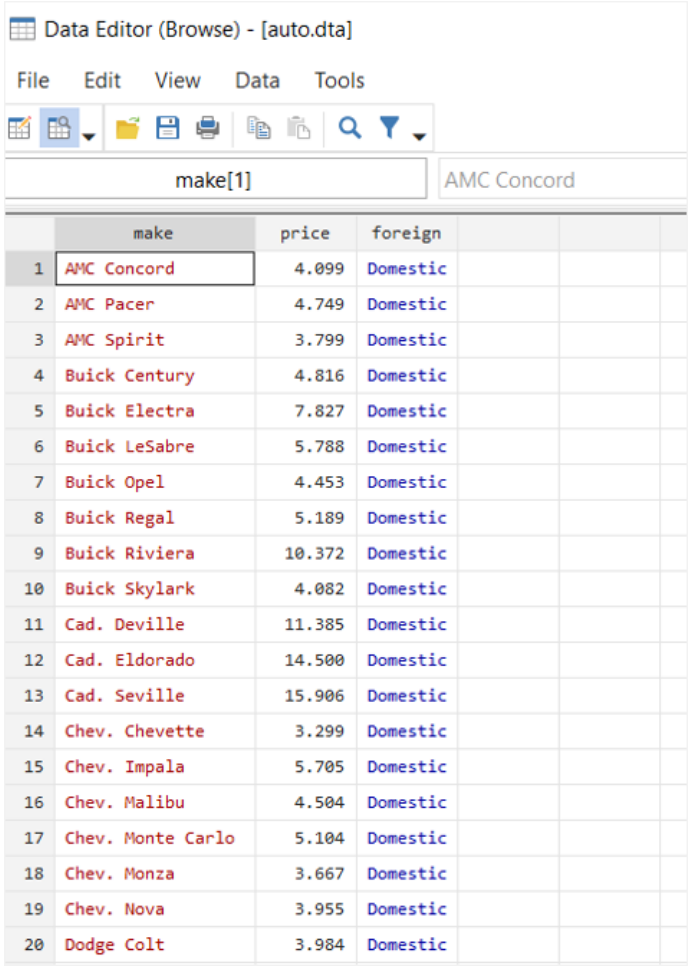
- By far the most common variable type are `numerics` - so numbers.
  - There are `integers`, `doubles` and `floats` amongst others. We don't want to go into too much detail. Just remember that `doubles` and `floats` can have decimal places, `ints` don't.
  - Numerics can be manipulated by the usual mathematical operations: adding, subtracting, taking logs, ...
- The second variable type you will encounter are `Strings`.
  - Strings may be words stored in a variable. To be more precise, you can collect any arbitrary series of characters in a string.
  - Data manipulation works different here since strings don't have numerical values. If you add two strings, Stata will paste them together with no spaces in-between. So `"Peter" + "Pan"` will become `"PeterPan"`.

Value labels

- Remember that Stata can only perform calculations with numeric variables. However, remembering the meaning of a value can be difficult, e.g., if it stands for a very specific category.
- **Example: individual labor force status** may be recorded in a variable with up to six different categories. It wouldn't help you very much to know that someone is in category three, if you don't know what this category is.
- `Value labels` are meta data that assign names to specific values of a variable without changing its numeric type.
- In Stata, value labels are depicted in `blue`.

Browsing the data

- Once the data are loaded, we can view the dataset as a spreadsheet using the command **browse**.
- Black entries are numeric, red entries are strings, and blue entries are numerics with value labels.
- Use **browse** to get an idea of your unique identifiers and eyeball your data for any problematic variables.



	make	price	foreign
1	AMC Concord	4.099	Domestic
2	AMC Pacer	4.749	Domestic
3	AMC Spirit	3.799	Domestic
4	Buick Century	4.816	Domestic
5	Buick Electra	7.827	Domestic
6	Buick LeSabre	5.788	Domestic
7	Buick Opel	4.453	Domestic
8	Buick Regal	5.189	Domestic
9	Buick Riviera	10.372	Domestic
10	Buick Skylark	4.082	Domestic
11	Cad. Deville	11.385	Domestic
12	Cad. Eldorado	14.500	Domestic
13	Cad. Seville	15.906	Domestic
14	Chev. Chevette	3.299	Domestic
15	Chev. Impala	5.705	Domestic
16	Chev. Malibu	4.504	Domestic
17	Chev. Monte Carlo	5.104	Domestic
18	Chev. Monza	3.667	Domestic
19	Chev. Nova	3.955	Domestic
20	Dodge Colt	3.984	Domestic

Describing the data

- To get an overall impression of the data, you can use the command **`describe`**.
- It lists all variables in the data set and provides information on ...
  - ... storage type,
  - ... `value labels`,
  - ... variable labels.
- At the top, you find the number of observations and variables, just like in the Properties Window.

```
. describe

Contains data from https://www.stata-press.com/data/r17/auto.dta
 Observations:            74                  1978 automobile data
    Variables:            12                  13 Apr 2020 17:45
                                              (_dta has notes)
```

Variable name	Storage type	Display format	Value 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 origin**

Sorted by: **foreign**

Listing Observations

- The `list` command displays the variable contents in the Output Window.
- Simply issuing the command `list` will list all observations and variables
  - Not recommended for large data sets.
- Specify variable names to list only those variables.

```
. list make price

     +-------------------------+
     |      make   |   price   |
     |-------------+-----------|
  1. | AMC Concord |   4.099   |
  2. | AMC Pacer   |   4.749   |
  3. | AMC Spirit  |   3.799   |
  4. | Buick Century |  4.816  |
  5. | Buick Electra |  7.827  |
     |-------------+-----------|
  6. | Buick LeSabre |  5.788  |
  7. | Buick Opel  |   4.453   |
  8. | Buick Regal |   5.189   |
  9. | Buick Riviera |  10.372 |
 10. | Buick Skylark |  4.082  |
     |-------------+-----------|
 11. | Cad. Deville |  11.385  |
 12. | Cad. Eldorado |  14.500 |
 13. | Cad. Seville |  15.906  |
 14. | Chev. Chevette |  3.299 |
 15. | Chev. Impala |  5.705   |
```

`in` control of your commands

- You can restrict your commands to a subset of your observations.
- After many commands, you can specify `in` to select observations (rows)
  - `list in` 1/10 displays the first ten rows.
  - `list in` 5/20 displays the rows five to 20.
  - `list in` -10/L displays the last ten rows.
- You can combine `in`'s row restriction with column restrictions.
- Check the `help` file of any command, to check if you can apply `in`.

```
. list make price in -10/L

     +-------------------------+
     |      make         price |
     |-------------------------|
65.  | Renault Le Car    3.895 |
66.  | Subaru            3.798 |
67.  | Toyota Celica     5.899 |
68.  | Toyota Corolla    3.748 |
69.  | Toyota Corona     5.719 |
     |-------------------------|
70.  | VW Dasher         7.140 |
71.  | VW Diesel         5.397 |
72.  | VW Rabbit         4.697 |
73.  | VW Scirocco       6.850 |
74.  | Volvo 260        11.995 |
     +-------------------------+
```

Codebook to inspect variable values

The command `codebook` gives you information on the values of each specified variable.

- For all variables:
  - Number of unique and missing values
- For numeric variables:
  - range, quantiles, means and standard deviation for continuous variables
  - Frequencies and `value labels` for discrete variables
- For `string` variables:
  - examples
  - warnings about leading, trailing & embedded blanks

```
. codebook make price

--------------------------------------------------------------------------------
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.

--------------------------------------------------------------------------------
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
```

Summarizing continuous variables

- The `summarize` command calculates a variable's number of **non-missing** observations, mean, standard deviation, minimum & maximum
- Use the `detail` option to get more statistics such as percentiles, variance, skewness, kurtosis.
- Access these statistics with a call to r()-list (see `help` file).
- You can `summarize` multiple variables at the same time.
- Summarizing dummy (0/1) variables makes sense if you want to know shares in your data.

```
. summarize price foreign

    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
       price |         74    6165,257    2949,496      3291      15906
     foreign |         74    ,2972973    ,4601885         0          1
```

`if` only...

- Using `if` lets you restrict your commands to rows that meet the condition specified following `if`.
- Use logical operators to specify `if`-conditions:
  - `==` to check for equality.
  - `<`, `>`, `<=`, `>=`
  - `!` not
  - `!=` not equal
  - `&` and
  - `|` or
- It is often more useful than `in` since you don't need to know row numbers!

```
. summarize price foreign

    Variable |        Obs        Mean    Std. dev.
-------------+---------------------------------------
       price |         74    6165,257    2949,496
     foreign |         74    ,2972973    ,4601885

. summarize price if foreign==1

    Variable |        Obs        Mean    Std. dev.
-------------+---------------------------------------
       price |         22    6384,682    2621,915
```

Tabulating categorical variables

- The command `tabulate` displays counts of each value of a variable.
- This is particularly useful for variables with a limited number of levels.
- For variables with `value labels`, use the `nolab` option to display underlying numeric values.

```
. tab foreign
```

Car origin	Freq.	Percent	Cum.
Domestic	52	70,27	70,27
Foreign	22	29,73	100,00
Total	74	100,00	

```
. tab foreign, nolabel
```

Car origin	Freq.	Percent	Cum.
0	52	70,27	70,27
1	22	29,73	100,00
Total	74	100,00	

Two-way tabulations

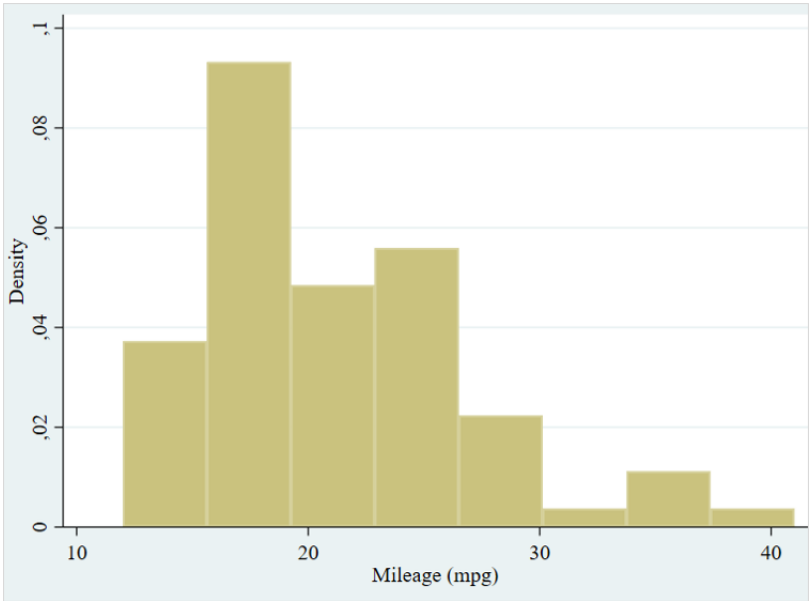
- `tabulate` can also calculate the joint frequencies of two variables.
- Use the `row` and `col` options to display row and column percentages.
- Notice, how the *total* does not equal the number of observations (N=74). We can conclude that there are five observations without a repair record.

```
. tab rep78 foreign
```

Repair record 1978	Car origin		
	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

Visual inspection: histograms

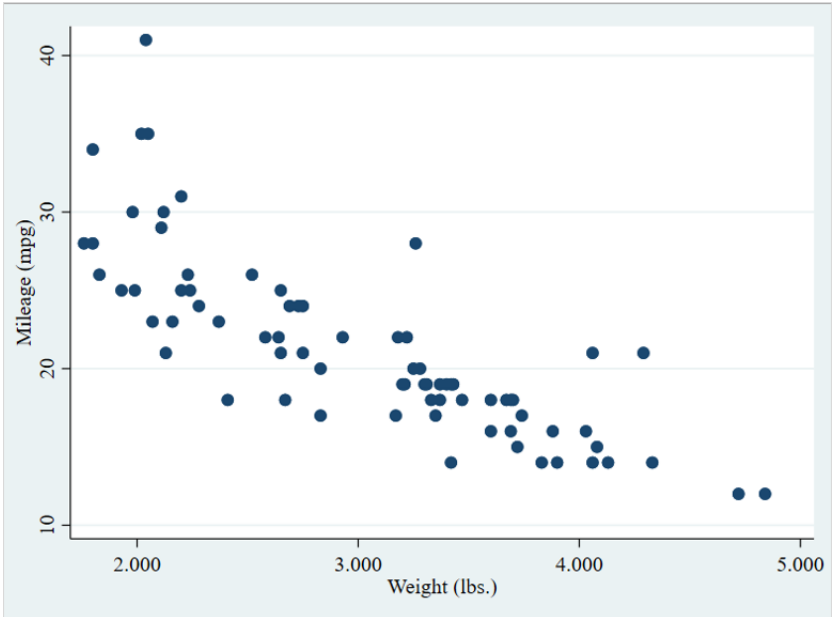
- To get an idea about the distribution of a variable, you can use Stata's graphic options.
- `histogram`s are especially useful for continuous variables or categorical variables with more than just two groups.
- Graphs will open in a separate window. Opening a new graph will close old windows. `histogram` mpg



Visual inspection: scatterplots

- To get an idea about a statistical association between two variables, you can use a `scatter`plot.
- That is especially useful in relatively small data sets.
- `scatter`plots belong to Stata's `twoway`-graphs. As such, they can be combined with other graphs providing a powerful visualization tool.

  `scatter` mpg weight



Other storage types I

- Stata lets you store whatever you want - strings or numbers - in so-called **`macros`**.
  - **`locals`** are deleted immediately after your code is executed.
  - **`globals`** store content as long as Stata is kept open. You may use globals across do-files, programs, etc.
- To create a **`local`** OR **`global`** called MyMacro which contains the elements {element1, element2, 3, 4} type:

  `local/global` MyMacro element1 element2 3 4
- To call the local within your code use `` `MyMacro' ``.
- To call the global, use `$MyMacro` (or `${MyMacro}` to distinguish the global's name from other trailing characters).
- Using macros is useful to avoid retyping long lists of variables - store them in a macro instead!

Other storage types II

- Another important container for your data are **scalars**. Here you can only store numbers.
- Imagine you want to get the mean of a variable. The **summarize** command calculates the mean and stores it in a **return list** (`r()`-list). You can call this list after summarize and store the mean in a scalar:

```
summarize mpg
scalar mpg_mean=r(mean)
```

- To use the scalar in your code, just call `` `=mpg_mean' ``.
- To get a list of all your stored macros and scalars type:

```
macro/scalar list
```

! Remember that locals are deleted after your code is executed. If you call **macro list** separately, you will not see what locals were used.