Slide Set 4
Data input/output and plots

Maria Ptashkina Damian Romero

Barcelona Graduate School of Economics

September 2020

Agenda

Introduction
- Data Types and Structs

Data Input / Output
- Importing and Exporting Data
- Basic Data Analysis
- Exercise 1

Plotting
- Exercise 2

Supplemental Material

Outline

Introduction
- Data Types and Structs

Data Input / Output
- Importing and Exporting Data
- Basic Data Analysis
- Exercise 1

Plotting
- Exercise 2

Supplemental Material

Introduction

Every modern programming language provides various means for storing variables, operating on them, and printing them

In computer science, a *data type* is defined as a set values and operations that can be performed on those values

MATLAB does enforce uniformity: All elements of a given **array** must be of the same type, called the *elementary type* (we will also learn about *'structs'* (structures) which can combine different data types)

To know the data type in MATLAB you can look in the Workspace or call the function `class`

Data Types

- Numeric: the default data type used by MATLAB to store a number is called *double* Other numeric types
- Logical: The relational operators, $==$, $\sim=$, $<$, $>$, $<=$, $>=$ on numeric types return a value of type *logical*
- Strings: a typical use is to store short pieces of text as *character* vectors, such as

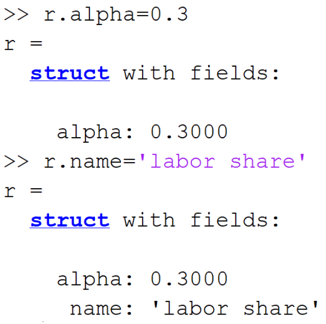
```
>> c = 'Learn Programming'
c =
    'Learn Programming'
>> length(c)
ans =
    17
```

(if interested, search and learn more about strings and *ASCII codes* in programming)

Structs

A *struct* (structure) consists of a set of elements, but unlike an array those elements can be of differing data types

Each element is indexed by user-defined name instead of a numerical index



```
>> r.alpha=0.3
r =
  struct with fields:

    alpha: 0.3000
>> r.name='labor share'
r =
  struct with fields:

    alpha: 0.3000
     name: 'labor share'
```

Outline

Introduction
Data Types and Structs

Data Input / Output
Importing and Exporting Data
Basic Data Analysis
Exercise 1

Plotting
Exercise 2

Supplemental Material

Navigating & File Types

The first step is to make sure that the current directory corresponds to the folder in which you want to save results (or from which you want to import files – but you can alos specify the path)

You can see the current working directory in the desktop or use command `pwd`

It is possible to list the names of the files in the current directory using the command `ls` ("list files")

MATLAB allows to save and load objects or results in several formats (`.mat`, `.txt`, `.dat`, `.csv`, etc.)

- `.mat` files: the file in which MATLAB saves all the variables is called a `MAT-file`

Exporting Data

To save results or objects in `.mat` format use command `save ('filename')` (saves the entire workspace)

To save only the desired variables, you can specify `save ('filename','variable')`

To add variables at the end of a mat-file use `save ('filename', 'variable', -append')`

To save a struct use `save ('result3','-struct','str')`

Importing Data

In order to work with a predefined datasets we can import data in different formats (.xls, .txt, .mat, etc...)

Before importing data it is always a good practice to clear the workspace

- To import a `mat-file`: use command `load` *`filename`*
- To import a `.txt` or a `.dat` file use command `importdata ('filename.ext')` `.ext` stands for file extension (`.dat` or `.txt`)
- To import an Excel file type `database = xlsread ('filename')`
  - If you have more than one sheet, you can select the sheet by specifying `database = xlsread ('filename','Sheet#')`
  - `database = xlsread ('filename','Sheet#','cells range')` will import only a partition of the data
  - Function `xlsread` will only import data in numeric format! To import also text strings and dates: `[NUM, TXT, RAW]= xlsread('filename','Sheet','range')`

Workspace and Array Editor

The Workspace contains all the objects currently loaded and information related to them

Through the Workspace you can create, delete or modify variables

Double-clicking on a variable you can also open it on the *Array Editor* allows to modify variables, eliminate rows and/or columns or single observations and can be really useful at the first stage of importing input data

Basic Data Analysis

To manipulate the database use the strategies we used to manipulate vectors and matrices (eg. square brackets to stack elements, transpose operators, addressing groups of observations or entire variables...)

You probably remember some standard functions from the Programmer's Toolbox, like `min(A)`, `max(A)`, `mean(A)`, `sum(A)`, `sort(A)`

You can also compute variance (`var(A)`), standard deviation (`std(A)`), and covariance (if `A` is a matrix, the function `cov(A)` computes the variance-covariance matrix of the columns of `A`, while `cov(A,B)` gives covariance between vectors `A` and `B`)

`diff(A,dim)` computes first difference of a vector `A`. If A is a matrix, the first difference is computed across dimension `dim`

`diff(A,n,dim)` computes the *n*-th difference of `A`, if `A` is a vector. If `A` is a matrix, the function compute the *n*-th difference across dimension `dim`

P.S. MATLAB is possibly not the most convenient data analysis tool

Exercise 1

1. Load Excel file `'USdata'`
2. Calculate the means, variances and standard deviations of each variable
3. Calculate a covariance matrix
4. Compute first difference of `RGDP`

Exercise 1: Solution

```
% Import data from Excel
clear all
data=xlsread('USdata');

% Calculate the means, variances and standard deviation
mean_data = mean(data);
var_data=var(data);
std_data=std(data);

% Calculate a covariance matrix
cov_data=cov(data);

% Compute first difference of RGDP
RGDP=data(:,1);
GDP_fd=diff(RGDP);
```

Dealing with NaN

If data contain `NaN`, then in order to use the functions above excluding `NaN` use `nanmean`, `nanstd`, `nanvar`, `nanmin`, etc. Function `nancov(A,B)` excludes the elements of A and B corresponding to `NaNs`

```
% substitute some datapoints in INV with NaN:
INVNaN=INVEST;
INVNaN(30:45,:)=NaN;
% compute stats ecluding NaN
avg_nan=nanmean(INVNaN);
std_nan=nanstd(INVNaN);
```

Outline

Introduction
- Data Types and Structs

Data Input / Output
- Importing and Exporting Data
- Basic Data Analysis
- Exercise 1

Plotting
- Exercise 2

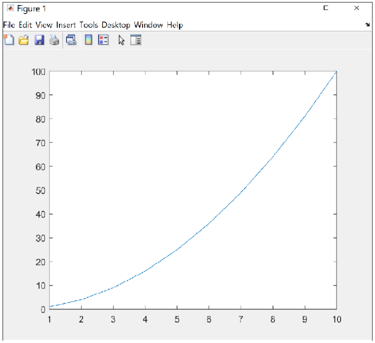
Supplemental Material

The Figure Window

MATLAB has a very powerful plotting facility

If you simply issue a plot command, a figure window will appear

```
>> a = (1:10).^ 2;
>> plot(a)
```



This is the default, and each one appears in a figure window with the label Figure 1. If we wish to leave the previous plot in Figure 1 and plot the next plot in Figure 2, we can do that by issuing the command `figure` before calling plot

Plotting Multiple Series

You can pass multiple vectors to `plot`

```
x1 = 0:0.1:2*pi; y1 = sin(x1);
x2 = pi/2:0.1:3*pi; y2 = cos(x2);

figure
plot(x1,y1,x2,y2)
```

Or use a `hold on` option (allows to plot more series in the same graph)

```
figure
hold on
plot(x1,y1)
plot(x2,y2)
hold off
```

Options

There are many more options to make your graphs shine

- You can change colors and markers, eg. `plot(x1,y1,'r',x2,y2,'k:')` Color and marker options
- `'LineWidth', pt` specifies the width of the line
- `grid on` displays a grid
- `axis tight` restricts axes so to perfectly fill the area of the graph
- or you can specify your own axis range using `axis([ xmin xmax ymin ymax ])`
- `xlabel` and `ylabel` label axis
- `title('graphname','FontSize', pt)` adds a title to the graph
- `legend('name1','name2',...,'position')` adds a legend

Exercise 2

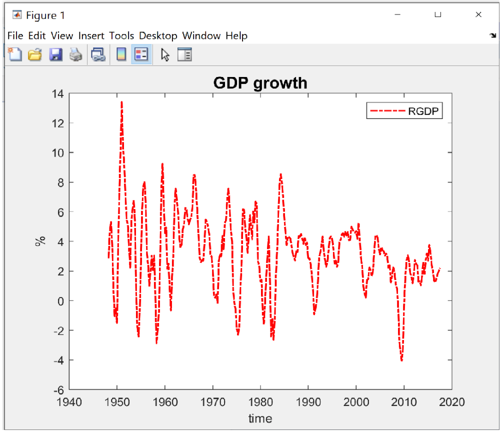
1. Generate a vector of quarterly time data from 1948:Q1 to 2017:Q2
2. Plot the RGDP against this time vector in red dash-dot line with linewidth of 1.5pt. Title your graph 'GDP growth' with 14th font size. Put axis lables and a legend

Hint for 1: time=1948.25:0.25:2017.5;

Exercise 2: Solution

```
time=1948.25:0.25:2017.5;

figure(1)
plot(time,RGDP,'r-.','LineWidth',1.5)
title('GDP growth','FontSize',14)
ylabel('%')
xlabel ('time')
legend('RGDP','NorthWest')
```



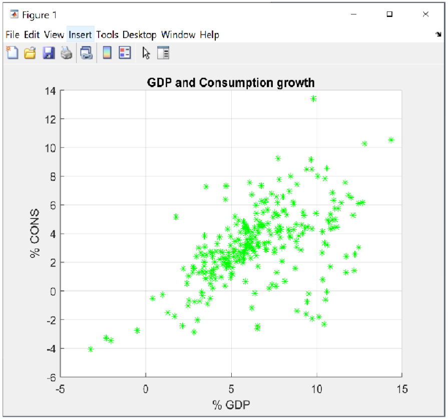
Scatter Plot

`scatter(X,Y,'colour')` produces a scatterplot of X and Y

All the options specified above can be used

```
% Define consumption vector
CONS=data(:,2);

figure
scatter(CONS,RGDP,'g*')
grid on
title('GDP and Consumption growth')
xlabel('% GDP')
ylabel('% CONS')
```

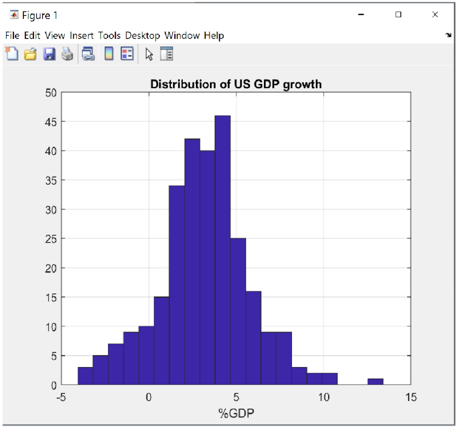


Histograms

`hist(x)`: draws a 10-bin histogram of vector x. If x is a matrix hist works across columns

`hist(x,m)` draws a m-bin histogram of vector x

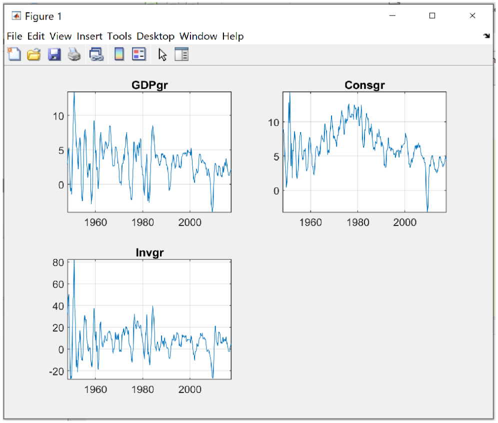
```
figure
hist(RGDP,20)
grid on
title('Distribution of US GDP growth')
xlabel('%GDP')
```



for-loop for Plotting

You can conveniently use a for-loop to make subplots

```
load data
k=size(database,2);
%col={'b-' 'r' 'g-.'};
figure
for i=1:k
    subplot(2,2,i)
    plot(time,database(:,i))
    title(names(i))
    axis tight
    grid on
end
```



Outline

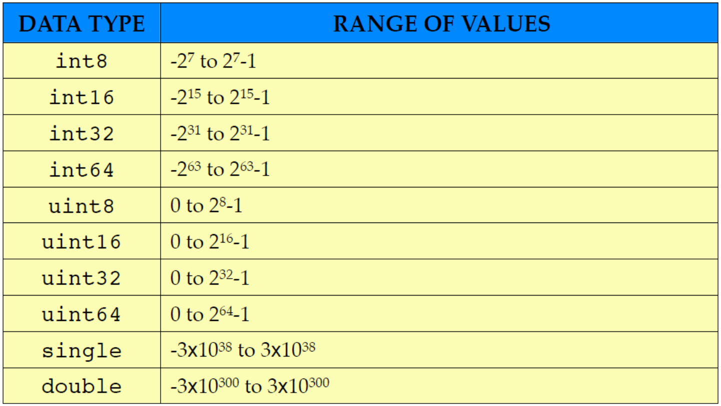
Introduction
Data Types and Structs

Data Input / Output
Importing and Exporting Data
Basic Data Analysis
Exercise 1

Plotting
Exercise 2

Supplemental Material

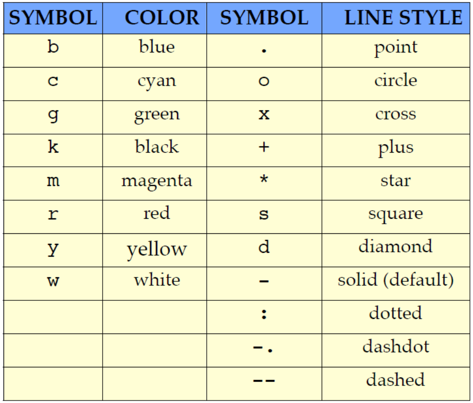
Numeric Data Types



DATA TYPE	RANGE OF VALUES
`int8`	-2^{7} to $2^{7}-1$
`int16`	-2^{15} to $2^{15}-1$
`int32`	-2^{31} to $2^{31}-1$
`int64`	-2^{63} to $2^{63}-1$
`uint8`	0 to $2^{8}-1$
`uint16`	0 to $2^{16}-1$
`uint32`	0 to $2^{32}-1$
`uint64`	0 to $2^{64}-1$
`single`	$-3\text{x}10^{38}$ to $3\text{x}10^{38}$
`double`	$-3\text{x}10^{300}$ to $3\text{x}10^{300}$

Back

Color and Marker Options



SYMBOL	COLOR	SYMBOL	LINE STYLE
b	blue	.	point
c	cyan	o	circle
g	green	x	cross
k	black	+	plus
m	magenta	*	star
r	red	s	square
y	yellow	d	diamond
w	white	-	solid (default)
		:	dotted
		-.	dashdot
		--	dashed

Back