Time Series Outliers Detection

Johnew Zhang

April 14, 2015

Contents

1 Software/Hardware Requirements

Multiple time series outliers detection algorithm is implemented in R and can be ran in Windows and Mac OS X. It is worth to mention that R is very similar to matlab in terms of general looping, defining variables and using functions. The followings are some basic utility for programming in R

- R: Native language interpreter

  Windows http://cran.r-project.org/bin/windows/base/

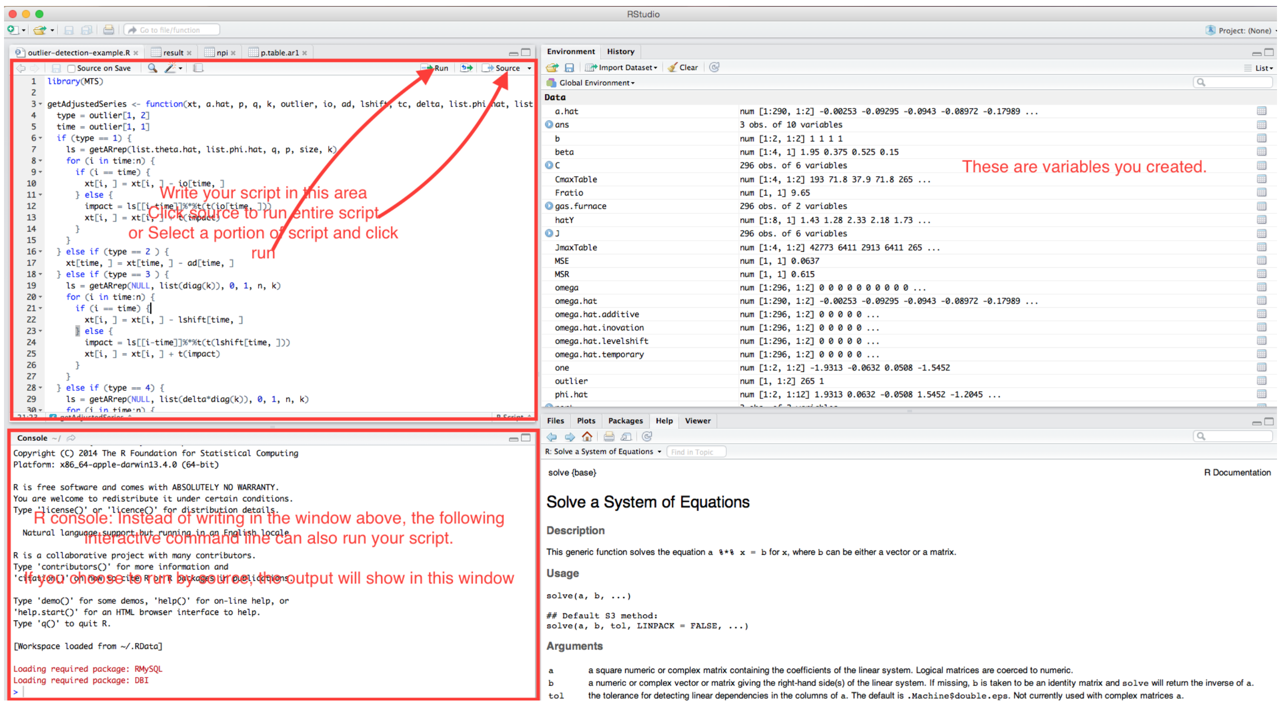
  Mac OS http://cran.r-project.org/bin/macosx/

- R Studio: Recommended R programming editor

  Windows http://download1.rstudio.org/RStudio-0.98.1102.exe

  Mac OS http://download1.rstudio.org/RStudio-0.98.1102.dmg

  The following is a screen shot of R in Mac OS to provide a general idea of the editing software.


RStudio
Go to file/function
Project: (None)
outlier-detection-example.R
result
npi
p.table.ar1
Source on Save
Run
Source
Write your script in this area
Click source to run entire script
or Select a portion of script and click run
Environment
History
Import Dataset
Clear
List
Global Environment
These are variables you created.
Console ~/
R console: Instead of writing in the window above, the following interactive command line can also run your script.
If you choose to run by source, the output will show in this window
Files
Plots
Packages
Help
Viewer
R: Solve a System of Equations
Find in Topic
getAdjustedSeries
R Script

```
library(MTS)

getAdjustedSeries <- function(xt, a.hat, p, q, k, outlier, io, ad, lshift, tc, delta, list.phi.hat, list
  type = outlier[1, 2]
  time = outlier[1, 1]
  if (type == 1) {
    ls = getARrep(list.theta.hat, list.phi.hat, q, p, size, k)
    for (i in time:n) {
      if (i == time) {
        xt[i, ] = xt[i, ] - io[time, ]
      } else {
        impact = ls[[i-time]]%*%t(t(io[time, ]))
        xt[i, ] = xt[i, ] + t(impact)
      }
    }
  } else if (type == 2 ) {
    xt[time, ] = xt[time, ] - ad[time, ]
  } else if (type == 3 ) {
    ls = getARrep(NULL, list(diag(k)), 0, 1, n, k)
    for (i in time:n) {
      if (i == time) {
        xt[i, ] = xt[i, ] - lshift[time, ]
      } else {
        impact = ls[[i-time]]%*%t(t(lshift[time, ]))
        xt[i, ] = xt[i, ] + t(impact)
      }
    }
  } else if (type == 4) {
    ls = getARrep(NULL, list(delta*diag(k)), 0, 1, n, k)
    for (i in time:n) {
```

Data

a.hat	num [1:290, 1:2] -0.00253 -0.09295 -0.0943 -0.08972 -0.17989 ...
ans	3 obs. of 10 variables
b	num [1:2, 1:2] 1 1 1 1
beta	num [1:4, 1] 1.95 0.375 0.525 0.15
C	296 obs. of 6 variables
CmaxTable	num [1:4, 1:2] 193 71.8 37.9 71.8 265 ...
Fratio	num [1, 1] 9.65
gas.furnace	296 obs. of 2 variables
hatY	num [1:8, 1] 1.43 1.28 2.33 2.18 1.73 ...
J	296 obs. of 6 variables
JmaxTable	num [1:4, 1:2] 42773 6411 2913 6411 265 ...
MSE	num [1, 1] 0.0637
MSR	num [1, 1] 0.615
omega	num [1:296, 1:2] 0 0 0 0 0 0 0 0 0 0 ...
omega.hat	num [1:290, 1:2] -0.00253 -0.09295 -0.0943 -0.08972 -0.17989 ...
omega.hat.additive	num [1:296, 1:2] 0 0 0 0 0 ...
omega.hat.inovation	num [1:296, 1:2] 0 0 0 0 0 ...
omega.hat.levelshift	num [1:296, 1:2] 0 0 0 0 0 ...
omega.hat.temporary	num [1:296, 1:2] 0 0 0 0 0 ...
one	num [1:2, 1:2] -1.9313 -0.0632 0.0508 -1.5452
outlier	num [1, 1:2] 265 1
phi.hat	num [1:2, 1:12] 1.9313 0.0632 -0.0508 1.5452 -1.2045 ...

```
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

Loading required package: RMySQL
Loading required package: DBI
> 
```

solve {base} R Documentation

Solve a System of Equations

Description

This generic function solves the equation `a %*% x = b` for `x`, where `b` can be either a vector or a matrix.

Usage

```
solve(a, b, ...)

## Default S3 method:
solve(a, b, tol, LINPACK = FALSE, ...)
```

Arguments

a	a square numeric or complex matrix containing the coefficients of the linear system. Logical matrices are coerced to numeric.
b	a numeric or complex vector or matrix giving the right-hand side(s) of the linear system. If missing, `b` is taken to be an identity matrix and `solve` will return the inverse of `a`.
tol	the tolerance for detecting linear dependencies in the columns of `a`. The default is `.Machine$double.eps`. Not currently used with complex matrices `a`.

- Required Libraries: If additional libraries are required for MTS, we can type

$$install.packages(<\text{ package name }>)$$

  in the R console to install the missing packages.

MTS Multiple Time Series Package

1

2 Time Series Analysis in R

2.1 Load Data

The very first step to perform our analysis is to load the data into the program. The most common data file format used in general is CSV. Of course, there are other means of importing data such as MySQL, flat text file and etc.. Here, CSV file is used. In our gas furnace example, the data file is called "gas-furnace.csv" and stored in Downloads folder. Hence the following command is used to load the data into R.

```
>gas.furnace <- read.csv("~/Downloads/gas-furnace.csv")
```

The read.csv function takes the absolute path of the data file.

2.2 Load Library

Once this data is in place, a time series library need to be called to perform the maximum likelihood estimation. To import a library, we can use the following

```
>library(MTS)
```

2.3 VARMA using R

Now let's have a brief introduction of how to use the MTS library. VARMA function in MTS perform a conditional ML estimation for the parameters Θ, Φ,and Σ. Multivariate Gaussian likelihood function is used.

Usage VARMA(da, p = 0, q = 0, include.mean = T, fixed = NULL, beta = NULL, sebeta = NULL, prelim = F, details, thres = 2)

Arguments **da** Data matrix (T-by-k) of a k-dimensional time series with sample size T.

p AR order

q MA order

include.mean A logical switch to control estimation of mean vector. Default is to include the mean in estimation

fixed A logical matrix to control zero coefficients in estimation. It is mainly use by the command refVARMA

[1]R should be installed first in order to use RStudio.

beta Parameter estimates to be used in model simplification, if needed

sebeta Standard errors of parameter estimates for use in model simplification

prelim A logical switch to control preliminary estimation. Default is none.

details A logical with to control the amount of output

thres A threshold used to set zero parameter constraints based on individual t-ratio. Default is 2.

In our VAR(6) model, we can apply the following

```
> est = VARMA(gas.furnace, p = 6, q = 0, include.mean = TRUE)
Number of parameters:  26
initial estimates:  0.8167 3.8028 1.9245 -0.0512 -1.1946 0.0984 0.1654 -0.0778 -0.1598
 0.0258 0.3818 -0.042 -0.2171 0.0314 0.0686 1.5447 -0.1422 -0.5902 -0.4357 -0.1744
 0.1522 0.1335 -0.1247 0.0586 0.2528 -0.0434
Par. lower-bounds:  -0.5009 2.1149 1.8072 -0.1432 -1.4487 -0.0713 -0.1252 -0.255 -0.4526
-0.1506 0.1056 -0.1972 -0.386 -0.0346 -0.0816 1.4268 -0.4677 -0.8076 -0.8079 -0.4014
-0.2229 -0.0925 -0.4786 -0.1402 0.0364 -0.128
Par. upper-bounds:  2.1343 5.4906 2.0417 0.0409 -0.9405 0.2681 0.4559 0.0994 0.1331
0.2022 0.6581 0.1131 -0.0482 0.0975 0.2188 1.6626 0.1832 -0.3729 -0.0636 0.0527 0.5274
0.3594 0.2291 0.2574 0.4692 0.0412
Final   Estimates:  0.7699884 3.824113 1.931322 -0.05076185 -1.204466 0.09990707
0.1696947 -0.0795978 -0.1601987 0.02685074 0.3801984 -0.04144202 -0.2136601
0.03052928 0.06315954 1.545225 -0.1334534 -0.592931 -0.4412377 -0.1710573
0.1520018 0.1323837 -0.1203567 0.05687094 0.249303 -0.04208589

Coefficient(s):
              Estimate  Std. Error  t value Pr(>|t|)
InputGasRate   0.76999     0.65431    1.177   0.2393
CO2            3.82411     0.82880    4.614 3.95e-06 ***
InputGasRate   1.93132     0.05812   33.228  < 2e-16 ***
CO2           -0.05076     0.04574   -1.110   0.2671
InputGasRate  -1.20447     0.12613   -9.549  < 2e-16 ***
CO2            0.09991     0.08432    1.185   0.2361
InputGasRate   0.16969     0.14433    1.176   0.2397
CO2           -0.07960     0.08810   -0.903   0.3663
InputGasRate  -0.16020     0.14549   -1.101   0.2709
CO2            0.02685     0.08770    0.306   0.7595
InputGasRate   0.38020     0.13723    2.770   0.0056 **
CO2           -0.04144     0.07712   -0.537   0.5910
InputGasRate  -0.21366     0.08394   -2.545   0.0109 *
```

```
CO2           0.03053     0.03281    0.931   0.3521
InputGasRate  0.06316     0.07424    0.851   0.3949
CO2           1.54523     0.05554   27.824  < 2e-16 ***
InputGasRate -0.13345     0.16108   -0.828   0.4074
CO2          -0.59293     0.10473   -5.661 1.50e-08 ***
InputGasRate -0.44124     0.18436   -2.393   0.0167 *
CO2          -0.17106     0.11240   -1.522   0.1280
InputGasRate  0.15200     0.18572    0.818   0.4131
CO2           0.13238     0.11190    1.183   0.2368
InputGasRate -0.12036     0.17534   -0.686   0.4925
CO2           0.05687     0.09849    0.577   0.5637
InputGasRate  0.24930     0.10717    2.326   0.0200 *
CO2          -0.04209     0.04192   -1.004   0.3154
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
---
Estimates in matrix form:
Constant term:
Estimates:  0.7699884 3.824113
AR coefficient matrix
AR( 1 )-matrix
       [,1]    [,2]
[1,] 1.9313 -0.0508
[2,] 0.0632  1.5452
AR( 2 )-matrix
       [,1]    [,2]
[1,] -1.204  0.0999
[2,] -0.133 -0.5929
AR( 3 )-matrix
       [,1]    [,2]
[1,]  0.170 -0.0796
[2,] -0.441 -0.1711
AR( 4 )-matrix
       [,1]   [,2]
[1,] -0.160 0.0269
[2,]  0.152 0.1324
AR( 5 )-matrix
      [,1]    [,2]
[1,]  0.38 -0.0414
[2,] -0.12  0.0569
AR( 6 )-matrix
```

```
       [,1]    [,2]
[1,] -0.214  0.0305
[2,]  0.249 -0.0421

Residuals cov-matrix:
             [,1]         [,2]
[1,]  0.034085314 -0.002294762
[2,] -0.002294762  0.055650286
----
aic=  -6.094661
bic=  -5.770508
```

The est variable is a list that contains all the estimates for AR(6) models. To access the ML estimation results, one can use the following command

```
# residuals
> est$residuals
# sigma
> est$Sigma
# phi estimates
> est$Phi
# theta estimates
> est$Theta
```

Note, all of the above commands return matrixes object. Hence, for example, in a 2-dimensional AR(6) model, Σ is a 2×2 matrix and Φ is a 12×2 matrix. To convert a multi-dimensional to a list of black entry is preferable. In other word, in our example, Φ is split into a list of 6 2×2 matrixes. In addition, since a conditional estimate is used, the residuals are p short compared to the total amount of observations, that is, the first p observations have zero residuals.

3 Multiple Time Series Outlier Detection

Let $x_t = (x_{1t}, \cdots, x_{kt})'$ be a k-dimensional time series that follows a vector autoregressive integrated moving-average, ARIMA, model

$$\Phi(B)x_t = c + \Theta(B)\epsilon_t,$$

where

$$\Phi(B) = I - \sum_{i=1}^{p} \Phi_i B^p, \Theta(B) = I - \sum_{i=1}^{p} \Theta_i B^p$$

are $k \times k$ matrix polynomials of finite degrees p and q, B is the backshift operator such that $Bx_t = x_{t-1}$,c is a k-dimensional constant vector, and $\{\epsilon = (\epsilon_{1t}, \cdots, \epsilon_{kt})'\}$ is a sequence of independent and identically distributed Gaussian random vector with zero mean and positive-definite covariance matrix Σ. We assume that $\Phi(B)$ and $\Theta(B)$ are left caprice and that all of the zeros of the determinants $|\Phi(B)|$ and $|\Theta(B)|$ are on or outside the unit circle.

Define the autoregressive representation as

$$\Pi(B)x_t = c_0 + \epsilon_t$$

where

$$\Pi(B) = I - \sum_{i=1}^{\infty} \Pi_i B^i = \{\Theta(B)\}^{-1}\Phi(B)$$

Define the moving-average representation as

$$x_t = c_* + \Psi(B)\epsilon_t$$

where

$$\Psi(B) = I - \sum_{i=1}^{\infty} \Psi_i B^i = \{\Phi(B)\}^{-1}\Theta(B)$$

3.1 The Procedure

Suppose observed time series has no outlier and provide model $ARMA(p, q)$

Step 1: Use ML estimation to estimate Φ, Θ and Σ. (MTS provides VARMA function to perform conditional ML estimation. As a result the first p residuals are assumed to be zero)

Step 2: Use resulted estimation to calculate estimated residuals, $\hat{a}_t$ and the estimated AR representation, $\hat{\Pi}_i$[2].

Step 3: Obtain $\omega_{i,h}$ for all h where the subscript i indicates type of outliers ($i = I, A, L, S$) [3] and the covariance estimator.

Step 4: To test the significance of a multivariate outlier at time index h, we consider the null hypothesis $H_0 : \omega = 0$ versus the alternative hypothesis $H_a : \omega \neq 0$. Two test statistics are used.

- $J_{i,h} = \hat{\omega}'_{i,h}\Sigma_{i,h}^{-1}\hat{\omega}_{i,h}$ where $i = I, A, L$ or T. (chi-square random variable with k degrees of freedom)

[2]A recursive algorithm is used to calculate the polynomials for AR representation. Detailed explanation is in the Appendix

[3]ω derivation is in the Appendix.

- The maximum z-statistics

$$C_{i,h} = \max_{1\le j\le k} |\hat{\omega}_{j,i,h}|/\sqrt{\sigma_{j,i,h}}$$

where $\hat{\omega}_{j,i,h}$ and $\sigma_{j,i,h}$ are the nth element of $\hat{\omega}_{i,h}$ and the (j,j)th element of $\Sigma_{i,h}$ respectively.

We define the overall test statistics as

$$J_{\max}(i,h_i) = \max_h J_{i,h}, C_{\max}(i,h)i^*) = \max_h C_{i,h}(i = I,A,L,T)$$

^{4} In case of multiple significant joint test statistics, we identify the outlier type based on the test that has the smallest empirical p-value^5. If none of $J_{\max}$ are significant, $C_{\max}$ will be tested. Otherwise, our procedure end.

Step 5: Once an outlier is detected, we will adjusted the original time series by subtracting $\alpha(B)\omega\epsilon_t^{(h)}$ as indicated in the Appendix. Then the process jumps back to Step 1.

4 Similarities and Differences Between R Program Results and Paper Results

From Proposed Paper	
43	MTC
55	MTC
265	MIO
199	MLS
113	MTC
288	MLS
287	MLS
236	MLS
82	MLS
262	MIO
91	MTC
197	MTC

From R program	
43	MTC
55	MTC
265	MIO
113	MTC
199	MTC
235	MLS
91	MTC
262	MIO
288	MLS
287	MLS
82	MLS

Above two tables show the overall outliers picked up by the paper and our R program. The R program only missed one outlier (197, MTC) and has one outlier a little bit off at (235, MLS) rather than (236, MLS). In addition, since we did not use the percentile to pick up the most critical outlier each time, it would not be reasonable to compare the iteration output since the first iteration step. However, it is still valuable to look at the difference

^{4}I: innovative outlier, A: additive outlier, L: level shift and T: temporary change

^{5}Via a simulation process, we can generate the empirical distribution table for both statistics.

from the R program and proposed paper in terms of the first iteration step. The following table illustrates such:

Jmax IO	Jmax AO	Jmax LS	Jmax TC [6]
-0.04%	0.00%	-0.02%	6.48%

As you notice, the Jmax TC is a little bit too high. It could be associated with the δ value (currently, we picked 0.7 as the default value) or our calculation algorithm. In terms of the empirical distribution of Jmax, I did a little program in the Appendix. However, the Jmax for TC issue is not resolved so the empirical distribution is not quite comparable to the proposed paper. Hence, here I will not list the distribution. Please feel free to provide advice on this portion.

5 Limitations and Comments

Here are a list of limitations:

- A conditional estimation is applied for VARMA estimation. However, we are still lack of estimation algorithm for the exact ML estimation. The conditional estimation program is implemented by the author of the paper, Ruey S. Tsay. However, here, we are not sure if he used the same program to generate the result for the paper. Therefore, the computed result from my program can be off.
- Currently, my outlier filter procedure is only filtered by the critical value provided for AR(6) model. In addition, since I don't have the entire empirical distribution for AR(6) model, a simple ranking is applied. Simply, I just filtered all J-statistic and C-statistic by the the critical value and then picked the maximum value as the proposed outlier. However, a implemented procedure to produce the distribution is included in the Appendix of this report. The running time for this procedure is very expensive so I decided not to use it until I received some help.

Here are a list of concerns or comments:

- In calculating the ω value for level shift, is inverting polynomial $(I - B)$ valid? So in this case, $(I - B)^{-1}$ would be a diverging alternating series. I raise this question because in my calculation of $J_{max,LS}$ is always higher than paper results. In similar case, $J_{max,TC}$ is even worse but inverting $(I - \delta B)$ results a diverging series. It could be my code? (I've verified few times and don't think there is any issue in my code or maybe I miss it.)
- Just by looking into the gas furnace example, the proposed outliers from my program are quite consistent from the ones from the paper. Maybe the proposed detection algorithm is robust.

[6] All potential outliers in the first iteration are the same between the proposed paper and the R program.

Appendix I

As according to the Outliers in multivariate time series paper, denote the observed time series by $y_t = (y_{1t}, \cdots, y_{kt})'$ and let $\omega = (\omega_1, \cdots, \omega_k)'$ be the size of the intial impact of an outlier on the series x_t. The fourt types of univariate outlier can be generalized to the multivariate case:

$$y_t = x_t + \alpha(B)\omega\xi_t^{(h)},$$

where

$$\alpha(B) = \begin{cases} \Psi(B) & \text{multivariate innovational outlier} \\ I & \text{multivariate additive outlier} \\ (1-B)^{-1} & \text{multivariate level shift} \\ \{D(\delta)\}^{-1} & \text{multivariate temporary change} \end{cases}$$

Here, Ψ is the MA representation of VARMA model. Next we can multipy above equation by $\Pi(B)$ and subtract a constant term from both sides, we have

$$a_t = \epsilon_t + \Pi(B)\alpha(B)\omega\xi_t^{(h)}.$$

Here let's write $\Pi(B)\alpha(B)$ as $\Pi^*(B)$. Therefore, if we suppose $\hat{a}_t$ is the estimated residuals and $\hat{\Pi}_i$ is the estimated coefficients of the autoregressive representation, we have the following

$$\hat{a}_t = (I - \sum_{i=1}^{\infty} \hat{\Pi^*}_i B^i)\xi_{(h)t}\omega + \epsilon_t = (\xi t^{(h)} - \sum_{i=1}^{\infty} \hat{\Pi^*}_i \xi_{t-i}^{(h)})\omega + \epsilon_t,$$

where $\epsilon_t \sim N(0, \Sigma)$, and the estimator of ω is

$$\hat{\omega}_{i,h} = -(\sum_{i=0}^{n-h} \hat{\Pi^*}_i' \Sigma^{-1} \hat{\Pi^*})^{-1} \sum_{i=0}^{n-h} \hat{\Pi^*}_i' \Sigma^{-1} \hat{a}_{h+i}$$

where $\Pi_0 = -I$ and $i = I, A, L, T$.

For the Π^*, we can obtain by calculating the coefficient of $\Pi(B)\alpha(B)$. Here, additive outlier and level shift are just special case of temporary change so let's only consider temporary change at the moment. That is the following

$$I - \sum_{i=1}^{\infty} \Pi_i^* B^i = (I - \sum_{i=1}^{\infty} \Pi_i B^i)(I - \delta B)^{-1}$$

Hence, above is just a simple polynomial devision. In general, suppose three polynomials

$$\Pi(B) = I - \sum_{i=1}^{\infty} \Pi_i B^i,$$

$$\Phi(B) = I - \sum_{i=1}^{p} \Phi_i B^i$$

and

$$\Theta(B) = I - \sum_{i=1}^{q} \Theta_i B^i$$

where

$$\Pi(B) = \Theta(B)^{-1}\Phi(B).$$

Then

$$\Pi_i = \Phi_i + \sum_{j=1}^{i} \Theta_j \Pi_{i-j}$$

where $i = j$, $\Pi_{i-j} = -I$, if $i > p$, $\Phi_i = 0$ and if $j > q$, $\Theta_j = 0$. Therefore, use the above reclusive relation, we can obtain the polynomial representation for temporary change.

In addition, the covariance matrix of the estimator is $\Sigma_{i,h} = (\sum_{i=0}^{n-h} \hat{\Pi}_i^{*\prime} \Sigma^{-1} \hat{\Pi}_i^{*})^{-1}$. Lastly, to reduce the effect of one particular outlier, we can just proceed to the following equation

$$x_t = y_t - \alpha(B)\omega\xi_t^{(h)}$$

where x_t is the adjusted series and y_t is the original series.

Appendix II

The following code segment can be attached to the multivariate time series outlier detection procedure. It presents a procedure to obtain the empirical distribution for J and C statistics. MTS, VARMAsim simulates 10,000 realization and then we rank the $J_{\max}$ and $C_{\max}$ from all the realization to form the empirical distribution.

```
p <- 6
q <- 0
k <- 2

 xt = gas.furnace
num_sim<- 10000

# this is the initial step-up for the p-value table
est = VARMA(xt, p = p, q = q, include.mean = TRUE)
p.table = getPTable(n ,p, q, est$Phi, est$Theta, est$Sigma, k, delta, num_sim)
```

```
 getPTable <- function(sample_size, p, q, phi, theta, sigma, k, delta, num_sim) {
  table = list()
  Jmax = matrix(0, num_sim, 4)
  Cmax = matrix(0, num_sim, 4)
  for (i in 1:num_sim) {
    print(i)
    res = getEmpiricalDist(sample_size, p, q, phi, theta, sigma, k, delta)
    Jmax[i, ] = res$Jmax
    Cmax[i, ] = res$Cmax
  }
  for (i in 1:4) {
    Jmax[, i] = sort(Jmax[, i])
    Cmax[, i] = sort(Cmax[, i])
  }
  table$Jmax = Jmax
  table$Cmax = Cmax
  return(table)
}

getEmpiricalDist<-function(sample_size, p, q, phi, theta, sigma, k, delta) {
  if (p== 0) {
    sim_xt = VARMAsim(sample_size, malags=c(q), theta = theta, sigma=sigma)
  } else if (q == 0) {
    sim_xt = VARMAsim(sample_size, arlags=c(p), phi = phi, sigma=sigma)
  } else {
    sim_xt = VARMAsim(sample_size, arlags=c(p), malags=c(q), phi = phi,
    theta = theta, sigma=sigma)
  }

  #est = VARMA(sim_xt$series, p = p, q = q, include.mean = TRUE)
  n = sample_size
  #phi = est$Phi
  #theta = est$theta

  list.phi.hat = mat2list(phi, k, p)
  list.theta.hat = mat2list(theta, k, q)
  list.ar.rep = getARrep(list.phi.hat, list.theta.hat, p, q, n, k)
  a.hat = sim_xt$noises
  sigma = var(sim_xt$noises)
  #a.hat = est$residuals
  omega.hat.innovation = getOutlierResidual(a.hat, list.ar.rep, sigma, n, k, 1, 0, p,
  TRUE)
  omega.hat.additive = getOutlierResidual(a.hat, list.ar.rep, sigma, n, k, 2, 0, p,
  TRUE)
  omega.hat.levelshift = getOutlierResidual(a.hat, list.ar.rep, sigma, n, k, 3, 0, p,
  TRUE)
  omega.hat.temporary = getOutlierResidual(a.hat, list.ar.rep, sigma, n, k, 4, delta, p,
  TRUE)

  stats = getStatistics(omega.hat.innovation, omega.hat.additive, omega.hat.levelshift,
   omega.hat.temporary, n, k)
  result = list()
  Jmax = matrix(0, 1, 4)
  Cmax = matrix(0, 1, 4)
  for (i in 1:4) {
    Jmax[1, i] = max(stats[, 2+i])
    Cmax[1, i] = max(stats[, 8+i])
  }
  result$Jmax = Jmax
  result$Cmax = Cmax
  return(result)
}
```