

Matlab Course 2015-2016

Sachapon Tungsong

Queen Mary University of London

s.tungsong@qmul.ac.uk

July 19, 2016

OVERVIEW OF LECTURE 5

- Stationary time series: Analysis and models
 - ① ACF/PACF functions
 - ② Stationarity tests
 - ③ ARMA models
- Volatility modeling (ARCH/GARCH)
- Appendix: Useful functions for analyzing time series data

AUTOCORRELATION FUNCTION (ACF)

- The ACF is useful in identifying the order of an MA model.
- A time series y_t has a lag- l ACF equal to 0 but lag- q ACF NOT equal to 0. If $l > q$, then y_t follows an MA(q) model.
- In Matlab, type `corr(y,numLags)`

PARTIAL AUTOCORRELATION FUNCTION (PACF)

- The PACF is useful in determining the order of an AR model.

$$y_t = \phi_{1,1}y_{t-1} + e_{1t}, \quad (1)$$

$$y_t = \phi_{1,2}y_{t-1} + \phi_{2,2}y_{t-2} + e_{2t}, \quad (2)$$

$$y_t = \phi_{1,3}y_{t-1} + \phi_{2,3}y_{t-2} + \phi_{3,3}y_{t-3} + e_{3t} \quad (3)$$

\vdots

- The estimate $\hat{\phi}_{2,2}$ in (2) is the lag-2 PACF of y_t , the estimate $\hat{\phi}_{3,3}$ in (3) is the lag-3 PACF of y_t and so on.

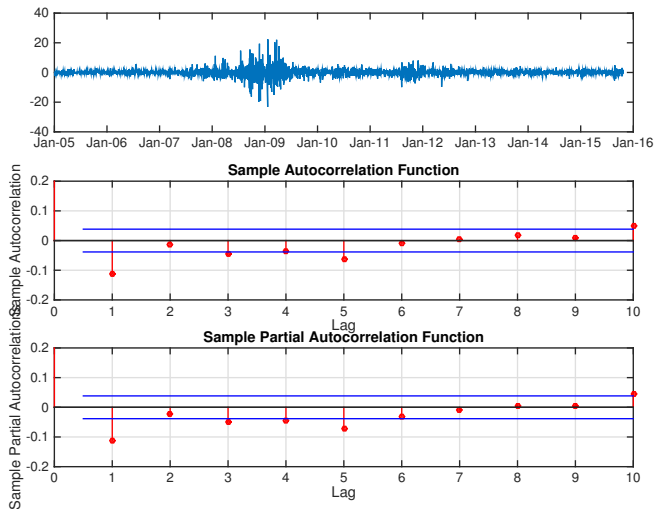
PARTIAL AUTOCORRELATION FUNCTION (PACF)

- The lag-2 PACF $\hat{\phi}_{2,2}$ shows the added contribution of y_{t-2} to y_t over the AR(1) model.
- Therefore, for an AR(p) model, the lag-p sample PACF should not be zero but $\hat{\phi}_{j,j}$ should be close to zero for all $j > p$.
- In Matlab, type `parcorr(y,numLags)`

ACF/PACF of JPM

```
1  %% Plot time series of JPM and corresponding dates in subplot 1
2  %% Plot ACF/PACF of JPM in subplots 2, 3
3  figure
4  subplot(3,1,1)
5  plot(commonDate,jpm)
6  datetick('x','mmm-yy')
7  subplot(3,1,2)
8  autocorr(jpm,10)
9  ylim([-0.2 0.2]) % Set limits for y axis
10 subplot(3,1,3)
11 parcorr(jpm,10)
12 ylim([-0.2 0.2])
```

Plot of JPM returns and ACF/PACF



STATIONARITY TEST

Augmented Dickey-Fuller test: Test whether a unit root is present in an autoregressive model.

- In simple AR(1) model $y_t = \phi y_{t-1} + e_t$, a unit root is present if $\phi = 1$, i.e., the time series is non-stationary.
- In Matlab, `[h, pVal] = adftest(y)`
- The null hypothesis of the DF test is that there is a unit root, i.e., $\phi = 1$. If the result $h = 0$, it means we fail to reject the null hypothesis of unit root. On the contrary, if the result $h = 1$, it means we have sufficient evidence to reject the null hypothesis and conclude that the time series is stationary.

ARMA MODELS

In Matlab, the command `model = arima(p,D,q)` creates model objects for stationary (if $D = 0$) or unit root non-stationary linear time series model (if $D > 0$).

p is the autoregressive (AR) order,

D is the order of integration, e.g., if $D = 1$, there's unit root in the time series,

q is the moving average (MA) order.

ARMA MODELS

- In some applications, the AR or MA models become cumbersome because we might need a high-order model with many parameters to adequately describe the dynamic structure of the data.
- ARMA models are thus introduced for parsimony as ARMA combines AR and MA models into a compact form so that the number of parameters are kept small.
- Most of the times we end up using ARMA(1,1)
- For return series in finance, the chance of using ARMA models is low (Tsay, 2005) but for volatility modeling, ARMA is highly relevant (GARCH is regarded as an ARMA model).

ARMA MODELS

```
14 %% ADF test
```

```
15 [h,pVal] = adftest(jpm)
```

```
16  
17 %% Fitting 5 models on JPM and compare
```

```
18 model = {};
```

```
19 model{1} = arima(1,0,0); % AR(1)
```

```
20 model{2} = arima(5,0,0); % AR(5)
```

```
21 model{3} = arima(0,0,1); % MA(1)
```

```
22 model{4} = arima(0,0,5); % MA(5)
```

```
23 model{5} = arima(1,0,1); % ARMA(1,1)
```

```
24  
25 for i = 1:5
```

```
26     [estModel{i},estParamCov{i},logL{i}] = estimate(model{i},jpm);
```

```
27 end
```

VOLATILITY MODELING

- The error term in a regression model (including ARIMA) is assumed to have constant volatility, BUT, this is not always observed empirically. Volatility of returns changes over time.
- Volatility clustering, i.e., large changes followed by large changes, of either sign, and small changes followed by small changes, are commonly observed in practice.
- Proxies for volatility, e.g., the absolute returns $|r_t|$ or their squares $|r_t|^2$ display a positive, significant and slowly decaying autocorrelation function: $\text{corr}(|r_t|, |r_{t+s}|) > 0$ for s ranging from a few minutes to several weeks.

ARCH/GARCH MODELS

- The ARCH (Engle, 1982) and GARCH (Bollerslev, 1986) models aim to more accurately describe the phenomenon of volatility clustering and related effects such as kurtosis (heavy tails).
- The main idea behind these models is that volatility is dependent upon past realizations of the return process and related volatility process.

- ARCH(q)

$$y_t = \mu + e_t \quad (4)$$

$$e_t = \sigma_t z_t, \quad (5)$$

$$\sigma_t^2 = c + a_1 e_{t-1}^2 + \dots + a_q e_{t-q}^2 \quad (6)$$

- The return equation (4) describes the return y_t as a function of a constant μ and error term e_t .
- z_t is a white noise with mean 0 and variance 1.
 $z_t \sim WN(0, 1)$

- GARCH(p,q)

$$y_t = \mu + e_t \quad (7)$$

$$e_t = \sigma_t z_t \quad (8)$$

$$\sigma_t^2 = c + a_1 e_{t-1}^2 + \dots + a_q e_{t-q}^2 + b_1 \sigma_{t-1}^2 + \dots + b_p \sigma_{t-p}^2 \quad (9)$$

- Extension of ARCH to include autoregressive terms of σ_t^2 .
- ARCH/GARCH models are stationary if the process σ_t^2 is stationary.

- Extends the return equation to allow for autoregression in returns.
- Example: AR(1) + GARCH(1,1)

$$y_t = y_{t-1} + e_t \quad (10)$$

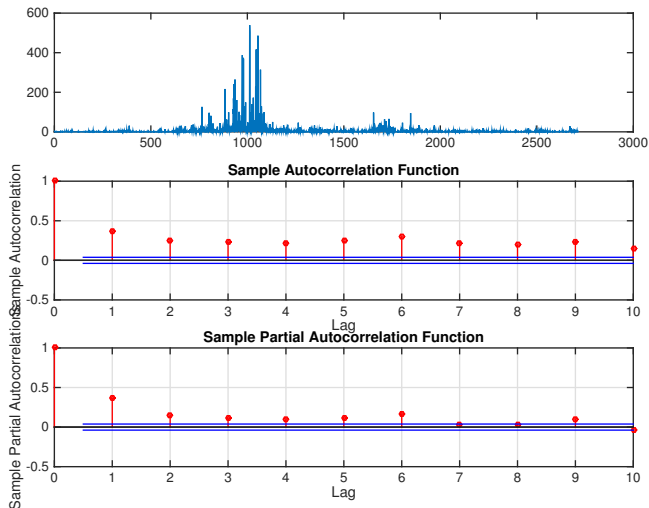
$$e_t = \sigma_t z_t \quad (11)$$

$$\sigma_t^2 = c + a_1 e_{t-1}^2 + b_1 \sigma_{t-1}^2 \quad (12)$$

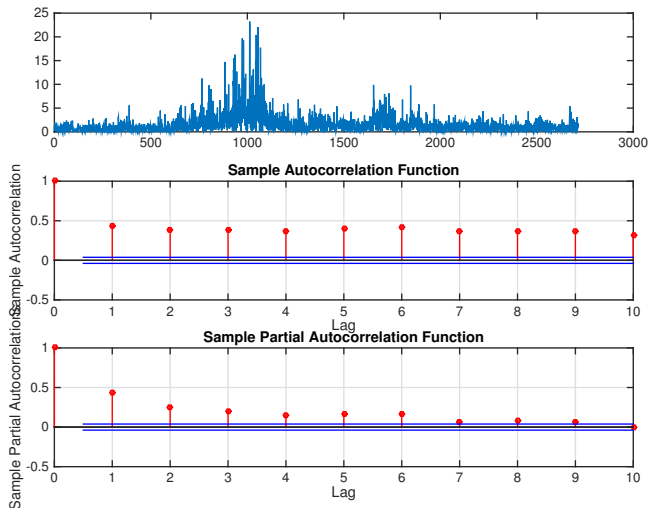
VISUAL INSPECTION OF RETURNS

```
1  %% Visual check JPM for cond. heteroskedasticity
2  -  figure
3  -  subplot(3,1,1)
4  -  plot(jpm.^2)
5  -  subplot(3,1,2)
6  -  autocorr(jpm.^2,10)
7  -  subplot(3,1,3)
8  -  parcorr(jpm.^2,10)
9
10 -  figure
11 -  subplot(3,1,1)
12 -  plot(abs(jpm))
13 -  subplot(3,1,2)
14 -  autocorr(abs(jpm),10)
15 -  subplot(3,1,3)
16 -  parcorr(abs(jpm),10)
```

PLOT OF SQUARED RETURNS



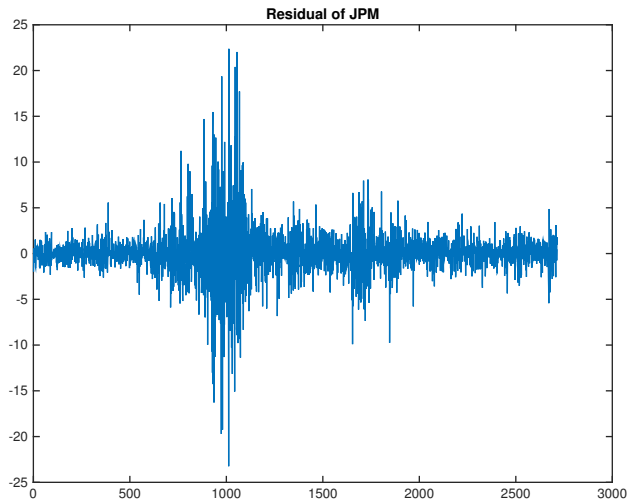
PLOT OF ABSOLUTE VALUES OF RETURNS



TEST FOR ARCH EFFECT IN RESIDUALS

```
19 %% Analyze the residuals
20 % Demean JPM or
21 % Regress JPM on a constant and obtain the residuals
22 - meanjpm = mean(jpm);
23 - residualjpm = jpm - meanjpm;
24
25 % Visual inspection of ARCH effects in the residuals
26 - figure
27 - plot(residualjpm)
28 - title('Residual of JPM')
29
30 % Test for ARCH effects in the residuals
31 % h = 0: no ARCH effect, h = 1: There is ARCH effect
32 - [h,pvalue,stat,cvalue] = archtest(residualjpm,'Lags',1,'Alpha',0.05);
33
```

JPM RESIDUAL PLOT



FITTING ARCH/GARCH MODELS

```
34 %% Fit various GARCH models onto the real data
35 % (1) GARCH(p,q)
36 - for i = 1:3
37 -     mdl{i} = garch(i,1);
38 -     [estModel{i},estParamCov{i},logL{i}] = ...
39 -         estimate(mdl{i},residualjpm);
40 - end
41
42 % (2) AR(1) + GARCH(1,1) Gaussian innovation
43 - mdl{4} = arima('ARLags',1,'Variance',garch(1,1));
44 - [estModel{4},estParamCov{4},logL{4}] = estimate(mdl{4},jpm);
45
46 % (3) AR(1) + GARCH(1,1) Student's t innovation
47 - mdl{5} = arima('ARLags',1,'Variance',garch(1,1));
48 - mdl{5}.Distribution = 't';
49 - [estModel{5},estParamCov{5},logL{5}] = estimate(mdl{5},jpm);
50 -
```

WHICH MODEL HAS THE BEST FIT?

- Log-likelihood.

The model with the highest log-likelihood has the best fit.

- Information criteria

- 1 Akaike Information Criterion (AIC).

The model with the lowest AIC has the best fit.

- 2 Bayesian Information Criterion (BIC).

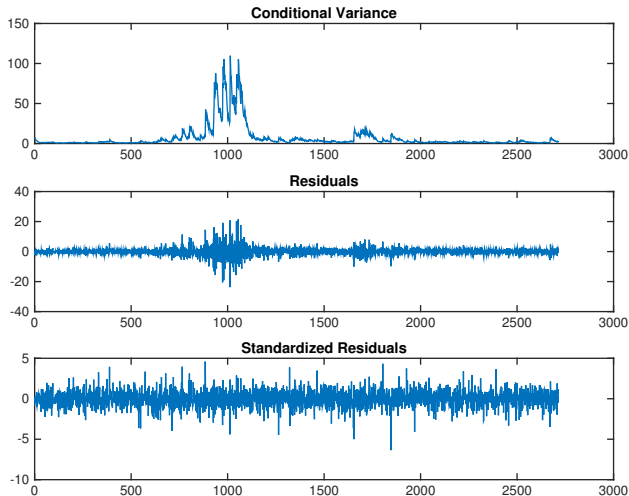
The model with the lowest BIC has the best fit.

```
52 %% Model comparison
53 - numObs = 2721;
54 - for i = 1:5
55 -     numParam(i) = size(estParamCov{i},1);
56 -     [aic(i),bic(i)] = aicbic(logL{i},numParam(i),numObs);
57 - end
```


INFER CONDITIONAL VARIANCES AND RESIDUALS FROM FITTED GARCH MODEL

```
60 %% Infer the conditional variances and residuals
61 - [res,v] = infer(estModel{5},jpm);
62 % Residuals and cond. variances from fitting Model5 on JPM
63
64 - figure
65 - subplot(3,1,1)
66 - plot(v)
67 - title('Conditional Variance')
68
69 - subplot(3,1,2)
70 - plot(res)
71 - title('Residuals')
72
73 - subplot(3,1,3)
74 - plot(res./sqrt(v))
75 - title('Standardized Residuals')
--
```

INFER CONDITIONAL VARIANCES AND RESIDUALS FROM FITTED GARCH MODEL



USEFUL FUNCTIONS FOR ANALYZING TIME SERIES DATA

USEFUL FUNCTIONS 1

`max, min, median, mode, std, var`

Respectively, the functions evaluate the maximum, minimum, median, mode, standard deviation, and variance of the data.

USEFUL FUNCTIONS 2

```
R = corrcoef(X);
```

Returns a matrix R of correlation coefficients calculated from an input matrix X.

```
R = corrcoef(X,Y);
```

Returns a matrix R of correlation coefficients calculated from column vectors X and Y.

```
[R,P,RLO,RUP = corrcoef(...);
```

Also returns matrices of RLO and RUP, containing lower and upper bounds for a 95 confidence interval.

```
cov(X); cov(X,Y);
```

Returns covariance matrices.

USEFUL FUNCTIONS 3

`hist(X)`

Bins the elements in vector `X` into 10 equally spaced containers and returns the number of elements in each container as a row vector.

`hist(X,nbins)`

`nbins` is a scalar, representing number of bins to be putting all data points into.

USEFUL FUNCTION 4: MOVING AVERAGE

```
output = tsmovavg(tsobj, 's', lag, dim);
```

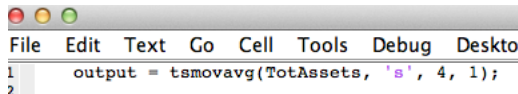
tsobj is the financial time series object.

's' is for simple. Alternatives are 'e' for exponential, etc.

lag number of previous data points used in calculation.

dim specifies dimension when input is a vector or matrix.

dim = 2 for row-oriented matrix: each row is a variable, and each column is an observation.



A screenshot of a code editor window with a macOS-style title bar (red, yellow, green buttons). The menu bar includes File, Edit, Text, Go, Cell, Tools, Debug, and Desktop. The code editor shows two lines of code: line 1: `output = tsmovavg(TotAssets, 's', 4, 1);` and line 2: (empty). The variable 's' in the code is highlighted in purple.

MOVING AVERAGES

```
6 WindowSize1 = 250;
7 WindowSize2 = 700;
8 sMoveAvg1 = tsmovavg(SWFtoUSD, 's', WindowSize1, 1);
9 sMoveAvg2 = tsmovavg(SWFtoUSD, 's', WindowSize2, 1);
10 eMoveAvg = tsmovavg(SWFtoUSD, 'e', WindowSize2, 1);
11
12
13 plot(date, sMoveAvg1); hold on;
14 plot(date, sMoveAvg2); hold on;
15 plot(date, eMoveAvg); hold on;
16 plot(date, SWFtoUSD);
17 legend('Simple Moving Average 1', 'Simple Moving Average 2', ...
18        'Exponential Moving Average', 'Swiss Franc to USD');
19 datetick('x', 12)
```


MOVING AVERAGES



USEFUL FUNCTION 5: DIFFERENCING

```
Y = diff(X);
```

Calculates the first differences between adjacent elements of X.

```
Y = diff(X,n);
```

Calculates the nth differences for elements of X.

```
Y = diff(X,n,dim);
```

Calculates the nth differences along the dimension specified by a scalar dim.
