# STAT7017 Big Data Final Project

*Rui Qiu*

*2018-10-29*

## Contents

## 1 Setup

First, we load some required libraries for this project. Basically, they are libraries of `tidyverse` which are responsible for data frame pipelines and the rest are mainly focusing on some particular calculations. We will mention those functions later.

```r
library(mvtnorm)
library(ggplot2)
library(ggfortify)
library(forecast)
library(readr)
library(dplyr)
library(reshape2)
library(heplots)
library(ks)
library(vars)
library(portes)

set.seed(7017)
```

## 2 Testing Covariance Matrices

### 2.1 Q1 Reproduce Box's Test for Equality of Covariance Matrices

The followings are some customized helper functions in order to apply Box's M test and print out some intermediate values. We have also configured some arguments as flags so that the function can be reused in the next few questions, but the output details might vary accordingly.

```r
box.u <- function(n.vec,p,g) {
    u <- (sum(1/(n.vec-1))-1/sum(n.vec-1))*((2*p**2+3*p-1)/(6*(p+1)*(g-1)))
    return(u)
}

log.det <- function(S) {
    return(unlist(determinant(S,logarithm=T))[[1]])
}

box.M <- function(p,g,n.vec,S.list,print.log=F) {
    S.pooled <- matrix(rep(0,p**2),ncol=p,nrow=p)

    for (i in 1:g) {
        S.pooled <- S.pooled + (n.vec[i]-1)*S.list[[i]]/sum(n.vec-1)
    }

    # for this Q1 only
    if (print.log) {
        cat("log determinant of S.pooled:",log.det(S.pooled),"\n")
    }

    M <- sum(n.vec-1)*log.det(S.pooled)
    for (i in 1:g) {
        M <-  M - (n.vec[i]-1)*log.det(S.list[[i]])
    }
    return(M)
}

box.C <- function(u, M) {
    return((1-u)*M)
}

M.test <- function(p,g,n.vec,S.list,test=F,summary=F) {
    u <- box.u(n.vec,p,g)
    if (test) {
        M <- box.M(p,g,n.vec,S.list,print.log=T)
    } else {
        M <- box.M(p,g,n.vec,S.list)
    }
    C <- box.C(u,M)
    nu <- 0.5*p*(p+1)*(g-1)
    if (summary) {
        cat("u =",u,"\n",
        "M =",M,"\n",
        "C =",C,"\n",
        "chi-sq with", nu, "degrees of freedom is", qchisq(.05,nu), "\n",
        "reject null hypothesis:", C > qchisq(.05,nu), "\n")
    }
    return(c(C,qchisq(.05,nu)))
}
```

When all the helper functions are settled, we are safe to reproduce example 6.12 with the following Wisconsin Nursing Homes data. Here, as expected, we printed out an intermediate value in the middle of the calculation, which should be the cause of differences in final values. Further explanation can be found in the following paragraph.

```
M.test(4,3,c(271,138,107),
       list(matrix(c(.291,0,0,0,-.001,.011,0,0,.002,0,.001,0,.010,.003,0,.01),
                   nrow=4,byrow=T),
            matrix(c(.561,0,0,0,.011,.025,0,0,.001,.004,.005,0,.037,.007,.002,.019),
                   nrow=4,byrow=T),
            matrix(c(.261,0,0,0,.030,.017,0,0,.003,-0.0,.004,0,.018,.006,.001,.013),
                   nrow=4,byrow=T)),
       test=T,summary=T)
```

```
## log determinant of S.pooled: -15.42671
## u = 0.01324946
##  M = 218.8517
##  C = 215.9521
##  chi-sq with 20 degrees of freedom is 10.85081
##  reject null hypothesis: TRUE

## [1] 215.95207  10.85081
```

Hence we reject the null hypothesis that $\Sigma_1 = \Sigma_2 = \Sigma_3$. We also notice that the reproduced valued of test statistic $C = 215.9521$ differs from the original value which is 285.5. To investigate this, we include an intermediate value, the logarithm of determinant of $\mathbf{S}_{\text{pooled}}$. Our reproduced value is $-15.42671$, while the textbook has $-15.564$. The standalone computational error should be trivial, but $\ln|\mathbf{S}_{\text{pooled}}|$ is iterated and subtracted 3 times when calculating the value of $M$ (correspondingly, $C$). Therefore, the difference of our test statistic is quite "large", but of course, does not affect our conclusion anyways.

## 2.2 Q2 Trials on Classic Iris Data

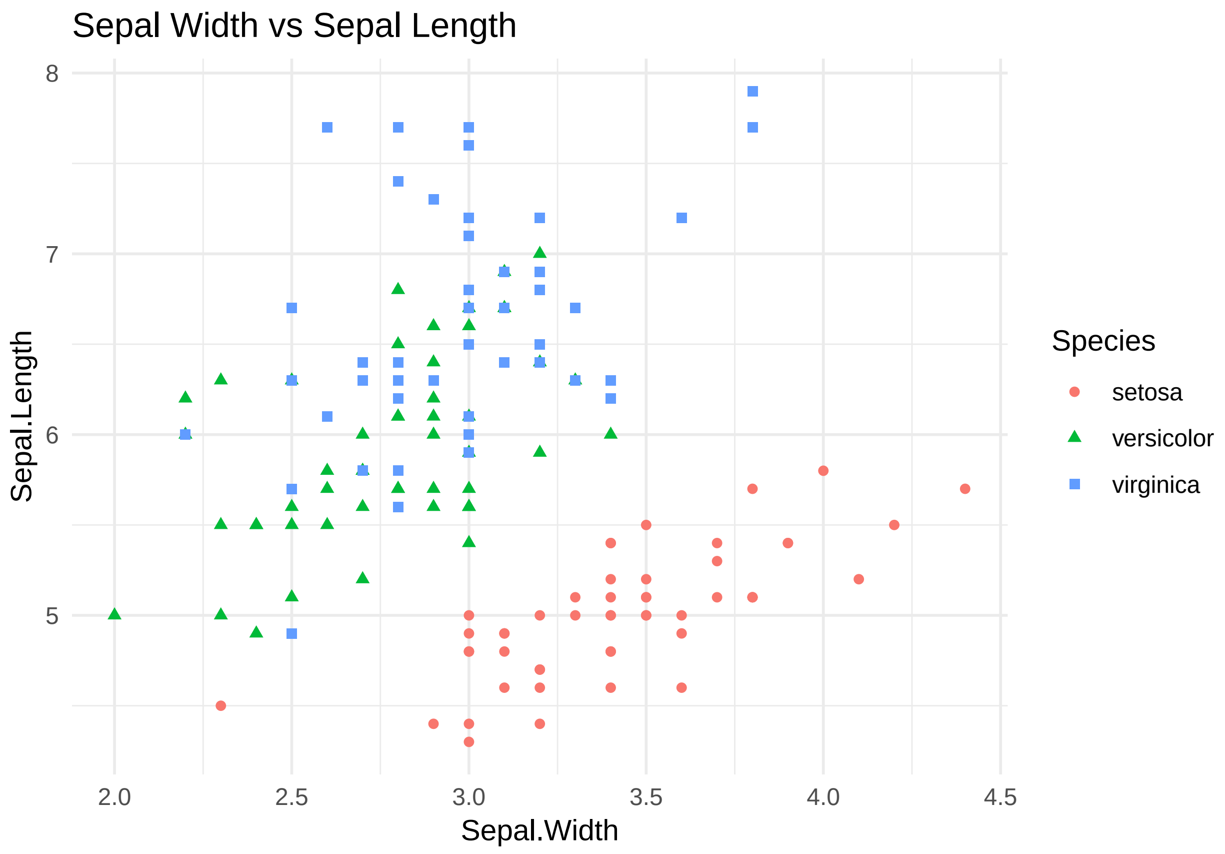
### 2.2.1 Explanatory Data Analysis

For the famous Fisher's Iris data, we still would like to have some exploratory data analysis before starting our covariance homogeneity test. For simplicity, we just take a quick look at the two scatterplots.

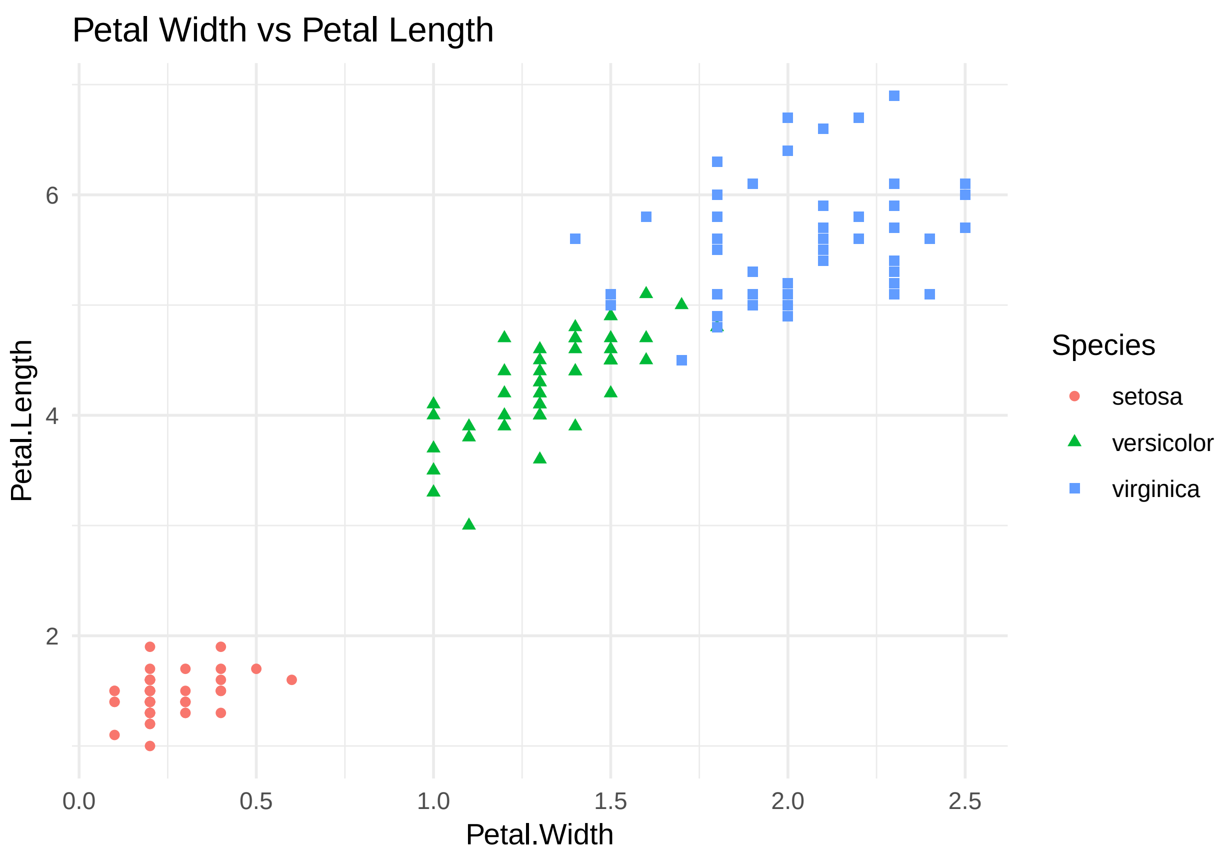
```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
ggplot(data=iris,aes(x=Sepal.Width, y=Sepal.Length)) +
    geom_point(aes(color=Species, shape=Species)) +
    ggtitle("Sepal Width vs Sepal Length") +
    theme_minimal()
```



```
ggplot(data=iris,aes(x=Petal.Width, y=Petal.Length)) +
    geom_point(aes(color=Species, shape=Species)) +
    ggtitle("Petal Width vs Petal Length") +
    theme_minimal()
```



In fact, the variance-covariance matrix between variables are really hard to compare graphically. So no direct conclusion can be drawn at the point.

### 2.2.2 Method 1: Recycle Pre-defined Functions in Q1

The major reason that we implement Box's M Test with functional programming is the ability to recycle and reuse them! Here we only need to separate the data frame and treat them as inputs of the function `M.test()`.

```
versicolor <- iris %>%
    filter(Species=="versicolor") %>%
    dplyr::select(-"Species")
setosa <- iris %>%
    filter(Species=="setosa") %>%
    dplyr::select(-"Species")
virginica <- iris %>%
    filter(Species=="virginica") %>%
    dplyr::select(-"Species")
n1 <- nrow(versicolor)
n2 <- nrow(setosa)
n3 <- nrow(virginica)
n <- nrow(iris)

S1 <- cov(versicolor)
S2 <- cov(setosa)
S3 <- cov(virginica)
```

In this question, we are dealing with two null hypotheses. We do nothing but plug in the values as requested.

Firstly, we test $H_0 : \Sigma_1 = \Sigma_2$.

```
M.test(4,2,c(n1,n2),list(S1,S2),summary=T)
```

```
## u = 0.04387755
##  M = 69.87649
##  C = 66.81048
##  chi-sq with 10 degrees of freedom is 3.940299
##  reject null hypothesis: TRUE
```

```
## [1] 66.810481  3.940299
```

The result of the comparison between test statistic and critical value is obvious, hence we reject $H_0$ in flavor of $\Sigma_1 \neq \Sigma_2$.

Similarly, we test $H_0 : \Sigma_1 = \Sigma_2 = \Sigma_3$ for the second part.

```
M.test(4,3,c(n1,n2,n3),list(S1,S2,S3),summary=T)
```

```
## u = 0.03900227
##  M = 146.6632
##  C = 140.943
##  chi-sq with 20 degrees of freedom is 10.85081
##  reject null hypothesis: TRUE
```

```
## [1] 140.94305  10.85081
```

Again, we reject the null hypothesis.

### 2.2.3 Method 2: Use `heplots` Package

Alternatively, we could directly use `boxM()` function in Michael Friendly's `heplots` package[1], which is a package mainly focuses on visualizing hypothesis tests in multivariate linear models.

What we need to do here is to firstly eliminte `virginica` species and drop the level along our data pipeline, and do a hypothesis test for $\Sigma_1 = \Sigma_2$. After that, we would repeat for testing $\Sigma_1 = \Sigma_2 = \Sigma_3$.

The results are pretty neat:

```
iris2 <- iris %>%
    filter(Species!="virginica") %>%
    droplevels()
res2 <- boxM(iris2[, 1:4], iris2[, "Species"])
res2
```

```
##
##  Box's M-test for Homogeneity of Covariance Matrices
##
## data:  iris2[, 1:4]
## Chi-Sq (approx.) = 66.81, df = 10, p-value = 1.823e-10
```

```
summary(res2)
```

```
## Summary for Box's M-test of Equality of Covariance Matrices
##
## Chi-Sq:   66.81048
## df:   10
## p-value: 1.823e-10
##
## log of Covariance determinants:
##     setosa versicolor     pooled
##  -13.06736  -10.87433  -11.25782
##
## Eigenvalues:
##        setosa  versicolor     pooled
## 1 0.236455690 0.487873944 0.32938906
```

[1] *heplots: Visualizing Hypothesis Tests in Multivariate Linear Models*, https://rdrr.io/rforge/heplots/man/boxM.html

```
## 2 0.036918732 0.072384096 0.07721833
## 3 0.026796399 0.054776085 0.05032393
## 4 0.009033261 0.009790365 0.01008296
##
## Statistics based on eigenvalues:
##                   setosa   versicolor       pooled
## product   2.113088e-06 1.893828e-05 1.290601e-05
## sum       3.092041e-01 6.248245e-01 4.670143e-01
## precision 5.576122e-03 7.338788e-03 7.405502e-03
## max       2.364557e-01 4.878739e-01 3.293891e-01
```

```
res3 <- boxM(iris[, 1:4], iris[, "Species"])
res3
```

```
##
##  Box's M-test for Homogeneity of Covariance Matrices
##
## data:  iris[, 1:4]
## Chi-Sq (approx.) = 140.94, df = 20, p-value < 2.2e-16
```

```
summary(res3)
```

```
## Summary for Box's M-test of Equality of Covariance Matrices
##
## Chi-Sq:   140.943
## df:   20
## p-value: < 2.2e-16
##
## log of Covariance determinants:
##     setosa versicolor  virginica     pooled
## -13.067360 -10.874325  -8.927058  -9.958539
##
## Eigenvalues:
##        setosa  versicolor  virginica     pooled
## 1 0.236455690 0.487873944 0.69525484 0.44356592
## 2 0.036918732 0.072384096 0.10655123 0.08618331
## 3 0.026796399 0.054776085 0.05229543 0.05535235
## 4 0.009033261 0.009790365 0.03426585 0.02236372
##
## Statistics based on eigenvalues:
##                   setosa   versicolor    virginica       pooled
## product   2.113088e-06 1.893828e-05 0.0001327479 4.732183e-05
## sum       3.092041e-01 6.248245e-01 0.8883673469 6.074653e-01
## precision 5.576122e-03 7.338788e-03 0.0169121236 1.304819e-02
## max       2.364557e-01 4.878739e-01 0.6952548382 4.435659e-01
```

There is nothing extra we need to emphasize, since the results are identical to our “home-brewed” version.

## 2.3 Q3 What Could Go Wrong with Box's $\chi^2$ Approximation?

In this problem, we fix the parameter $g$ to be 2 and $p$ changes from 5 to 20 as requested. Since the problem itself does not mention how $n_l$'s behave, we also fix them to be $n_1 = n_2 = 25 > 20$. For each value of $p$, we iterate to resample data for $sim = 1000$ times.

In addition, my personal preference for monitoring complex for-loops is to add a timer keeping track of how long it takes. This will be repeated several times in the next few questions.

```
sim <- 1000
fixed.g <- 2
p.vec <- c(5:15)
n1 <- 25
n2 <- 25

table <- data.frame()

timer <- Sys.time()
for (iter in 1:length(p.vec)) {
    p <- p.vec[iter]

    for (s in 1:sim) {
        X1 <- matrix(rnorm(p*n1),n1,p)
        X2 <- matrix(rnorm(p*n2,mean=0,sd=1),n2,p)
        X2.prime <- matrix(rnorm(p*n2,mean=0,sd=1.5),n2,p)
        S1 <- cov(X1)
        S2 <- cov(X2)
        S2.prime <- cov(X2.prime)

        table <- rbind(table,c(p,M.test(p,fixed.g,c(n1,n2),list(S1,S2)),1))
        table <- rbind(table,c(p,M.test(p,fixed.g,c(n1,n2),list(S1,S2.prime)),2))
    }
}

Sys.time()-timer
```

```
## Time difference of 19.42585 secs
```

```
names(table) <- c("p","TS","chisq","type")

large.summary <- data.frame()
for (pv in p.vec) {
    nu <- 0.5*pv*(pv+1)*(fixed.g-1)

    table.size <- table %>%
        filter(p==pv,type==1)
    table.power <- table %>%
        filter(p==pv,type==2)

    pv.size <- mean(table.size$TS>qchisq(.05,nu,lower.tail=F))
    pv.power <- mean(table.power$TS>qchisq(.05,nu,lower.tail=F))
    large.summary <- rbind(large.summary, c(n1, n2, pv, fixed.g, pv.size, pv.power))
}
names(large.summary) <- c("n1","n2","p","g","size", "power")
```

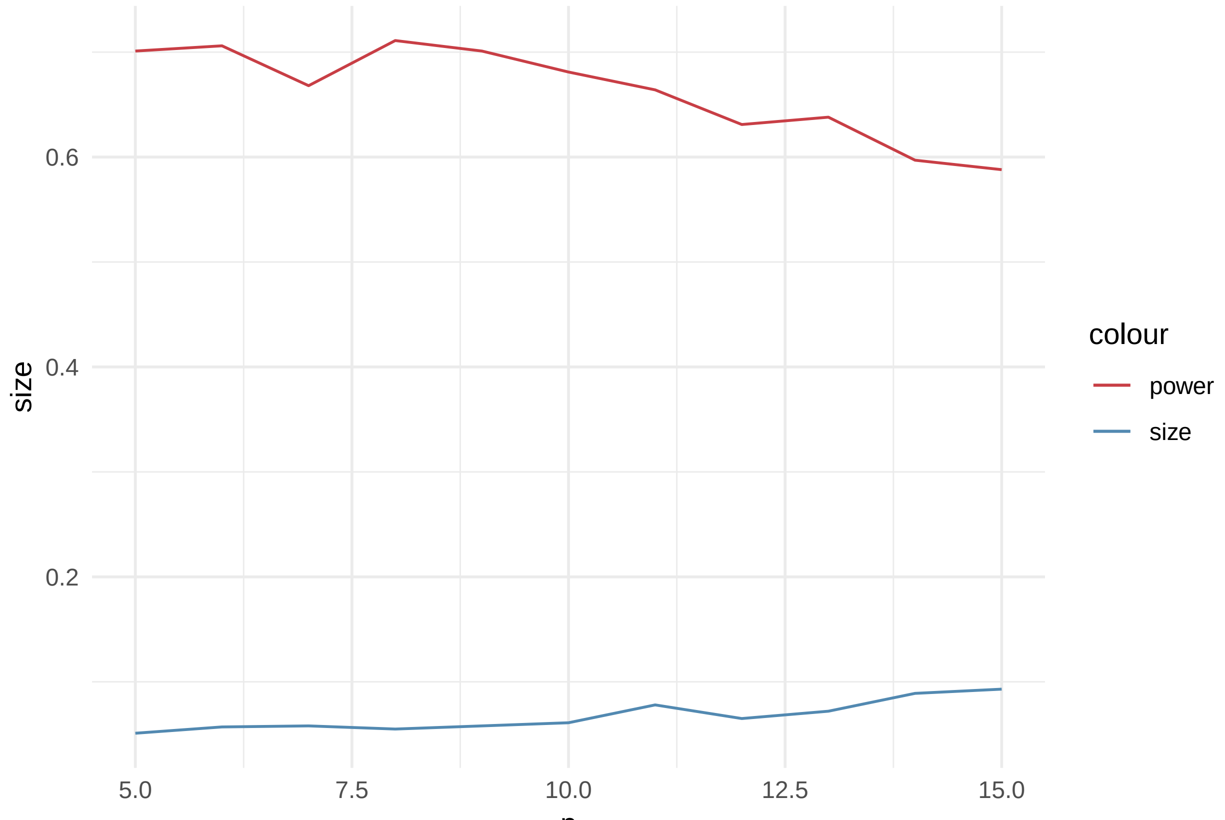
Here we take a look at a summary table of different runs of simulations.

```
large.summary
```

```
##    n1 n2  p g  size power
## 1  25 25  5 2 0.051 0.701
## 2  25 25  6 2 0.057 0.706
## 3  25 25  7 2 0.058 0.668
## 4  25 25  8 2 0.055 0.711
## 5  25 25  9 2 0.058 0.701
## 6  25 25 10 2 0.061 0.681
## 7  25 25 11 2 0.078 0.664
## 8  25 25 12 2 0.065 0.631
## 9  25 25 13 2 0.072 0.638
## 10 25 25 14 2 0.089 0.597
## 11 25 25 15 2 0.093 0.588
```

To understand the performance of simulations, it is better to visualize the size and power changes as a line plot.

```
ggplot(large.summary, aes(x=p)) +
    geom_line(aes(y=size,color="size")) +
    geom_line(aes(y=power,color="power")) +
    scale_colour_manual(values=c("#C83E45", "#5289B1")) +
    theme_minimal()
```



One thing noticable is that, since we have set $n_1 = n_2 = 20$ which are definitely not large enough to be called "infinitely large", our number of parameters $p$ soon catâches up with $n_1$ and $n_2$.

But how badly this performs? The power has a weak start with a value around 0.7 and gradually decreases to a more unpleasant range. On the other hand, the size is generally stable but not always significant ($< .05$). Overall, we would suggest that the Box's $\chi^2$ approximation performs poorly when $p > 5$ while holding $g$ fixed below 5.

## 2.4 Q4 Testing Covariance Matrices WITHOUT Adjustments

Besides the required dimension p = 5, 10, 50, 100, 300, we add an additional 375 for comparison.

```r
n <- 500
p.vec <- c(5, 10, 50, 100, 300, 375)
N <- n - 1
sim <- 1000

table <- data.frame()

timer <- Sys.time()

for (i in 1:length(p.vec)) {
    p <- p.vec[i]
    mu <- rep(0, p)
    Sigma <- diag(p)
    Sigma2 <- diag(.05,p)
    Sigma2[1,1] <- 1

    for (s in 1:sim) {
        X <- rmvnorm(n, mu, Sigma)
        S <- cov(X)

        rho <- 1-(2*p**2+3*p-1)/(6*(n-1)*(p+1))
        ts <- -2*rho*(0.5*p*N + (0.5*N) *
                          (unlist(determinant(S,logarithm=T))[[1]]-sum(diag(S))))

        X2 <- rmvnorm(n, mu, Sigma2)
        S2 <- cov(X2)
        ts2 <- -2*rho*(0.5*p*N + (0.5*N) *
                           (unlist(determinant(S2,logarithm=T))[[1]]-sum(diag(S2))))

        table <- rbind(table,c(p,ts,1))
        table <- rbind(table,c(p,ts2,2))
    }
}

Sys.time()-timer
```

```
## Time difference of 12.51302 mins
```

```r
names(table) <- c("p","ts","type")

large.summary <- data.frame()
for (pv in p.vec) {
    table.size <- table %>%
        filter(p==pv,type==1)
    table.power <- table %>%
        filter(p==pv,type==2)

    f <- 0.5*pv*(pv+1)
    gamma2 <- pv*(2*pv**4+6*pv**3+pv**2-12*pv-13)/(288*(pv+1))
    rho <- 1-(2*pv**2+3*pv-1)/(6*(n-1)*(pv+1))

    pv.size <- mean(table.size$ts>(
        qchisq(.95,f)+gamma2/rho**2/N**2*(qchisq(.95,f+4)-qchisq(.95,f))))
    pv.power <- mean(table.power$ts>(
        qchisq(.95,f)+gamma2/rho**2/N**2*(qchisq(.95,f+4)-qchisq(.95,f))))
    large.summary <- rbind(large.summary, c(pv, pv.size, pv.power))
}
names(large.summary) <- c("p", "size", "power")
```

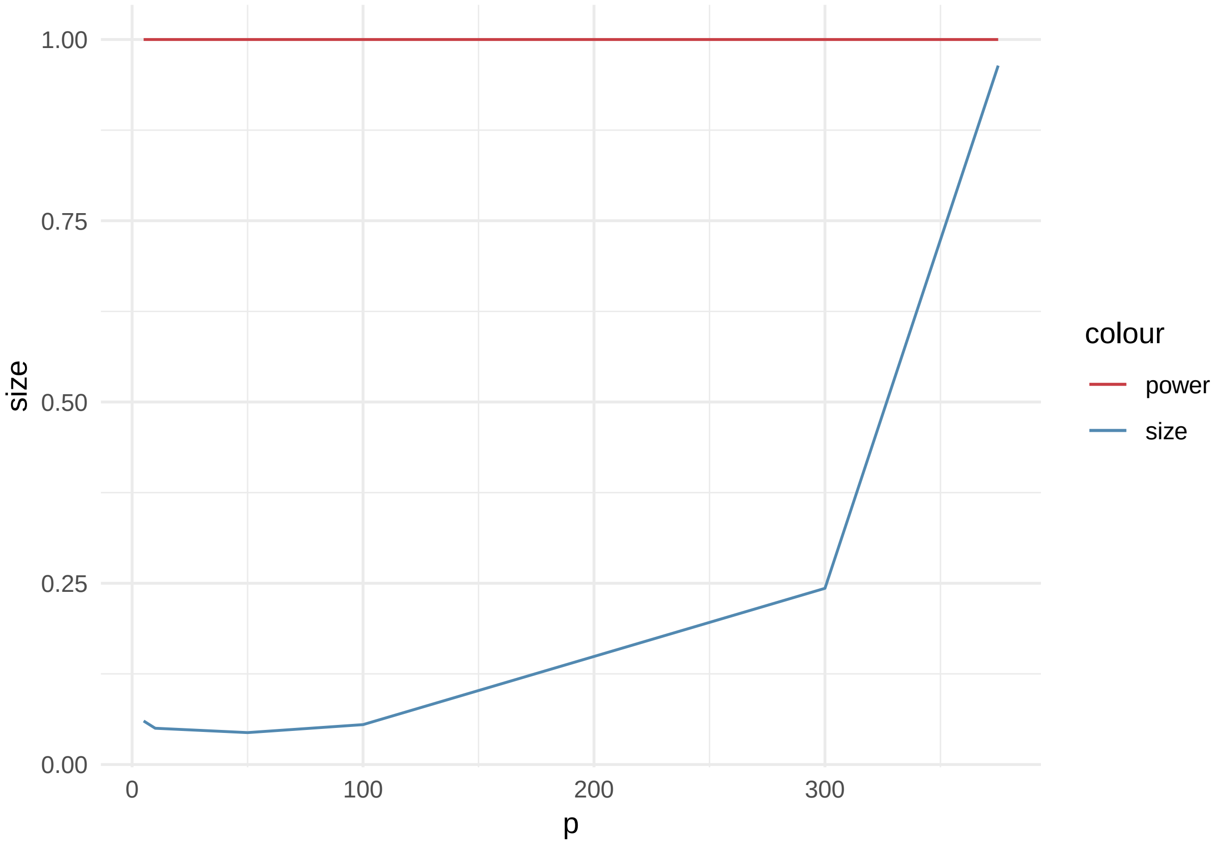
Summary:

```r
large.summary
```

```
##     p  size power
## 1   5 0.060     1
## 2  10 0.050     1
## 3  50 0.044     1
## 4 100 0.055     1
## 5 300 0.243     1
## 6 375 0.964     1
```

The line plot of summary table above:

```r
ggplot(large.summary, aes(x=p)) +
    geom_line(aes(y=size,color="size")) +
    geom_line(aes(y=power,color="power")) +
    scale_colour_manual(values=c("#C83E45", "#5289B1")) +
    theme_minimal()
```



The power is good. And the size is good at the beginning, but when $n$ gets really large, size tends to 1.

Note that our sample size is 500 which is not a small number, so when dimension $p$ gets large, modern RMT indicates that the likelihood ratio statistic drifts to infinity almost surely. Therefore, the classicial $\chi^2$ approximation may lead us to many false rejections of $H_0$ in case of high-dimensional data (Bai, Jiang, Yao, Zheng, 2009).

## 2.5 Q5 Why Adjustment?

**Proof:**

$$\begin{aligned}T_1 &= \mathrm{tr}\mathbb{S} - \log|\mathbb{S}| - p \\ &= p \cdot F^{\mathbb{S}}(f) \\ &= p \cdot \int (x - \log x - 1) dF^{y_N}(x) \\ &= p \cdot \int f(x) d(F^{\mathbb{S}}(x) - F^{y_N}(x)) + p \cdot F^{y_N}(f)\end{aligned}$$

Since $N = n-1, y_N = p/N$:

$$\begin{aligned}F^{y_N}(f) &= 1 - \frac{y_N - 1}{y_N}\log(1 - y_N) \\ &\doteq 1 - \frac{y-1}{y}\log(1-y) \\ &= 1 + \frac{1-y}{y}\log(y-1) \\ &= d_1(y)\end{aligned}$$

By Theorem in lecture notes,

$$T_1 - p \cdot F^{y_N}(f) \doteq T_1 - p \cdot d_1(y_N)$$

should weakly converges to a Gaussian vector with the mean

$$\mu_1 = -\frac{1}{2}\log(1-y),$$

and variance

$$\sigma_1^2 = -2\log(1-y) - 2y.$$

## 2.6 Q6 Testing Covariance Matrices WITH adjustments

This question is a refined version of Q4. An improvement has been made based on the theory of Q5.

Note that we include a case with $p = 375$ for consistency as well.

```r
n <- 500
p.vec <- c(5, 10, 50, 100, 300, 375)
N <- n - 1
sim <- 1000

table <- data.frame()

timer <- Sys.time()

for (i in 1:length(p.vec)) {
    p <- p.vec[i]
    mu <- rep(0, p)
    Sigma <- diag(p)
    Sigma2 <- diag(.05,p)
    Sigma2[1,1] <- 1

    for (s in 1:sim) {
        y <- p/n
        YN <- p/N
        d1 <- 1+(1-YN)/YN*log(1-YN)

        X <- rmvnorm(n, mu, Sigma)
        S <- cov(X)
        T1 <- sum(diag(S))-unlist(determinant(S,logarithm=T))[[1]]-p

        ts <- T1-p*d1

        X2 <- rmvnorm(n, mu, Sigma2)
        S2 <- cov(X2)
        T1.2 <- sum(diag(S2))-unlist(determinant(S2,logarithm=T))[[1]]-p

        ts.2 <- T1.2-p*d1

        table <- rbind(table,c(p,ts,1))
        table <- rbind(table,c(p,ts.2,2))
    }
}

Sys.time()-timer
```

```
## Time difference of 12.62086 mins
```

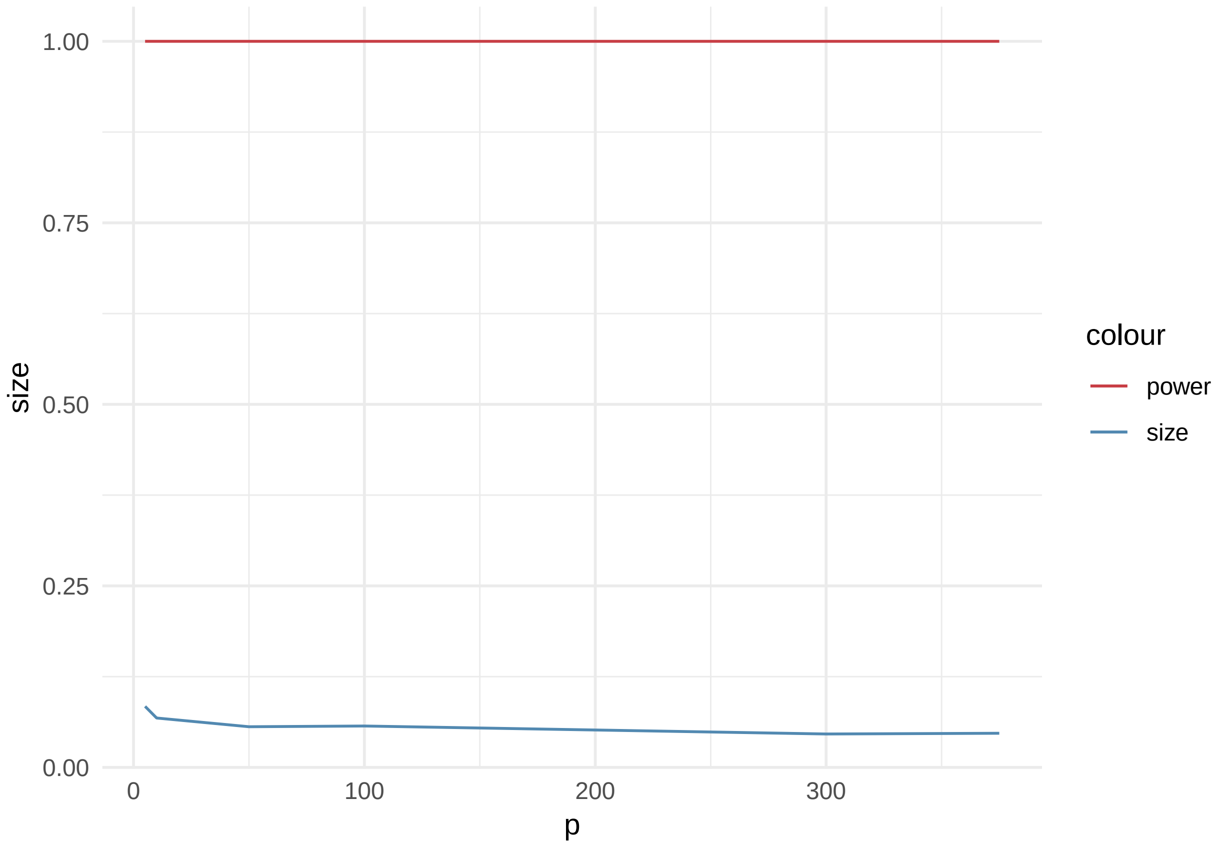
```r
names(table) <- c("p","ts","type")

large.summary <- data.frame()

for (pv in p.vec) {
    y <- pv/n
    mu1 <- -0.5*log(1-y)
    sigma1 <- sqrt(-2*log(1-y)-2*y)

    table.size <- table %>%
        filter(p==pv,type==1)
    table.power <- table %>%
        filter(p==pv,type==2)

    pv.size <- mean(table.size$ts>qnorm(.95,mu1,sigma1))
    pv.power <- mean(table.power$ts>qnorm(.95,mu1,sigma1))
    large.summary <- rbind(large.summary, c(pv, pv.size, pv.power))
}
names(large.summary) <- c("p", "size", "power")
```

Check out the table again:

```
large.summary
```

```
##     p  size power
## 1   5 0.084     1
## 2  10 0.068     1
## 3  50 0.056     1
## 4 100 0.057     1
## 5 300 0.046     1
## 6 375 0.047     1
```

```
ggplot(large.summary, aes(x=p)) +
    geom_line(aes(y=size,color="size")) +
    geom_line(aes(y=power,color="power")) +
    scale_colour_manual(values=c("#C83E45", "#5289B1")) +
    theme_minimal()
```



This time, the size converges to 0.05 as expected, which in comparison, is of course an improvement from Q4.

# 3 Multivariate Time Series

## 3.1 Q7 MTS Basics

### 3.1.1 (a)-(d) MTS Generation and Visulization

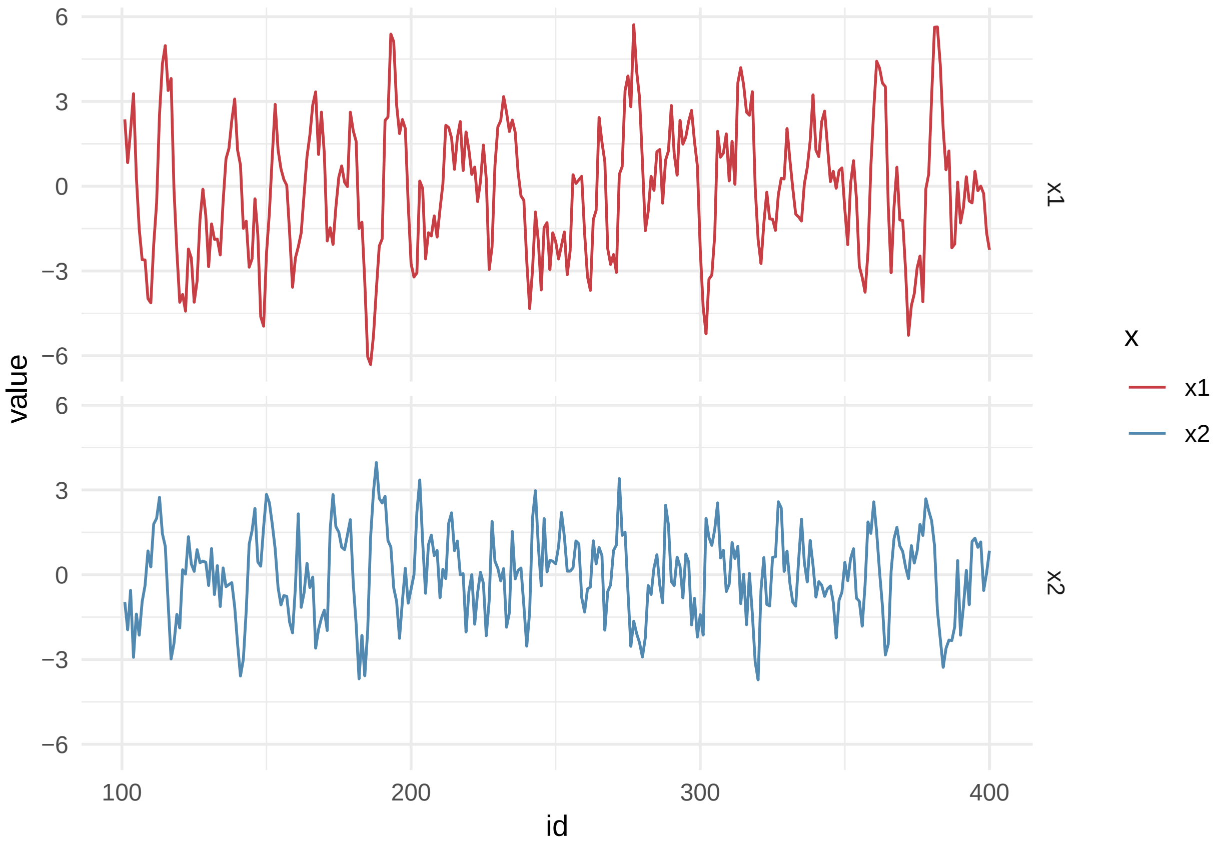
For this question, my multivariate first order VAR/VMA model generating function might seem a little bit complex due to the introduction of multiple "flag" arguments. To be specific,

- `viz` controls if a time series plot is needed in the output.
- `lag` controls the upper limit of $\tau$.
- `add.S0` controls if a lag-0 autocovariance matrix is added to the "correlation matrix list", since it is also required for the portmanteau test.
- `burnin` is the number of discarded observations. By default, we include the first 100 observations as burn-ins. For example, if the required observation number is 300, we generate 300+100 such observations and discard the first 100 accordingly.
- `type` is set to be `lag` by default for part (a), (b), (c) and (d). In this case, we are storing observations into the data frame `df`. But if it is set to be `qm`, i.e. when we conduct a portmanteau test, we are operating on residuals of observations, so that the data frame `df` will be assigned with residuals instead.

```r
mts.gen.1 <- function(matrix.A, matrix.Sig, obs, method, viz=T, lag=0,
                      add.S0=F, burnin=100, type="lag") {
    df <- data.frame()
    if (method=="var") {
        epsilons <- rmvnorm((obs+burnin), c(0,0), matrix.Sig)
        Xt <- c(0,0)
        for (t in 1:(obs+burnin)) {
            Xt <- matrix.A%*%Xt + epsilons[t,]
            df <- rbind(df, c(Xt))
        }
        df <- df[(1+burnin):(obs+burnin),]
    } else if (method=="vma" || method=="ma") {
        epsilons <- rmvnorm(obs+burnin+1,c(0,0),matrix.Sig)
        for (t in 1:(obs+burnin)) {
            Xt <- epsilons[t+1,] + matrix.A%*%epsilons[t,]
            df <- rbind(df, c(Xt))
        }
        df <- df[(1+burnin):(obs+burnin),]
    }
    names(df) <- c("x1","x2")
    df$id <- (1+burnin):(obs+burnin)
    df2 <- melt(df, id.var = "id", variable.name = "x")
    if (viz) {
        print(ggplot(df2,aes(x=id,y=value)) +
                  geom_line(aes(color=x)) +
                  facet_grid(x ~ .) +
                  scale_colour_manual(values=c("#C83E45", "#5289B1")) +
                  theme_minimal())
    }
    if (lag>0) {
        if (type=="qm") {
            # if we conduct a portmanteau test on residuals,
            # reassign the residual df to current variable df
            mod <- VAR(df,p=1,lag.max=20)
            df <- as.data.frame(cbind(mod$varresult$x1$residuals,mod$varresult$x2$residuals))
            names(df) <- c("x1","x2")
        }
        rho.tau <- list()
        X.bar <- c(mean(df$x1), mean(df$x2))
        S0 <- cov(df[,1:2])
        D <- diag(1/sqrt(diag(S0)))

        # also record lag-0 autocovariance matrix, need to be used when doing portmanteau test
        if (add.S0) {
            rho.tau[[lag+1]] <- cov2cor(S0)  # the same as D %*% S0 %*% D
        }

        for (tau in 1:lag) {
            Stau <- 0
            for (t in (tau+1):nrow(df)) {
                Stau <- Stau + (c(df$x1[t], df$x2[t]) - X.bar) %*%
                    t(c(df$x1[t-tau],df$x2[t-tau]) - X.bar)
            }
            Stau <- Stau / (nrow(df)-1)
            rho.tau[[tau]] <- D %*% Stau %*% D
        }
        return(rho.tau)
    }
}

## (a) & (b) & (c) & (d)
A <- matrix(c(0.8,0.4,-0.3,0.6),byrow = T,ncol=2)
Sigma <- matrix(c(2,0.5,0.5,1),byrow = T,ncol=2)
```

Part (a) and (b) with time series visualization and its lag-5 correlation matrices.

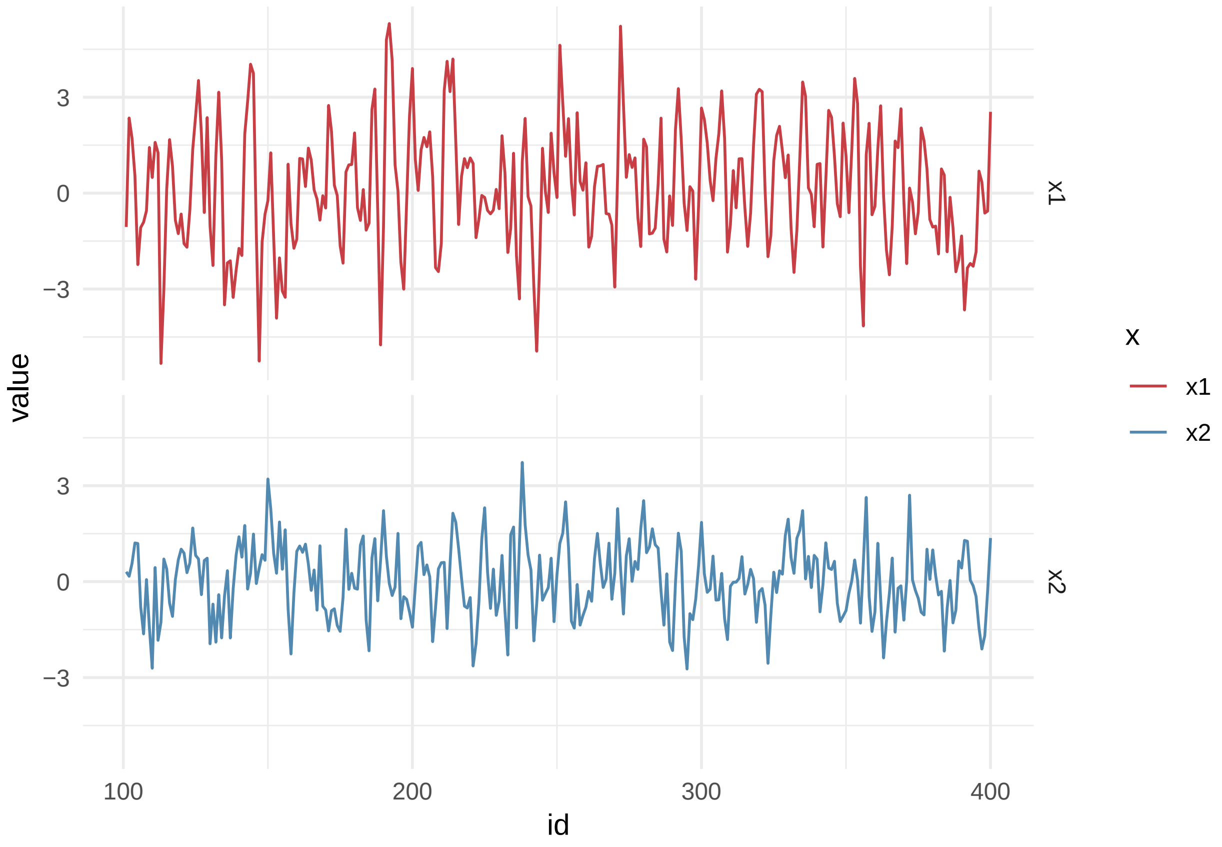
```r
mts.gen.1(A, Sigma, 300, "var", viz=T, lag=5, add.S0=F, burnin=100, type="lag")
```



```
## [[1]]
##            [,1]      [,2]
## [1,]  0.7859249 0.1526886
## [2,] -0.4931342 0.6202365
##
## [[2]]
##            [,1]      [,2]
## [1,]  0.5184852 0.2744502
## [2,] -0.6265335 0.3241870
##
## [[3]]
##            [,1]      [,2]
## [1,]  0.2670795 0.3394800
## [2,] -0.6097292 0.1038583
##
## [[4]]
##             [,1]        [,2]
## [1,]  0.02212521  0.34494099
## [2,] -0.51048627 -0.06452215
##
## [[5]]
##            [,1]       [,2]
## [1,] -0.1600059  0.2918388
## [2,] -0.3385234 -0.2013264
```

Part (c) and (d) with time series visualization and its lag-2 correlation matrices.

```r
mts.gen.1(A, Sigma, 300, "vma", viz=T, lag=2, add.S0=F, burnin=100, type="lag")
```



```
## [[1]]
##            [,1]      [,2]
## [1,]  0.4739894 0.2453092
## [2,] -0.2099934 0.3844573
##
## [[2]]
##              [,1]        [,2]
## [1,] -0.07149522 -0.09460774
## [2,] -0.04072408  0.01342928
```

### 3.1.2 (e) A Replication of Li and McLeod's Table

In this question, all parameters are set to be the values used in the paper (Li, McLeod, 1981). Specifically,

$$A = \begin{pmatrix} -0.2 & 0.3 \\ -0.6 & 1.1 \end{pmatrix}, B = \begin{pmatrix} 0.4 & 0.1 \\ -1 & 0.5 \end{pmatrix}, C = \begin{pmatrix} -1.5 & 1.2 \\ -0.9 & 0.5 \end{pmatrix}.$$

```
A <- matrix(c(-.2, .3, -.6, 1.1), byrow = T, ncol=2)
B <- matrix(c(.4, .1, -1, .5), byrow = T, ncol=2)
C <- matrix(c(-1.5, 1.2, -.9, .5), byrow = T, ncol=2)
k <- 2 # 2 by 2 matrices
m <- 20
n <- 200
sim <- 1000

alpha.vec <- c(.25, -.25, .5, -.5, .75, -.75)

Delta.matrix <- function(alpha.value) {
    return(matrix(c(1, alpha.value, alpha.value, 1), byrow = T, ncol=2))
}

# kronecker(), vec() in `ks` package

inside.Qm <- function(rhol,rho0) {
    temp <- t(vec(t(rhol))) %*% kronecker(solve(rho0), solve(rho0)) %*% vec(t(rhol))
    return(unlist(temp)[[1]])
}

summary.table <- data.frame()

timer <- Sys.time()

for (i in 1:sim) {
    for (alpha in alpha.vec) {
        delta.matrix <- Delta.matrix(alpha)
        rho1 <- mts.gen.1(A, delta.matrix, n, "var", viz=F, lag=m, add.S0=T, type="qm")
        rho2 <- mts.gen.1(B, delta.matrix, n, "var", viz=F, lag=m, add.S0=T, type="qm")
        rho3 <- mts.gen.1(C, delta.matrix, n, "var", viz=F, lag=m, add.S0=T, type="qm")
        QA <- 0
        QB <- 0
        QC <- 0
        for (j in 1:m) {
            QA <- QA + inside.Qm(rho1[[j]], rho1[[m+1]])
            QB <- QB + inside.Qm(rho2[[j]], rho2[[m+1]])
            QC <- QC + inside.Qm(rho3[[j]], rho3[[m+1]])
        }
        QA <- n * QA
        QAs <- QA + k**2*m*(m+1)/(2*n)
        QB <- n * QB
        QBs <- QB + k**2*m*(m+1)/(2*n)
        QC <- n * QC
        QCs <- QC + k**2*m*(m+1)/(2*n)
        summary.table <- rbind(summary.table,
                               c(alpha, QA, QAs, QB, QBs, QC, QCs))
    }
}
Sys.time() - timer
```

```
## Time difference of 46.25538 mins
```

```
names(summary.table) <- c("alpha", "QA", "QAs", "QB", "QBs", "QC", "QCs")

critical <- qchisq(.05,76,lower.tail = F)  # k^2(m-p-q) = 4 * (20 - 1 - 0) = 76

summary.table %>%
    mutate(QA = QA>critical,
           QAs = QAs>critical,
           QB = QB>critical,
           QBs = QBs>critical,
           QC = QC>critical,
           QCs = QCs>critical) %>%
    group_by(alpha) %>%
    summarize(
        A.Q20 = sum(QA),
        A.Q20.star = sum(QAs),
        B.Q20 = sum(QB),
        B.Q20.star = sum(QBs),
        C.Q20 = sum(QC),
        C.Q20.star = sum(QCs)
    )
```

```
## # A tibble: 6 x 7
##   alpha A.Q20 A.Q20.star B.Q20 B.Q20.star C.Q20 C.Q20.star
##   <dbl> <int>      <int> <int>      <int> <int>      <int>
## 1 -0.75    30         47    23         41    29         49
## 2 -0.5     26         59    30         54    28         58
## 3 -0.25    26         45    26         54    28         49
## 4  0.25    28         58    19         41    27         49
## 5  0.5     31         52    21         42    33         55
## 6  0.75    23         53    23         46    32         60
```

**Note 1**: The original table in paper (Li, McLeod, 1981) might have a serious typo, where "per cent" should actually be "count". Consider a test with typo 1 error about 50%. Since the simulation run has a valeu of 1000 instead of 100, I guess this in fact is the "count", which also explains why $Q^*_m$ is better in the perspective of "closer to 0.05".

**Note 2**: The difference between our simulation and the original table might originate from the fact that we discard 100 observations in the implented function as burnin, while the paper didn't specifies their approach or their particular parameter.

**Note 3**: Our algorithm has an approximate 30 minutes' runtime, which is not ideal. There is still some space for improvement.

#### 3.1.3 (f) An Experiment on Federal Debts Data

Professor McLeod also created a package called `portes`[2] which is perfectly suitable for conducting various hypothesis testing based on various test statistics.

[2] *portes: Portmanteau Tests for Univariate and Multivariate Time Series Models*, https://rdrr.io/cran/portes/

```
debt <- read.table("q-fdebt.txt",header = T) %>%
    dplyr::select(hbfin, hbfrbn, hbpun) %>%
    mutate(hbfin = log(hbfin), hbfrbn = log(hbfrbn), hbpun = log(hbpun))
debt <- as.matrix(debt)
DiffData <- matrix(numeric(ncol(debt) * (nrow(debt)-1)), ncol = ncol(debt))
for (i in 1:ncol(debt)) DiffData[, i] <- diff(debt[, i], lag = 1)
Fit <- ar.ols(DiffData, aic=F, order.max = 1)
```

In `portes` package, we could conduct hypothesis testing theoretically or using Monte Carlo simulation, here we include both versions.

```
# theoretical
LiMcLeod(Fit, lags=10)
```

```
##  lags statistic df      p-value
##    10  220.9151 81 6.550316e-15
```

```
# Monte Carlo version
portest(Fit, lags=10, test="LiMcLeod", ncores=4)
```

```
##         lags    statistic      p-value
## 1.000000e+01 2.209151e+02 9.990010e-04
```

The results are the same, i.e. reject null hypothesis in favor of believing at least one of $\rho_\tau \neq 0$ for $\tau \in \{1, \dots, 10\}$.

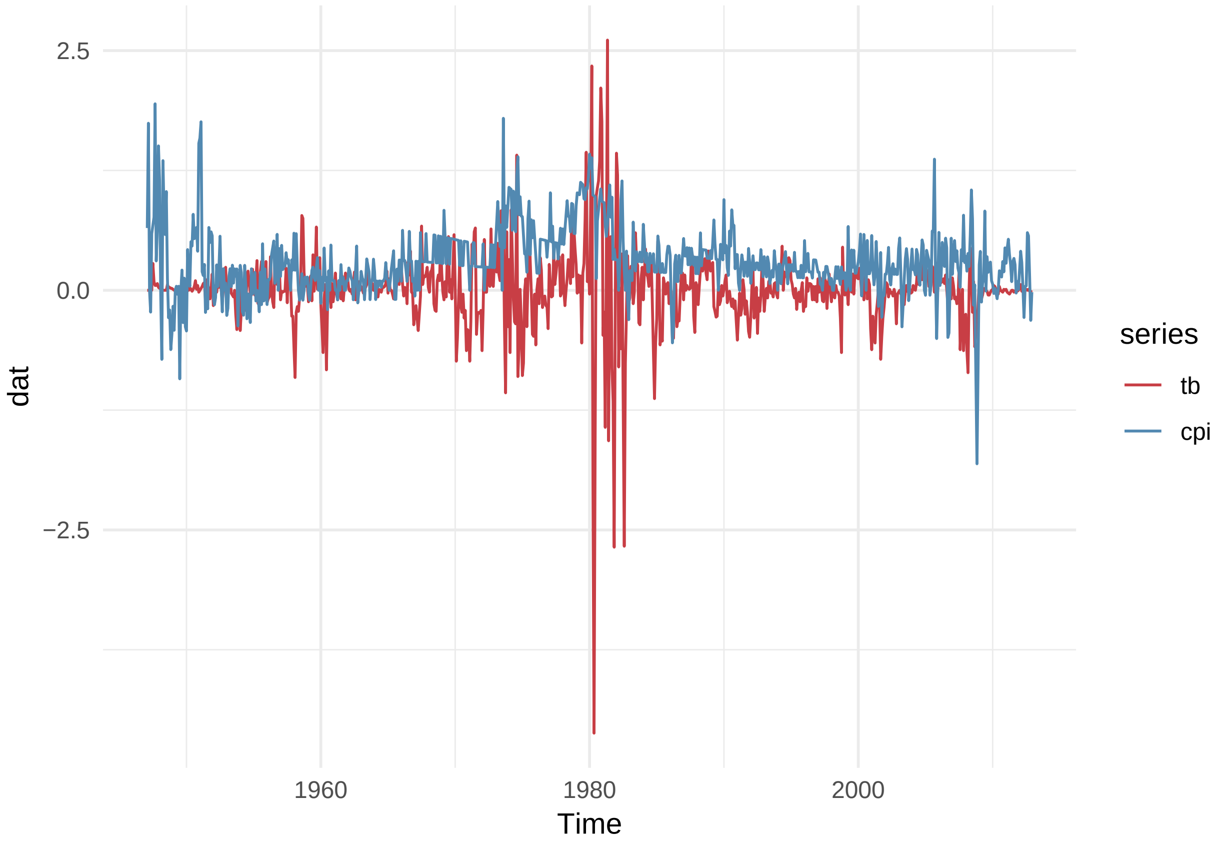
## 3.2 Q8 VAR Model Order Selection on CPI Data

### 3.2.1 (a) Plotting CPI Data

Unlike Q7 part (f), this question requires more complex transformation on CPI index.

$$CPI_{\text{rate}} = 100 \times (\log(CPI_{i+1}) - \log(CPI_i))$$

```r
dat <- read.table("m-cpitb3m.txt",header = T) %>%
    dplyr::select(tb=tb3m, cpi=cpiaucsl) %>%
    mutate(cpi=100*log(cpi))

dat <- diff(ts(dat, frequency = 12,start = c(1947,1)))

autoplot(dat) +
    scale_colour_manual(values=c("#C83E45", "#5289B1")) +
    theme_minimal()
```



### 3.2.2 (b) The Selection Algorithm of $m$

**Note:** The problem itself might have a small typo/inconsistency in notations where the number of variables is $k$ in some sentences but $p$ in others. In order to maintain the consistency in coding, we use `k` as this parameter.

A shortcut to achieve the goal of this part is to build a function that automatically selects such order $m$ based on some selecting rule(s). In package `MTS`[3], `VARorder()` function is a good option. However, we simplified it by removing the order selection criteria like AIC and BIC, only keeping the Bartlett's approximation as the test statistic and the calculated p-value.

[3] *MTS: All-Purpose Toolkit for Analyzing Multivariate Time Series (MTS) and Estimating Multivariate Volatility Models*, https://rdrr.io/cran/MTS/man/VARorder.html

At the end of the next function `VARorder2()`, we implemented our own version of $m$ selection, which is based on the description in Step 5 of this problem.

```r
VARorder2 <- function (x, P = 10) {
    x1 <- as.matrix(x)
    T <- nrow(x1)
    k <- ncol(x1)
    if (P < 1) {P = 1}
    obs = T - P
    y = x1[(P + 1):T, , drop = FALSE]
    ist = P + 1
    xmtx = cbind(rep(1, obs), x1[P:(T - 1), ])
    if (P > 1) {
        for (i in 2:P) {
            xmtx = cbind(xmtx, x1[(ist - i):(T - i), ])
        }
    }
    chidet = rep(0, (P + 1))
    s = cov(y) * (obs - 1)/obs
    chidet[1] = log(det(s))
    y = as.matrix(y)
    for (l in 1:P) {
        idm = k * l + 1
        xm = xmtx[, 1:idm]
        xm = as.matrix(xm)
        xpx <- crossprod(xm, xm)
        xpy <- crossprod(xm, y)
        beta <- solve(xpx, xpy)
        yhat <- xm %*% beta
        resi <- y - yhat
        sse <- crossprod(resi, resi)/obs
        d1 = log(det(sse))
        chidet[l + 1] = d1
    }
    Mstat = rep(0, P)
    pv = rep(0, P)
    for (j in 1:P) {
        Mstat[j] = (T - P - k * j - 1.5) * (chidet[j] - chidet[j + 1])
        pv[j] = 1 - pchisq(Mstat[j], k ** 2)
    }

    output = cbind(c(0:P), c(0, Mstat), c(0, pv))
    colnames(output) <- c("l", "M(l)", "p-value")

    # which m? i.e. starting from this l=m, the p-value of following models
    # should all be greater than 0.05

    m.output <- 1
    for (m in 1:nrow(output)) {
        if (all(output[m:nrow(output),3] >= 0.05)) {
            m.output <- m
            break()
        }
    }

    print(round(output, 3))
    if (output[nrow(output),3] < 0.05) {
        m.output <- nrow(output)
    }
    cat("Select VAR (", m.output-1, ")")
}
```

Recall the first step of this order selection methodology, $P$ is a selected positive order that we would like to consider as the maximum order. We just set it to be 20 and 50.

```
VARorder2(dat, 20)
```

```
##        l    M(l) p-value
##  [1,]  0   0.000   0.000
##  [2,]  1 467.936   0.000
##  [3,]  2  68.585   0.000
##  [4,]  3   8.899   0.064
##  [5,]  4  18.671   0.001
##  [6,]  5  22.484   0.000
##  [7,]  6  52.560   0.000
##  [8,]  7  12.815   0.012
##  [9,]  8  24.420   0.000
## [10,]  9  16.047   0.003
## [11,] 10   8.168   0.086
## [12,] 11   3.329   0.504
## [13,] 12  33.928   0.000
## [14,] 13  13.295   0.010
## [15,] 14   3.527   0.474
## [16,] 15  10.519   0.033
## [17,] 16  14.753   0.005
## [18,] 17   2.329   0.676
## [19,] 18   0.365   0.985
## [20,] 19  12.226   0.016
## [21,] 20  12.308   0.015
## Select VAR ( 20 )
```

```
VARorder2(dat, 50)
```

```
##        l    M(l) p-value
##  [1,]  0   0.000   0.000
##  [2,]  1 441.733   0.000
##  [3,]  2  66.496   0.000
##  [4,]  3   9.399   0.052
##  [5,]  4  25.067   0.000
##  [6,]  5  19.022   0.001
##  [7,]  6  55.022   0.000
##  [8,]  7  19.353   0.001
##  [9,]  8  21.604   0.000
## [10,]  9  23.523   0.000
## [11,] 10  15.876   0.003
## [12,] 11   5.698   0.223
## [13,] 12  23.276   0.000
## [14,] 13  13.020   0.011
## [15,] 14   8.099   0.088
## [16,] 15  19.250   0.001
## [17,] 16  13.831   0.008
## [18,] 17   5.467   0.243
## [19,] 18   2.474   0.649
## [20,] 19  13.430   0.009
## [21,] 20  11.943   0.018
## [22,] 21   2.940   0.568
## [23,] 22   1.783   0.776
## [24,] 23   3.090   0.543
## [25,] 24  12.153   0.016
## [26,] 25   8.388   0.078
## [27,] 26   4.946   0.293
## [28,] 27   1.894   0.755
## [29,] 28   1.726   0.786
## [30,] 29   8.431   0.077
## [31,] 30   0.724   0.948
## [32,] 31  10.173   0.038
## [33,] 32   1.822   0.768
## [34,] 33   0.308   0.989
## [35,] 34   0.291   0.990
## [36,] 35   5.653   0.227
## [37,] 36   2.444   0.655
## [38,] 37   0.444   0.979
## [39,] 38   0.866   0.929
## [40,] 39   6.145   0.189
## [41,] 40   6.934   0.139
## [42,] 41   0.881   0.927
## [43,] 42   0.892   0.926
## [44,] 43   4.728   0.316
## [45,] 44   0.774   0.942
## [46,] 45   8.692   0.069
## [47,] 46   2.158   0.707
## [48,] 47   2.106   0.716
## [49,] 48   3.445   0.486
## [50,] 49   3.054   0.549
## [51,] 50   2.979   0.561
## Select VAR ( 32 )
```

We didn't run the algorithm with hundreds of different values of $P$. We totally understand that the result of selected $m$ depends on the $P$ we would like to consider. But seriously, for small $P$, say we have $P$ around 10, the $m$ value is pretty much close to $P$. But if $P$ becomes large enough, $m$ would tend to a stable value.

### 3.2.3 (c) The Curse of Dimensionality

The intuition of this part is to increase the dimensionality of multivariate time series object and run the previous $m$-selection algorithm in part (b). The method we choose to increase the dimensionality is "bootstraping" by resampling the given two variates into new variates.

The `increase.dim()` function we defined here will automatically increase the dimension to $p$ then run the order selection algorithm. Out target is to see what is happening when $p$ keeps growing. We will set $p$ to be $10, 25, 50$.

```r
library(tsDyn)
increase.dim <- function(data, p) {
    dat.p <- dat
    # increase dimension up to 3 or more (upper limit p)
    for (dim.p in 3:p) {
        dat.p <- cbind(dat.p, sample(dat[,sample(1:2, 1, prob=c(0.5,0.5))],
                                     nrow(dat), replace=T))
    }
    dim(dat.p)
    mod <- lineVar(dat.p, lag=1)
    new.mod <- VAR.boot(mod, "resample")
    VARorder2(new.mod, P=20)
}

increase.dim(dat, 10)
```

```
##         l    M(l) p-value
##  [1,]  0   0.000   0.000
##  [2,]  1 608.674   0.000
##  [3,]  2  89.173   0.773
##  [4,]  3  98.026   0.537
##  [5,]  4  80.760   0.921
##  [6,]  5  78.809   0.942
##  [7,]  6  93.331   0.668
##  [8,]  7  89.448   0.766
##  [9,]  8 110.501   0.222
## [10,]  9 109.152   0.250
## [11,] 10  89.639   0.762
## [12,] 11 105.577   0.332
## [13,] 12  89.135   0.773
## [14,] 13 124.502   0.049
## [15,] 14 124.377   0.050
## [16,] 15 109.658   0.239
## [17,] 16 101.651   0.435
## [18,] 17  92.212   0.698
## [19,] 18  89.250   0.771
## [20,] 19 114.667   0.150
## [21,] 20  86.146   0.837
## Select VAR ( 15 )
```

```
increase.dim(dat, 25)
```

```
##         l     M(l) p-value
##  [1,]  0    0.000   0.000
##  [2,]  1 1642.306   0.000
##  [3,]  2  552.896   0.982
##  [4,]  3  579.420   0.904
##  [5,]  4  597.672   0.778
##  [6,]  5  637.358   0.357
##  [7,]  6  647.049   0.263
##  [8,]  7  705.242   0.014
##  [9,]  8  639.134   0.339
## [10,]  9  623.394   0.511
## [11,] 10  665.047   0.130
## [12,] 11  604.374   0.716
## [13,] 12  568.839   0.947
## [14,] 13  625.116   0.491
## [15,] 14  693.473   0.029
## [16,] 15  584.201   0.877
## [17,] 16  604.500   0.715
## [18,] 17  565.845   0.956
## [19,] 18  680.880   0.060
## [20,] 19  616.604   0.587
## [21,] 20  598.119   0.774
## Select VAR ( 15 )
```

```
increase.dim(dat, 50)  # computationally singular
```

```
## Error in solve.default(xpx, xpy): system is computationally singular: reciprocal condition number = 3.954
```

We soon have an error returned when $p = 50$. This is due to the call for the inversion of large covariance matrices, which requires a great amount of computational power (Zang, 2012).

## 3.3 Q9 A Tale of Two Densities: ESD and LSD

The six plots in paper (Liu, Aue, Paul, 2015) can be divided into the following distinguished parameter set-ups. Each set-up will be responsible for 2 different curves representing two sets of observations, MA(1) and i.i.d. Additionally, within each plot, two LSD curves of MA(1) and i.i.d. are required as well.

- plot 1
  - $p = 20, n = 40, \tau = 0$
  - $p = 50, n = 100, \tau = 0$
- plot 2
  - $p = 20, n = 20, \tau = 0$
  - $p = 50, n = 50, \tau = 0$
- plot 3
  - $p = 20, n = 40, \tau = 1$
  - $p = 50, n = 100, \tau = 1$
- plot 4
  - $p = 20, n = 20, \tau = 1$
  - $p = 50, n = 50, \tau = 1$
- plot 5
  - $p = 20, n = 40, \tau = 2$
  - $p = 50, n = 100, \tau = 2$
- plot 6
  - $p = 20, n = 20, \tau = 2$
  - $p = 50, n = 50, \tau = 2$

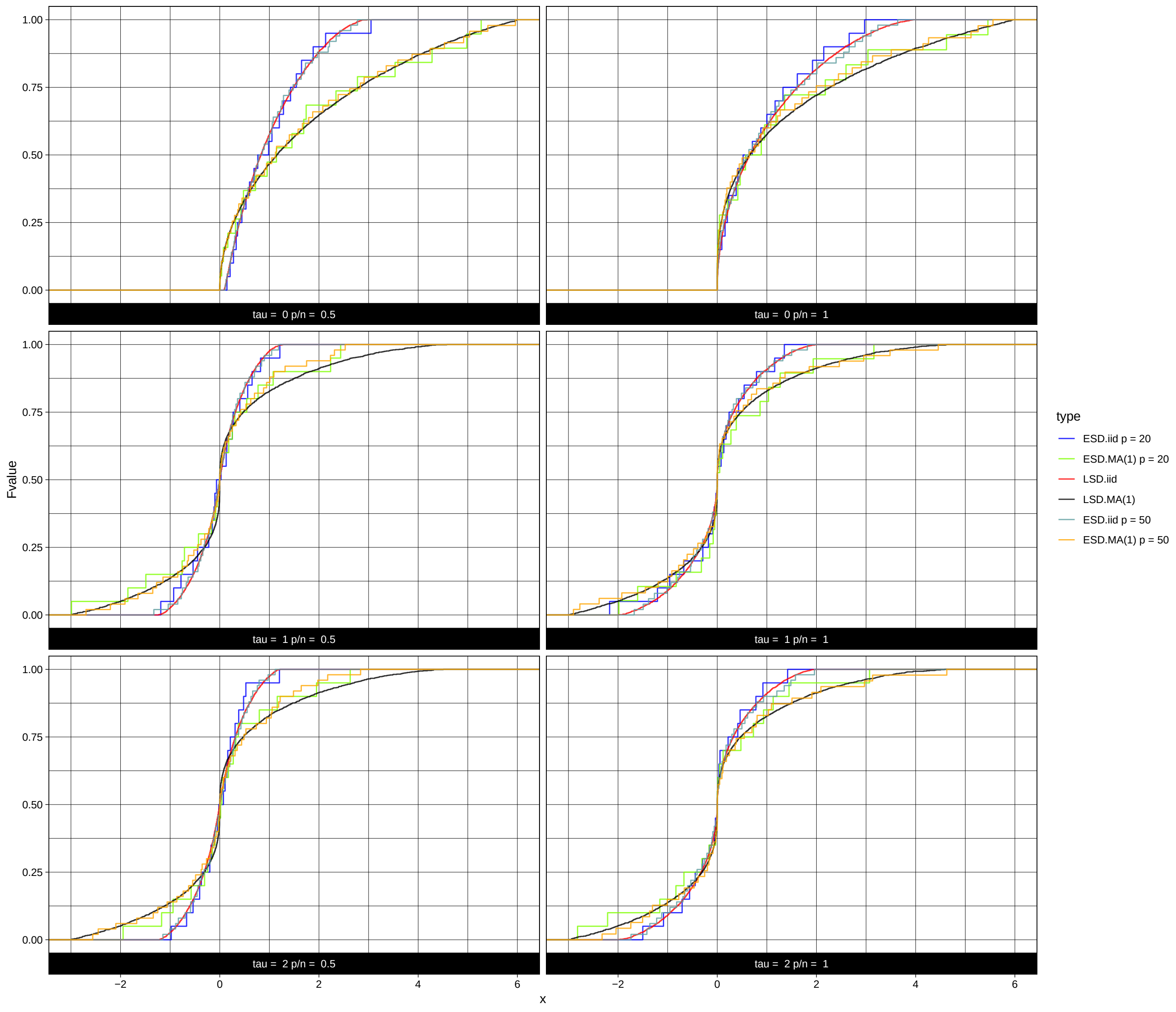
In our coding, two essentail parts are the lag-$\tau$ sample cross-covariance matrix and the empirical spectral density function. Then for each possible combination of parameters, we will calculate the cumulative density function for six curves respectively.

```r
ps <- c(20,50)
ratios <- c(.5, 1)
taus <- c(0,1,2)

lag.sample.acf <- function(n, tau, X) {
    if (tau==0) {
        return(cov(X))
    } else {
        Ctau <- 0
        for (t in 1:(n-tau)) {
            Ctau <- Ctau + X[t,] %*% t(X[t+tau,]) + X[t+tau,] %*% t(X[t,])
        }
        Ctau <- Ctau/(2*n)
        return(Ctau)
    }
}

df <- data.frame()
for (ratio in ratios) {
    for (tau in taus){
        for (p in ps) {
            n <- p / ratio

            # draw the ESDs for iid
            Z <- rmvnorm(n,mean=rep(0,p),sigma=diag(p))
            S <- lag.sample.acf(n, tau, Z)
            ev <- eigen(S)$values

            F_ <- function(x) {
                total <- 0.
                for (i in 1:p) {
                    if (ev[i] <= x){
                        total <- total + 1
                    }
                }
                return(total/p)
            }
            empirical.cdf <- Vectorize(F_)

            df <- rbind(df, data.frame(p=rep(p,n),
                                       ratio=rep(ratio,n),
                                       tau=rep(tau, n),
                                       x=ev,
                                       Fvalue=empirical.cdf(ev),
                                       type=rep(paste("ESD.iid p =",p),n)))

            # draw the ESDs for MA(1)
            A1 <- diag(ncol(Z))
            Z.copy <- Z
            Z.copy <- cbind(rep(1,ncol(Z.copy)), Z.copy[,1:(ncol(Z.copy)-1)])
            X <- Z+Z.copy
            ev2 <- eigen(lag.sample.acf(n, tau, X))$values
            df <- rbind(df, data.frame(p=rep(p,n),
                                       ratio=rep(ratio,n),
                                       tau=rep(tau, n),
                                       x=ev2,
                                       Fvalue=empirical.cdf(ev2),
                                       type=rep(paste("ESD.MA(1) p =",p),n)))

            # draw the LSD for iid
            pp <- 500
            nn <- pp / ratio
            Zn <- rmvnorm(nn,mean=rep(0,pp),sigma=diag(pp))
            Sn <- lag.sample.acf(nn,tau,Zn)
            evn <- eigen(Sn)$values

            df <- rbind(df, data.frame(p=rep(pp,length(evn)),
                                       ratio=rep(ratio,length(evn)),
                                       tau=rep(tau,length(evn)),
                                       x=evn,
                                       Fvalue=empirical.cdf(evn),
                                       type=rep("LSD.iid",length(evn))))

            # draw the LSD for MA(1)
            A1n <- diag(ncol(Zn))
            Zn.copy <- Zn
            Zn.copy <- cbind(rep(1,ncol(Zn.copy)), Zn.copy[,1:(ncol(Zn.copy)-1)])
            Xn <- Zn+Zn.copy
            ev2n <- eigen(lag.sample.acf(length(evn), tau, Xn))$values
            df <- rbind(df, data.frame(p=rep(pp,length(ev2n)),
                                       ratio=rep(ratio,length(ev2n)),
                                       tau=rep(tau, length(ev2n)),
                                       x=ev2n,
                                       Fvalue=empirical.cdf(ev2n),
                                       type=rep("LSD.MA(1)",length(ev2n))))
        }
    }
```

```
}
```

After running the algorithm, a data frame `df` is initiated to store the complete data and be called for visualization.

```
# plotting

df$para <- paste("tau = ", df$tau, "p/n = ", df$ratio)
ggplot(df, aes(x=x, y=Fvalue, color=type)) +
    stat_ecdf(geom = "step", size=.5, alpha=.8) +
    facet_wrap(para ~ ., nrow=3, strip.position = "bottom") +
    scale_x_continuous(limits = c(-3,6)) +
    scale_y_continuous(limits = c(0,1)) +
    scale_color_manual(values=c("LSD.iid" = "red",
                                "LSD.MA(1)" = "black",
                                "ESD.iid p = 20" = "blue",
                                "ESD.MA(1) p = 20" = "chartreuse",
                                "ESD.iid p = 50" = "cadetblue",
                                "ESD.MA(1) p = 50" = "orange")) +
    theme_linedraw()
```



## 4 References

- Sugiura, Nagao (1968). *Unbiasedness of some test criteria for the equality of one or two covariance matrices.* Annals of Mathematical Statistics Vol. 39, No. 5, 1686–1692.
- Anderson (2003). *An introduction to Multivariate Statistical Analysis.* Wiley.
- Johnson, Wichern (2007). *Applied Multivariate Statistical Analysis.* Pearson Prentice Hall.
- Bai, Jiang, Yao, Zheng (2009). *Corrections to LRT on large-dimensional covariance matrix by RMT.* Annals of Statistics Vol 37, No. 6B, 3822–3840.
- Zheng, Bai, Yao (2017). *CLT for eigenvalue statistics of large-dimensional general Fisher matrices with applications.* Bernouilli 23(2), 1130–1178.
- Li, McLeod (1981). *Distribution of the Residual Autocorrelations in Multivariate ARMA Time Series Models*, J.R. Stat. Soc. B 43, No. 2, 231–239.
- Tiao and Box (1981). *Modelling multiple time series with applications.* Journal of the American Statistical Association, 76. 802 – 816.
- Liu, Aue, Paul (2015). *On the Marchenko-Pastur Law for Linear Time Series.* Annals of Statistics Vol. 43, No. 2, 675–712.
- Liu, Aue, Paul (2017). *Spectral analysis of sample autocovariance matrices of a class of linear time series in moderately high dimensions.* Bernouilli 23(4A), 2181–2209.
- *Experiment or simulation to undestand type I and type II errors.* Cross Validated. https://stats.stackexchange.com/a/40874 [Accessed on 2018-10-29]
- Friendly, Siga (2018). *Visualizing Tests for Equality of Covariance Matrices.* https://arxiv.org/pdf/1805.05756.pdf [Accessed on 2018-10-29]
- Mahdi, McLeod (2016). *Improved Multivariate Portmanteau Test.* https://arxiv.org/abs/1611.00442 [Accessed on 2018-10-29]
- Mahdi, McLeod (2018). *Portmanteau Test Statistics.* https://cran.r-project.org/web/packages/portes/vignettes/portesFunctions.pdf [Accessed on 2018-10-29]
- Zang (2012). *Modeling Strategies for Large Dimensional Vector Autoregressions.* https://academiccommons.columbia.edu/doi/10.7916/D8JW8N0V [Accessed on 2018-10-29]