TABLE OF CONTENTS

Part I. Full Sample

```
clearvars
clc
```

Downloading Data

```
Data = readtimetable('Data_MT.xlsx', 'Sheet', 'Forecasting data','Range','A237:L584');
Data.Properties.VariableNames = {'D' 'S' 'O' 'RD' 'N', 'IO', 'F', 'E', 'B', 'C', 'P'};
Data = Data(timerange('1998-08-31','2021-01-01'),:);
Data.F =[]; % Delete F

variables       = Data.Properties.VariableNames; % Define name of variables
delta_variables = strcat({'delta '}, variables);
j               = size(Data,2);  % number of variables

 data = table2array(Data);
ldata = diff(data,1,1);     % First Difference
```

In-Sample: August 31, 1998 to December 31, 2018

```
 data01 = Data(timerange('1998-08-31','2019-01-01'),:);
 time   = data01.Time;
 data01 = table2array(data01);
ldata01 = diff(data01,1,1);
```

Out-of-Sample: January 31, 2019 to December 31, 2020

```
 data19  = Data(timerange('2019-01-31','2021-01-01'), :);
 time19  = data19.Time;
 data19  = table2array(data19);
ldata19  = diff(data19,1,1);
```

Statistics

Descriptive statistics

```
Descriptive_Statistics_levels          = table();
Descriptive_Statistics_levels.mean     = mean(data,   1)';
Descriptive_Statistics_levels.std      = std(data,    1)';
Descriptive_Statistics_levels.skewness = skewness(data,[],1)';
Descriptive_Statistics_levels.kurtosis = kurtosis(data,[], 1)';
Descriptive_Statistics_levels.minimum  = min(data)';
Descriptive_Statistics_levels.maximum  = max(data)';
Descriptive_Statistics_levels.Properties.RowNames = variables

Descriptive_Statistics_1st_difference          = table();
Descriptive_Statistics_1st_difference.mean     = mean(ldata,   1)';
Descriptive_Statistics_1st_difference.std      = std(ldata,    1)';
Descriptive_Statistics_1st_difference.skewness = skewness(ldata,[], 1)';
Descriptive_Statistics_1st_difference.kurtosis = kurtosis(ldata,[], 1)';
Descriptive_Statistics_1st_difference.minimum  = min(ldata)';
Descriptive_Statistics_1st_difference.maximum  = max(ldata)';
Descriptive_Statistics_1st_difference.Properties.RowNames = delta_variables
```

Multicollinearity

Correlation matrix

```
corr_levels = corr(data);
corr_levels = table(corr_levels);
corr_levels = splitvars(corr_levels);
corr_levels.Properties.RowNames      = variables;
corr_levels.Properties.VariableNames = variables

corr_1st_difference = corr(ldata);
corr_1st_difference = table(corr_1st_difference);
corr_1st_difference = splitvars(corr_1st_difference);
corr_1st_difference.Properties.RowNames      = delta_variables;
corr_1st_difference.Properties.VariableNames = delta_variables
```

Variance Inflation Factor

```
VIF = array2table(diag(inv(corrcoef(data(:,2:end))))');
VIF.Properties.VariableNames = variables(2:end)
```

Belsley Collinearity diagnotic

```
[sv,conIdx,varDecomp] = collintest(data(:,2:end),'display','off');
Collintest = array2table([sv,conIdx,varDecomp]);
Collintest.Properties.VariableNames = [{'sValue','condIdx'}, variables(2:end)]
```

Unit root test

Augmented Dickey-Fuller test

H0: Non-stationary vs. HA: Stationary

```
models  = 1;      % including a constant in alternative model
maxlag  = 12;     % 12 for monthly data (arbitrary number)
ic      = 'BIC';  % BIC as information criteria
alpha   = [ 0.01, 0.05, 0.10 ]';

stat = nan(1,j); pval = nan(1,j);
cval = nan(6,j); lags = nan(1,j);
```

First Difference

```
for i = 1:j
    [stat(:,i), pval(:,i), cval(:,i), ~, lags(:,i)]...
        = augdfautolag(ldata(:,i), models, maxlag, ic);
end

delta_stationary_variables     = delta_variables(pval < alpha(2));
delta_non_stationary_variables = delta_variables(pval > alpha(2));

ADF_1st_difference = array2table([stat; cval(2,:); pval; lags]);
ADF_1st_difference.Properties.RowNames = {'Test Statistic','Critical Value','p-
value','lags'};
ADF_1st_difference.Properties.VariableNames = delta_variables
```

Price Levels

```
for i = 1:j
[stat(:,i),  pval(:,i),  cval(:,i), ~, lags(:,i)]  = augdfautolag(data(:,i), models,
maxlag, ic);
end

stationary_variables     = variables(pval < alpha(2)); % stationary variable names
non_stationary_variables = variables(pval > alpha(2)); % non stationary variable names
non_stationary_variable  = non_stationary_variables(2:end); % excluding D

ADF_level = array2table([stat; cval(2,:); pval; lags]);
ADF_level.Properties.RowNames = ADF_1st_difference.Properties.RowNames;
ADF_level.Properties.VariableNames = variables
```

Cointegration

Define non-stationary variables

```
non_stationary = pval(2:end) > alpha(2); % columns of non-stationary variables (exl. D)

Y_t = data(:, 1);
X_t = data(:,2:end);
X_t = X_t(:,non_stationary);  % non stationary variables (exl. D)
```

Single Cointegration

Augmented Dickey Fuller test

```
[~, c] = size(X_t);
stat = nan(1, c); pval = nan(1, c);
cval = nan(6, c); lags = nan(1, c);

 for i = 1:c
     reg_t = fitlm(X_t(:,i), Y_t);
     [stat(:,i), pval(:,i),  cval(:,i), ~, lags(:,i)] = ...
                          augdfautolag(reg_t.Residuals.Raw, models, maxlag, ic);
 end

cointegrated_variables = non_stationary_variable(pval < alpha(2));
coint = array2table([stat; cval(2,:); pval; lags]);
coint.Properties.VariableNames = strcat({'D ~ '}, non_stationary_variable);
coint.Properties.RowNames = ADF_1st_difference.Properties.RowNames

cointegrated = pval < alpha(2);
cointegrated_X_t = X_t(:, cointegrated); % Define coitnegrated variables
```

Information criteria

```
InformationCriteria = varorder([Y_t, X_t], 12); % no. of lags in VAR
lag = InformationCriteria.bicor - 1             % lags in VECM (lag in VAR - 1)

for i = 1:sum(pval < alpha(2))
    IC = varorder([Y_t, cointegrated_X_t(:,i)], 12); % no. of lags
    bicor(1,i) = IC.bicor;
end

lags = array2table([bicor - 1]);  % lags in ECM
lags.Properties.RowNames = {'lags of ECM'};
lags.Properties.VariableNames = cointegrated_variables
```

Multiple Cointegration

Johanssen test

Null hypothesis *H*(*r*) of cointegration rank less than or equal to *r*

H1*: Intercepts; Data: no deterministic trends in the levels of the data.

```
[htrace, pValue1, stat1, cValue1, mles1] = jcitest([Y_t, X_t],'test','trace', 'model',
'H1*','lags',lag);

[hmaxeig, pValue2, stat2, cValue2, mles2] = jcitest([Y_t, X_t],'test','maxeig','model',
'H1*','lags',lag);
```

Part II. Models

Define In-Sample variables (August 31, 1998 to December 31, 2018)

```
Y_t = data01(:,1);
X_t = data01(:,2:end);
X_t = X_t(:, non_stationary);

Y_t_min_1 = lagmatrix(Y_t,1); Y_t_min_1(1,:) = [];
X_t_min_1 = lagmatrix(X_t,1); X_t_min_1(1,:) = [];

delta_Y_t = diff(Y_t,1);
delta_X_t = diff(X_t,1,1);
delta_Y_t_min_1 = lagmatrix(Y_t_min_1,1); delta_Y_t_min_1(1,:) = [];
delta_X_t_min_1 = lagmatrix(X_t_min_1,1); delta_X_t_min_1(1,:) = [];
```

Models

NAÏVE

```
RW = arima(0,1,0);
RW.Constant = 0;
RW.Variance = 1;
RW = estimate(RW,Y_t);

RW_model = summarize(RW);

error_naive = infer(RW,    Y_t);
```

ARMA

```
 detrended = detrend(Y_t);
 trend     = Y_t - detrended;

 Detrending_Demolition_Prices = figure;
 plot(time, detrended,time, Y_t, time, trend, ':k', 'LineWidth', 1.5); axis tight
 legend([{'Detrended Demolition price'},{'Demolition price'},{'Time trend'}],
'Location','northwest'); legend('boxoff')
 xlabel('Date'); ylabel('Price'); title('Detrending');

 ARMA       = arima('Constant', NaN, 'ARLags', 1, 'MALags', 1);
 ARMA       = estimate(ARMA,   detrended);
 ARMA_model = summarize(ARMA);

 error_ARMA  = infer(ARMA,  detrended);
```

VECM

```
 rank       = sum(table2array(hmaxeig)); % number of cointegration relationship
 nseries    = sum(non_stationary) + 1;   % number of series

 mdl        = vecm(nseries, rank, lag); mdl.SeriesNames = non_stationary_variables;
 VECM       = estimate(mdl,   [Y_t, X_t],'model','H1*');
 VECM_model = summarize(VECM);

 error_VECM  = infer(VECM, [Y_t, X_t]);
```

ECM

```
 cointegrated_X_t = X_t(:, cointegrated);

 c = size(cointegrated_X_t,2);

 for i = 1:c
     mdl        = vecm(2, 1, 1);
     ECM        = estimate(mdl,   [Y_t, cointegrated_X_t(:,i)],'model','H1*');

     ECM_CointegrationConstant(i,1) = ECM.CointegrationConstant;
     ECM_Cointegration(i,1)         = ECM.Cointegration(1);
     ECM_Constant(i,:)    = ECM.Constant(1);
     ECM_Adjustment(i,:)  = ECM.Adjustment(1);
     ECM_ShortRun(i,:)    = ECM.ShortRun{1}(1,:);
     ECM_Impact(i,:)      = ECM.Impact(1,1);

     ECM_model            = summarize(ECM);
     ECM_AIC(:,i)         = ECM_model.AIC;
```

```
    ECM_BIC(:,i)         = ECM_model.BIC;
    ECM_LLH(:,i)         = ECM_model.LogLikelihood;

    ECM_SE(:,i)          = ECM_model.Table.StandardError;
    ECM_pvalue(:,i)      = ECM_model.Table.PValue;

    errors           = infer(ECM, [Y_t, cointegrated_X_t(:,i)]);
    error_ECM(:,i)   = errors(:,1);
end

ECM_models = array2table([ECM_CointegrationConstant, ECM_Cointegration, ECM_Constant,
ECM_Impact, ECM_ShortRun]);
ECM_models.Properties.VariableNames = {'Cointegration Constant','Cointegration',
'Constant','ECT','D t-1', 't-1'};
ECM_models.Properties.RowNames = cointegrated_variables;
```

Plots

```
name = [{'VECM','ARMA'}, strcat({'ECM '}, cointegrated_variables), {'Näive'}];
Estimated_errors = [ error_VECM(:,1) error_ARMA(3:end) error_ECM error_naive(3:end)];

l = size(Estimated_errors,2);
Fitted_values = Estimated_errors + Y_t(3:end);

figure_InSample_Residuals = figure;

for j = 1:l-1
    subplot(3,2,j);
    h1 = plot(time(3:end), Estimated_errors(:,l),'LineWidth',1.0); hold on;
    h2 = plot(time(3:end), Estimated_errors(:,j),'LineWidth',1.0); axis tight
    legend([h1 h2],[{'Näive'}, name(j)],'Location','southwest'); legend('boxoff')
    xlabel('Date'); ylabel('Residuals'); title(name(j));
end

figure_InSample_Fitted_values = figure;

for j = 1:l-1
    subplot(3,2,j);
    h1 = plot(time(3:end), Y_t(3:end),'LineWidth',1.0); hold on;
    h2 = plot(time(3:end), Fitted_values(:,j),'LineWidth',1.0); axis tight
    legend([h1 h2],[{'D'}, name(j)],'Location','northwest'); legend('boxoff')
    xlabel('Date'); ylabel('Price level'); title(name(j));
end
```

Forecasting

Define out-of-sample variables (January 31, 2019 to December 31, 2020)

```
Y_t_19 = data19(:,1);
X_t_19 = data19(:,2:end);
X_t_19 = X_t_19(:, non_stationary);

Y_t_min_1_19 = lagmatrix(Y_t_19,1); Y_t_min_1_19(1,:) = [];
X_t_min_1_19 = lagmatrix(X_t_19,1); X_t_min_1_19(1,:) = [];

delta_Y_t_19 = diff(Y_t_19,1);
delta_X_t_19 = diff(X_t_19,1,1);

delta_Y_t_min_1_19 = lagmatrix(Y_t_min_1_19,1); delta_Y_t_min_1_19(1,:) = [];
delta_X_t_min_1_19 = lagmatrix(X_t_min_1_19,1); delta_X_t_min_1_19(1,:) = [];
```

Forecasting Models

```
cointegrated_X_t_19 = X_t(:, cointegrated);

N19             = size(Y_t_19,1);
Forecast_ARMA   = nan(N19,1);
Forecast_VECM   = nan(N19,7);
Forecasting_ECM = nan(N19,2);

nperiods = 1;

for t = 2:N19
    Forecast_ARMA(t,:)  = forecast(ARMA, nperiods,  Y_t_19(1:t));
    Forecast_VECM(t,:)  = forecast(VECM, nperiods, [Y_t_19(1:t), X_t_19(1:t,:)]);

    for i = 1:c
        ECM  = estimate(mdl,  [Y_t, cointegrated_X_t(:,i)]);
Forecasting_ECM(t,:) = forecast(ECM,nperiods,[Y_t_19(1:t),cointegrated_X_t_19(1:t,i)]);

        Forecast_ECM(:,i)    = Forecasting_ECM(:,1);
    end
end
```

Forecasted Residuals

```
Forecasted_values = [Forecast_VECM(2:end,1),Forecast_ARMA(2:end),Forecast_ECM(2:end,:),
Y_t_min_1_19];

 Forecasted_errors =  Y_t_19(2:end) - Forecasted_values;
```

Figures

```
figure_OutOfSample_forecasted_errors = figure;

for j = 1:l-1
    subplot(3,2,j);
    h1 = plot(time19(2:end), Forecasted_errors(:,l),'LineWidth',1.5); hold on;
    h2 = plot(time19(2:end), Forecasted_errors(:,j),'LineWidth',1.5); axis tight
    legend([h1 h2],[{'Näive'}, name(j)],'Location','southwest'); legend('boxoff')
    xlabel('Date'); ylabel('Residuals'); title(name(j));
end

figure_OutOfSample_forecasted_values = figure;

for j = 1:l-1
    subplot(3,2,j);
    h1 = plot(time19(2:end), Y_t_19(2:end),'LineWidth',1.5); hold on;
    h2 = plot(time19(2:end), Forecasted_values(:,j),'LineWidth',1.5); axis tight
    legend([h1 h2],[{'D'}, name(j)],'Location','southwest'); legend('boxoff')
    xlabel('Date'); ylabel('Price'); title(name(j));
end
```

Model accuracy

In-Sample Results

```
Model_measure = array2table([VECM_model.LogLikelihood, ARMA_model.LogLikelihood, ...
ECM_LLH; VECM_model.AIC, ARMA_model.AIC, ECM_AIC; VECM_model.BIC, ARMA_model.BIC, ...
ECM_BIC]);

 Model_measure.Properties.RowNames      = {'LogLikelihood','AIC','BIC'};
 Model_measure.Properties.VariableNames = name(1:end-1)

 In_Sample_results = table();
 In_Sample_results.RMSE   = sqrt(mean(Estimated_errors.^2))';
 In_Sample_results.MAE    = mean(abs(Estimated_errors))';
 In_Sample_results.ME     = mean(Estimated_errors)';
 In_Sample_results.TS     = In_Sample_results.ME./In_Sample_results.MAE;
 In_Sample_results.mean   = mean(Fitted_values)';
 In_Sample_results.RMSE_norm = In_Sample_results.RMSE./mean(Y_t);
 In_Sample_results.Properties.VariableNames = {'RMSE','MAE','ME','TS','Mean','RMSE
normalized'};
 In_Sample_results.Properties.RowNames          = name
```

Out-of-Sample Results

```
 Out_of_Sample_results = table();
 Out_of_Sample_results.RMSE = sqrt(mean(Forecasted_errors.^2))';
 Out_of_Sample_results.MAE  = mean(abs(Forecasted_errors))';
 Out_of_Sample_results.ME   = mean(Forecasted_errors)';
 Out_of_Sample_results.TS   = Out_of_Sample_results.ME./Out_of_Sample_results.MAE;
 Out_of_Sample_results.mean = mean(Forecasted_values)';
 Out_of_Sample_results.RMSE_norm = Out_of_Sample_results.RMSE./mean(Y_t_19);
 Out_of_Sample_results.Properties.VariableNames = {'RMSE','MAE','ME','TS','Mean','RMSE
normalized'};
 Out_of_Sample_results.Properties.RowNames          = name
```

Part III. Stability test

```
 afc = (time > '2008-12-31');   % after the financial crisis

 Y_t = data01(:,1);
 X_t = data01(:,2:end);
 X_t = X_t(:,non_stationary);

 cointegrated_X_t = X_t(:, cointegrated);

 Y_t_min_1 = lagmatrix(Y_t,1); Y_t_min_1(1,:) = [];
 X_t_min_1 = lagmatrix(X_t,1); X_t_min_1(1,:) = [];

 cointegrated_X_t_min_1 = lagmatrix(cointegrated_X_t,1);
 cointegrated_X_t_min_1(1,:)=[];

 delta_Y_t = diff(Y_t,1);
 delta_X_t = diff(X_t,1,1);
 delta_cointegrated_X_t_min_1 = diff(cointegrated_X_t_min_1);

 delta_Y_t_min_1 = lagmatrix(Y_t_min_1,1); delta_Y_t_min_1(1,:) = [];
 delta_X_t_min_1 = lagmatrix(X_t_min_1,1); delta_X_t_min_1(1,:) = [];
```

Parameter Estimation

VECM

```
XX = [VECM.Cointegration.*[Y_t_min_1 X_t_min_1]' + VECM.CointegrationConstant]';
XY = Y_t_min_1 - sum(XX,2);
```

ECM

```
c = size(cointegrated_X_t,2);

for i = 1:c
    XXX(:,i) = [ECM_Cointegration(i).*cointegrated_X_t_min_1(:,i)...
                + ECM_CointegrationConstant(i)]';

    XXY(:,i) = Y_t_min_1 - XXX(:,i);
end
```

ARMA

```
delta_detrended_t_min_1      = lagmatrix(diff(detrended),1);
delta_detrended_t_min_1(1,:) = [];
```

Chow Test

```
bp   = find(time == '2007-12-31'); % breaking point
```

VECM

```
 [h1 p1 stat1 cv1] = chowtest([XY(2:end), delta_Y_t_min_1, delta_X_t_min_1], ...
delta_Y_t(2:end), bp-3);
```

ARMA

```
 [h2 p2 stat2 cv2] = chowtest([Y_t_min_1(2:end), delta_detrended_t_min_1  ], ...
Y_t(3:end),  bp-3);
```

ECM

```
for i = 1:c
   [h3(i) p3(:,i) stat3(:,i) cv3(:,i)] = chowtest(...
       [delta_cointegrated_X_t_min_1(:,i), XXY(2:end,i), delta_Y_t_min_1],...
         delta_Y_t(2:end), bp-3);
end
```

Summary

```
Stability = array2table([h1 h2 h3; p1 p2 p3; stat1 stat2 stat3; cv1 cv2 cv3]);
Stability.Properties.VariableNames = name(1:end-1);
Stability.Properties.RowNames = {'ChowTest','p value','stat','cval'}
```

CUSUM Test

VECM

```
cusumtest([XY(2:end), delta_Y_t_min_1, delta_X_t_min_1],...
           delta_Y_t(2:end), 'Intercept', true,'Alpha',0.1);
```

ARMA

```
cusumtest([Y_t_min_1(2:end), delta_detrended_t_min_1 ],...
           Y_t(3:end),'Intercept',true);
```

ECM

```
for i = 1:c
    cusumtest([delta_cointegrated_X_t_min_1(:,i), XXY(2:end,i), delta_Y_t_min_1],...
               delta_Y_t(2:end),'Intercept',true);
end
```