Application of Matlab for Finance
Week 3

Shi, Yining

September 13, 2017

Today's Class

- Input and Output, Reading from and Writing data
- Plot and SubPlot
- CW Q1 Hints: `datenum(.)`

Input/Output

- `xlsread(.) xlswrite(.)`
- `csvread(.) csvwrite(.)`
- `readtable(.) writetable(.)`
- `save(.) load(.)`

xlsread(.)

- `num = xlsread(filename)` reads numeric data from the first sheet of Excel file named `filename`, and store the data into a matrix called `num`
- `num = xlsread(filename, sheet)` reads the specified worksheet
- `num = xlsread(filename, sheet, range)` reads the specified worksheet and range
- `filename`, `sheet` and `range` are strings enclosed in single quotes: `data = xlsread('Stock.xls','Price', 'A1:B100')`
- `xlsread(.)` ignores any outer rows or columns of the spreadsheet that contain no numeric data. Also any nonnumeric cells are assigned as NaN (Not-a-Number element).
- `[num,txt,raw] = xlsread(.)` store the entire worksheet in `raw`, all numeric data in `num` and text data in `txt`

xlswrite(.)

- `xlswrite(filename, M)` writes matrix `M` to the first worksheet in the Excel file `filename`
- `xlswrite(filename, M, sheet)` writes matrix `M` to the specified worksheet `sheet` in the Excel file `filename`
- `xlswrite(filename, M, sheet, range)` writes matrix `M` to a rectangular region specified by `range`
- `filename`, `sheet` and `range` are strings enclosed in single quotes
- Eg. `xlswrite('MyOutput.xls',Returns,'E500','AE:AE50')`

Exercises 1

- Use the `xlsread()` function to read information from the excel file `Stock.xls` sheet `Price` into MATLAB with a matrix called `data`.
- Store the first column of data in a variable called `stock_price_1` and the second column of data in a variable called `stock_price_2`
- Help function `tick2ret()`, and use it to calculate simple return of each stock, naming `stock_return_1` and `stock_return_2` respectively
- Write the returns into a news excel file called `Returns.xls` with two new sheets named `returns_1` and `returns_2` respectively.

Exercises 1


```
data = xlsread('Stock.xls', 'Price');

stock_price_1 = data(:,1);
stock_price_2 = data(:,2);

% calculate return
stock_return_1 = tick2ret(stock_price_1);
stock_return_2 = tick2ret(stock_price_2);

xlswrite('Returns.xls', stock_return_1, 'return_1');
xlswrite('Returns.xls', stock_return_2, 'return_2');
```

csvread(.)

- `M = csvread(filename)` reads a comma-separated value formatted file from `filename` and stored the data in M.
  - There is only one sheet in a .csv file
  - The code requires the .csv file contains only numeric data
- `M = csvread(filename, R1, C1)` reads data from file starting the specific row R1 and column C1. The R1 and C1 arguments are zero based, so that R1=0 and C1=0 specify the first value in the file (Cell 'A1').
- Hint: row (col) = actual file row (column) number -1
- `M = csvread(filename, R1, C1, [R1 C1 R2 C2])` reads only the range specified.
- Eg. `M = csvread('data.csv',0,0,[0,0,6,1])` reads data in range of 'A1' to 'B7' from file data.csv

csvwrite(.)

- `csvwrite(filename,M)` writes matrix M into .csv file `filename`, which is a string enclosed in single quotes.
- `csvwrite(filename, M,R1,C1)` writes matrix M into the specified row R1 and column C1 offset. Note `R1=0` and `C1=0` specify the first value in the file.

- Note1: `xlsread` and `xlswrite` may not work with the Mac/Linux system. Suggestion: save .xls file as .csv → Use `csvread()`.
- Note2: `csvread()` and `csvwrit()` only work with numeric values, and won't not work for csv files with a mix of text and numeric values. Mac users please use the `readtable()` with a 'Format' parameter to specify the data types for each columns.

readtable and writetable

- `T = readtable(filename, 'Format', format_specs)` create a table `T` from column-oriented data such .csv, .txt or .dat.
- `T = readtable('Price.xls','Format', '%D%s%f%d')`
- All data from `'Price.xls'` is stored in table variable `T`
- `'Format'` is the option name argument indicating the following (3rd) input is the format specification
- `'%D%s%f%d'` is the specification value of 'Format'.
  - `'%'` stands a column indicator: 4`'%'`= 4 columns in our data file.
  - 1st column has datetime object (`'%D'`)
  - 2nd column is string (`'%s'`)
  - 3rd column is floating number (`'%f'`)
  - 4th column is integer format (`'%d'`)
- `writetable(your_table, filename)` saves the Matlab table to an external data file.

save and load

- `save('my_dataset.mat', var1, var2, ...)` saves selected variables (var1, var2, ...) in the current workspace to the matfile named `'my_dataset.mat'`.
- `load('my_dataset.mat', var1, var2, ...)` loads variables (var1, var2, ...) from datafile `'my_dataset.mat'` into the current workspace.
- matfile is Matlab's proprietary data structure and it is designed to work with Matlab only. Other programming language may have difficulties to read data from matfile, so the portability of matfile is quite limited.

Exercises 2.a

- Observe the data format of file `AAPL.csv`, read the data using `csvread()` function into matrix `aapl_csv`

- Comment on why the code does not work, what you should do to fix the issue?
  - still use `csvread()`
  - use `xlsread()` to read both numeric data into `aapl_num` and textual data into `aapl_txt`

Exercises 2.a


```
1  aapl_csv = csvread('AAPL.csv');
2  % Above code does not work because AAPL.csv contains string,
3  % date and number. csvread can only read numeric data format
4
5  % Read only numeric data from cell B2
6  % To refer cell in csvread, row=(B)2-1=1, col=2-1=1
7  aapl_csv = csvread('AAPL.csv',1,1);
8
9  % Read with xlsread store num in aapl_xls and string in txt
10 [aapl_num, aapl_txt] = xlsread('AAPL.csv');
```

Exercises 2.b

- Read `AAPL.csv` using `readtable()` function into table name `aapl`
  - The format of data in `AAPL.csv` is Datetime, float, float, float, float, integer, float (specification is `'%D%f%f%f%f%d%f'`)
- Calculate simple return by using AAPL's adjusted close price.
  - Use the period (`.`) to refer sequences in a table: `aapl.AdjClose`

- Store the return into the aapl table under column name `Return` (Ensure the size of returns matches with length of the table)
- Save the updated table to a new csv file named `output_AAPL.csv`.

- Save the table as `AAPL.mat` matfile. Clear the current workspace, and then load the dataset back to our workspace using `load()`.

Exercises 2b


```
aapl = readtable('AAPL.csv','Format', '%D%f%f%f%f%d%f');
aapl_ret = tick2ret(aapl.AdjClose);

% match the return length with the table with the first ...
    day return = 0
ret = [0; aapl_ret];
aapl.Return = ret;
writetable(aapl, 'output_AAPL.csv');

% save and load
save('AAPL.mat', 'aapl')
clear all
load('AAPL.mat')
```

Summary

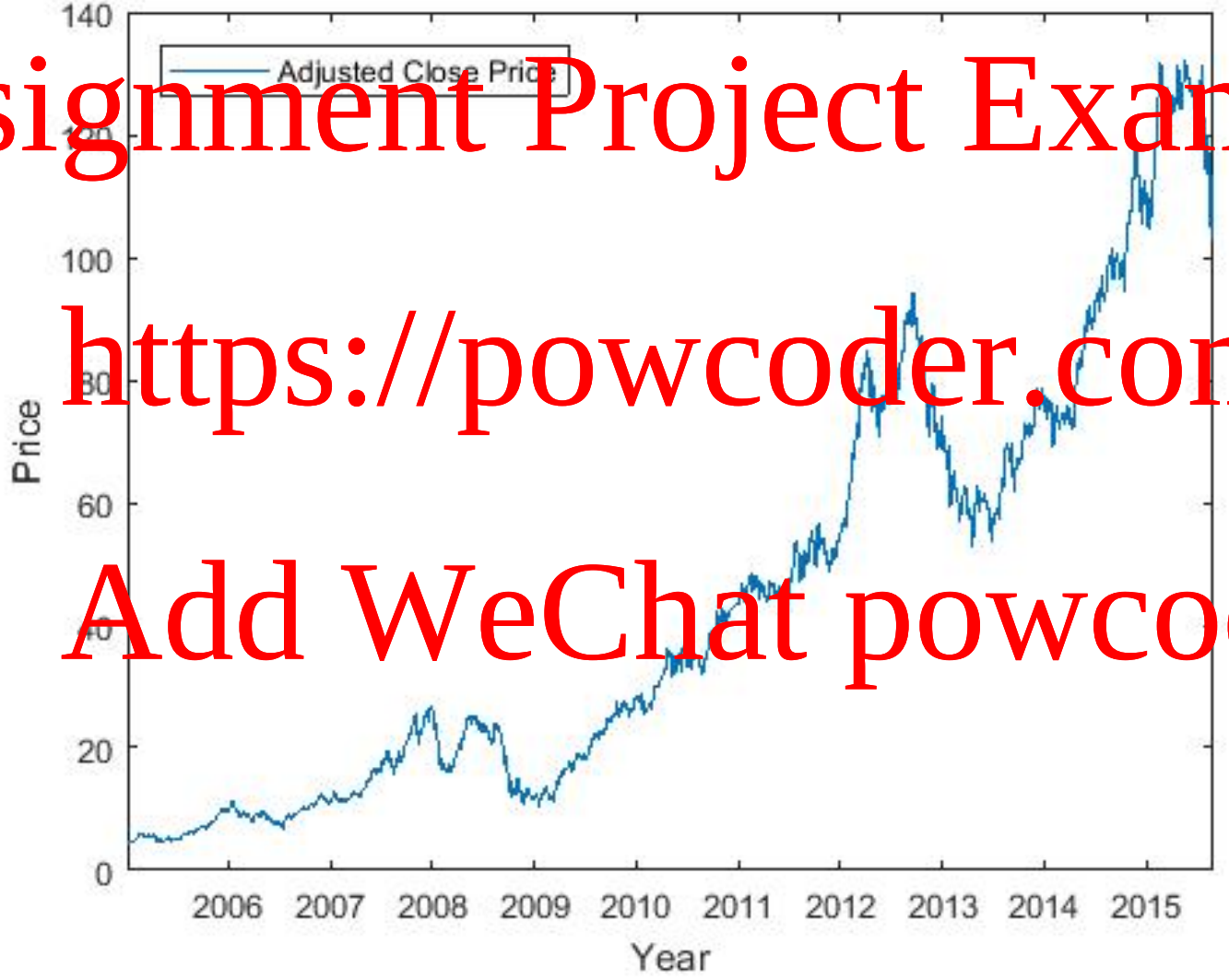
- `xlsread` can read csv file, but `csvread` cannot read excel file.
- When all data are numbers, `csvread` is faster and preferred.
- When the data contain multiple formats such as dates, number and string, `xlsread` and `readtable` allow flexibility.

Simple Time Series Plot

- Create a simple time-series plot of the adjusted close price of AAPL
- Add relevant title, label and legend.


```
figure % create figure environment
plot(aapl.Date, aapl.AdjClose)
legend('Adjusted Close Price', 'Location','NorthWest')
ylabel('Price')
xlabel('Year')
```

Simple Time Series Plot



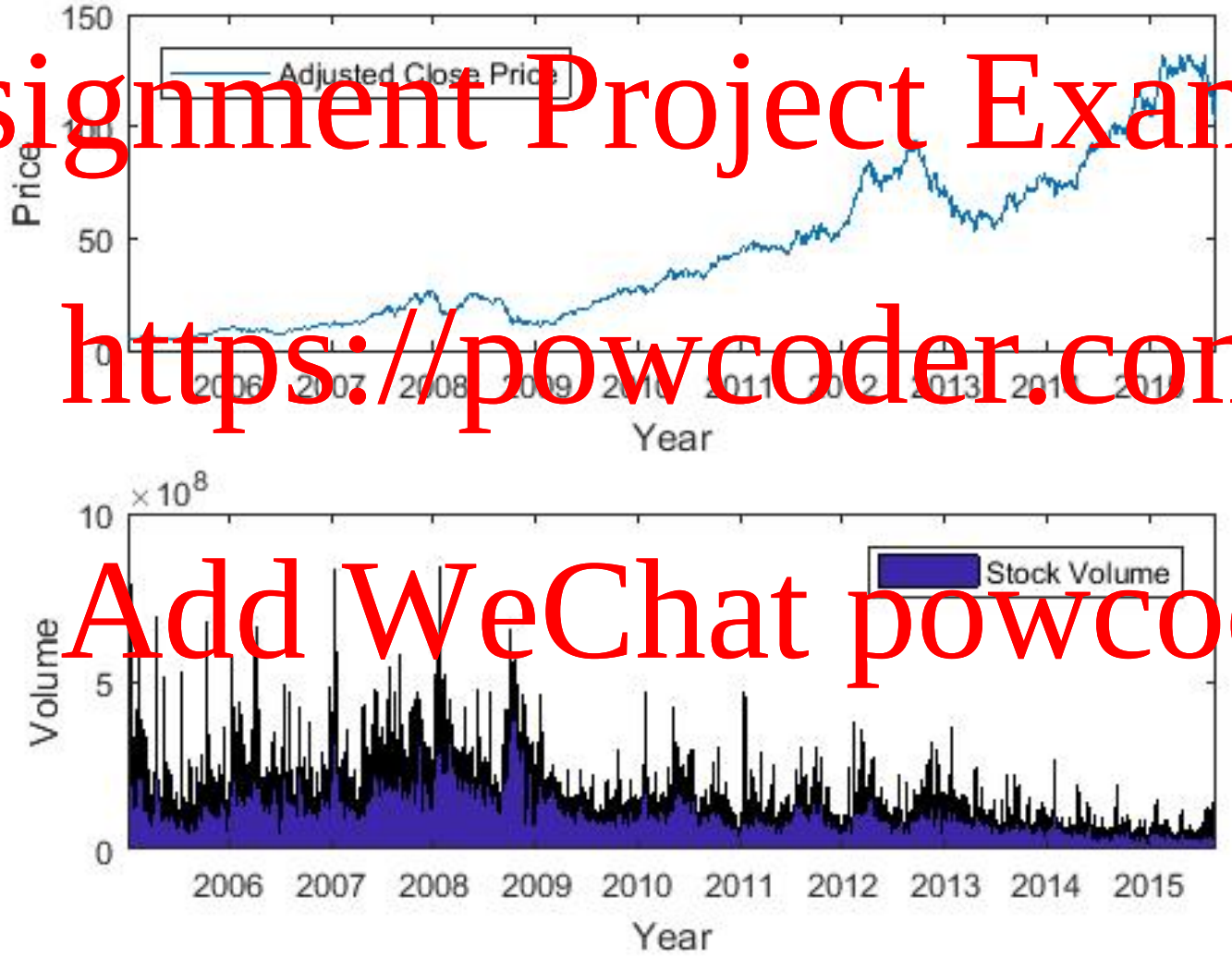
SubPlot

- Now create a figure with 2 sub-plot on AAPL
  - The top panel plots the time-series of the adjusted close price
  - The lower panel is a area plot of the stock volume.


```
1  subplot(2,1,1)
2  plot(aapl.Date, aapl.AdjClose);
3  legend('Adjusted Close Price', 'Location','NorthWest');
4  ylabel('Price');
5  xlabel('Year');
6
7  subplot(2,1,2)
8  area(aapl.Date, aapl.Volume);
9  legend('Stock Volume', 'Location','NorthEast');
10 ylabel('Volume')
11 xlabel('Year');
```


- `subplot(m,n,p)` create a figure with various plots
  - The actual plot codes of plotting comes after `subplot(m,n,p)`
  - `m,n` define the plots layout structure: `m` rows `n` column panels.
  - `p` defines which sub-plot is defined in the following codes.

SubPlot



Hint: CW Q1-How to separate the data

- Option 1: find the index of the target cut-off date manually and use it for the coding
- Option 2: with `readtable(.)`


```
%% Hints: CW Q1
% if data = readtable(.) with DD for dates format
% e.g., exercise 2 b
cut = find(aapl.Date == '18/01/2005'); % find the ...
    cut-off date
data_is = aapl(1:cut,:); % read in-sample
data_oos = aapl(cut+1:end,:); % read out-sample
```


- `cut` is the index number for the target cut-off date;
- `aapl(1:cut,:)` reads row 1 up to row `cut` as in-sample data;
- `aapl(cut+1:end,:)` reads row `cut+1` up to the `end` row as out-of-sample data.

Hint: CW Q1-How to separate the data

- Option 3: with `xlsread(.)`, `datenum(.)` and `datestr(.)`


```
% if [data, txt] = xlsread(.). e.g., exercise 1
t_string = aapl_txt(2:end,1); % read date string
% convert date string to date number
t_num = datenum(t_string, 'dd/mm/yyyy');
target_date = datenum('01/01/2005','dd/mm/yyyy');
cut = find(t_num == target_date);
data_is = aapl_num(1:cut,:);
data_oos = aapl_num(cut+1:end,:);
```


- `datenum(.)` converts date strings into numerical format, which is necessary for time series analysis and figure plot.
- The second input `'dd/mm/yyyy'` defines the date format in the data.
- `target_date` is the numerical date format for the cut-off date.

TakeAway

- Input & Output: You can start with the coursework question 1
- Figures: plot, subplot