Code Instructions

Setting Up

In order to implement the method described in the manuscript, two source functions are needed.

```
source("cdf2quantile.R")
source("Yule_Walker.R")
```

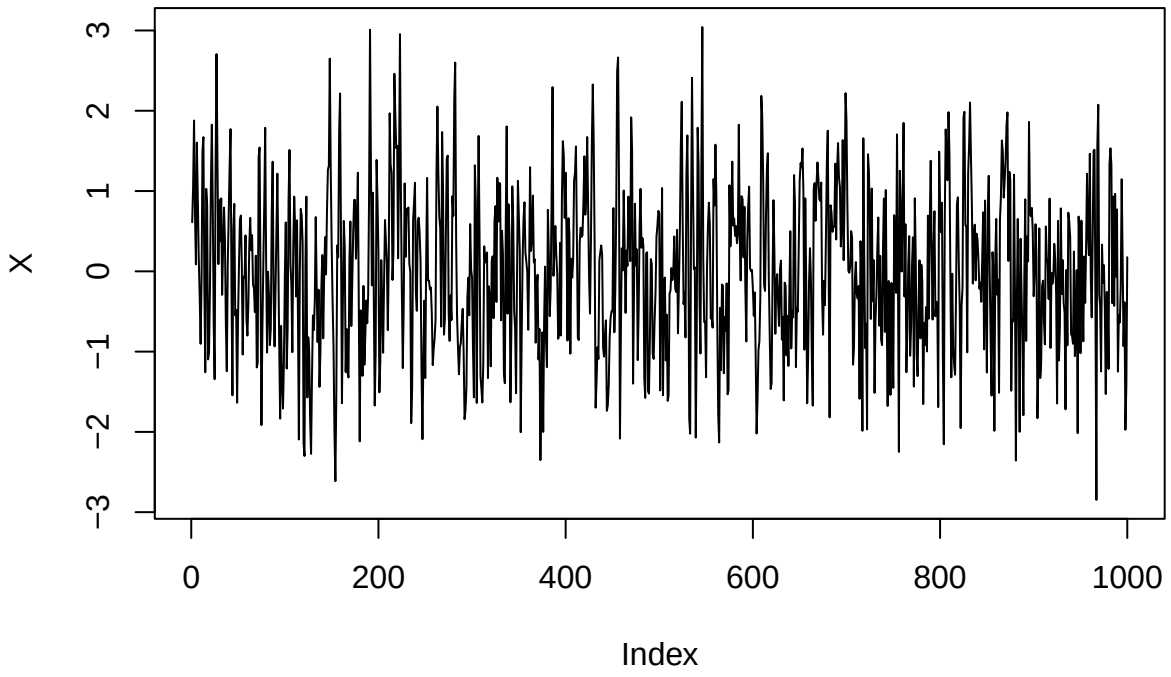
- cdf2quantile: This source function contains the code in order to obtain the relevant quantiles (q_L, q_U) for the confidence intervals based on the pre-defined nominal level α.
- Yule_Walker: This source function contains all the code for the Durbin-Levinson algorithm.

Data Generation

In order to generate the data, we firstly define the sample size T. Then, we define the true value denoted in the code as `truth`$= [2T/3]$. Finally, we define the time-series before (X_bf) and after (X_af) the change point.

```
T <- 1000 # time series length
truth <- floor(2*T/3) # true value

set.seed(1201)

w <- 200

# noise
epsilon <- rnorm(T+2*w,0,1)

# model MA(1) (before the change point)
X_bf <- rep(0,truth+w)
ma_par <- 0.7
for(t in 2:(truth+w)){
  X_bf[1] <- epsilon[1]
  X_bf[t] <- epsilon[t]+ma_par*epsilon[t-1]
  }
X_bf <- (X_bf-mean(X_bf))/sd(X_bf)
X_bf <- X_bf[(w+1):(truth+w)]

# model Nonlinear (after the change point)
X_af <- rep(0,T-truth+w)
nonlinear_par <- 0.5
for(t in 2:(T-truth+w)){
  X_af[1] <- epsilon[truth+1]
```

```
  X_af[t] <- nonlinear_par*abs(X_af[t-1])+epsilon[truth+t]
}
X_af <- (X_af-mean(X_af))/sd(X_af)
X_af <- X_af[(w+1):(T-truth+w)]

X <- c(X_bf,X_af) # entire of time-series
plot(X, type = 'l')
```



Algorithm

In order to run the algorithm described in the manuscript and estimate the change point, we need to consider the following steps.

Step 1. Estimate lags for X_bf and X_af.

```
# lag selection
lag_p1 <- as.vector(VARselect(X_bf)$selection[1])
lag_p2 <- as.vector(VARselect(X_af)$selection[1])
```

```
## [1] "The estimated lags are:"
```

```
## [1] 6 3
```

Step 2. Estimate ϕ's using the Yule-Walker method. Note that the Yule-Walker equations are

$$\hat{\Gamma}_1 X_{bf} = \hat{\gamma}_1 \qquad \text{and} \qquad \hat{\Gamma}_2 X_{af} = \hat{\gamma}_2.$$

Solutions to these two equations are $\hat{\phi}_1$ and $\hat{\phi}_2$. To obtain the Yule-Walker estimators, we use the Durbin-Levinson algorithm described in the source code `"Yule_Walker.R"`.

```r
# Estimating phi's using the Yule-Walker method
gamma.hat <- gamma_hat(X_bf)
gamma.0.hat <- gamma.hat[1]
gamma.n.hat <- gamma.hat[-1]
DL.1step.out <- DL(X_bf-mean(X_bf),gamma.0.hat,gamma.n.hat)
phi1 <- DL.1step.out[1+lag_p1,lag_p1:1]

gamma.hat <- gamma_hat(X_af)
gamma.0.hat <- gamma.hat[1]
gamma.n.hat <- gamma.hat[-1]
DL.1step.out <- DL(X_af-mean(X_af),gamma.0.hat,gamma.n.hat)
phi2 <- DL.1step.out[1+lag_p2,lag_p2:1]
```

```r
phi1
```

```
## [1]  0.68115116 -0.44381101  0.27209842 -0.20388792  0.19698920 -0.05870993
```

```r
phi2
```

```
## [1]  0.26696689 -0.07494063  0.13740962
```

Step 3: Construct the objective function $\mathcal{L}(.)$.

$$\mathcal{L}(\tau) \stackrel{\text{def}}{=} \sum_{t=1}^{\lfloor T\tau \rfloor} (X_t - \hat{\phi}_1 Z_t)^2 + \sum_{t=\lfloor T\tau \rfloor+1}^{T} (X_t - \hat{\phi}_2 Z_t)^2,$$

for any time point $\tau \in [0, 1]$.

```r
# objective function
dist <- 20
Ttau <- (dist):(T-dist)
M <- length(Ttau)
L <- rep(0,M)

X_bf_loop <- foreach(i=1:M) %do% X[1:Ttau[i]]
X_af_loop <- foreach(i=1:M) %do% X[(Ttau[i]+1):T]

lag_p1_loop <- foreach(i=1:M) %do% as.vector(VARselect(X_bf_loop[[i]])$selection[1])
lag_p2_loop <- foreach(i=1:M) %do% as.vector(VARselect(X_af_loop[[i]])$selection[1])

gamma.hat <- foreach(i=1:M) %do% gamma_hat(X_bf_loop[[i]])
gamma.0.hat <- foreach(i=1:M) %do% gamma.hat[[i]][1]
gamma.n.hat <- foreach(i=1:M) %do% gamma.hat[[i]][-1]
DL.1step.out <- foreach(i=1:M) %do%
  DL(X_bf_loop[[i]]-mean(X_bf_loop[[i]]),gamma.0.hat[[i]],gamma.n.hat[[i]])
phi1_loop <- sapply(1:M,function(i){DL.1step.out[[i]][1+lag_p1_loop[[i]],lag_p1_loop[[i]]:1]})

gamma.hat <- foreach(i=1:M) %do% gamma_hat(X_af_loop[[i]])
gamma.0.hat <- foreach(i=1:M) %do% gamma.hat[[i]][1]
gamma.n.hat <- foreach(i=1:M) %do% gamma.hat[[i]][-1]
DL.1step.out <- foreach(i=1:M) %do% DL(X_af_loop[[i]]-mean(X_af_loop[[i]]),
```

```
              gamma.0.hat[[i]],gamma.n.hat[[i]])
phi2_loop <- sapply(1:M,function(i)
  {DL.1step.out[[i]][1+lag_p2_loop[[i]],lag_p2_loop[[i]]:1]})

for(i in 1:M){
  Z_bf_loop <- matrix(0,nrow=Ttau[i],ncol=lag_p1_loop[[i]])
  for(t in 1:Ttau[i]){
    for(j in 1:lag_p1_loop[[i]]){
    Z_bf_loop[t,j] <- ifelse(j<t,X_bf_loop[[i]][t-j],0)
    }
    }
  Z_af_loop <- matrix(0,nrow=T-Ttau[i],ncol=lag_p2_loop[[i]])
  for(t in 1:T-Ttau[i]){
    for(j in 1:lag_p2_loop[[i]]){
    Z_af_loop[t,j] <- ifelse(j<t,X_af_loop[[i]][t-j],0)
    }
    }
L[i] <- sum((X_bf_loop[[i]]-Z_bf_loop%*%matrix(phi1_loop[[i]],ncol=1))^2)+
  sum((X_af_loop[[i]]-Z_af_loop%*%matrix(phi2_loop[[i]],ncol=1))^2)
}
L[1:10]
```

```
##  [1] 793.0233 794.8503 793.5218 792.4912 794.0686 790.7762 795.0625 790.6163
##  [9] 794.3233 795.8475
```

Step 4: Estimate near optimal change point $[T\hat{\tau}]$ from the objective function.

```
# near optimal estimator
Min_data <- data.frame(cbind(Ttau,L))
Ttau_hat <- Min_data$Ttau[which(Min_data$L==min(Min_data$L))]
Ttau_hat
```

```
## [1] 724
```

Step 5: Refit AR(p) coefficients $\hat{\phi}_1$ and $\hat{\phi}_2$.

```
# refitting process
X_bf_new <- X[1:Ttau_hat]
X_af_new <- X[(Ttau_hat+1):T]

lag_p1_new <- as.vector(VARselect(X_bf_new)$selection[1])
lag_p2_new <- as.vector(VARselect(X_af_new)$selection[1])
lag_new_max <- max(lag_p1_new,lag_p2_new)

gamma.hat <- gamma_hat(X_bf_new)
gamma.0.hat <- gamma.hat[1]
gamma.n.hat <- gamma.hat[-1]
DL.1step.out <- DL(X_bf_new-mean(X_bf_new),gamma.0.hat,gamma.n.hat)
phi1_new <- DL.1step.out[1+lag_p1_new,lag_p1_new:1]

gamma.hat <- gamma_hat(X_af_new)
gamma.0.hat <- gamma.hat[1]
```

```
gamma.n.hat <- gamma.hat[-1]
DL.1step.out <- DL(X_af_new-mean(X_af_new),gamma.0.hat,gamma.n.hat)
phi2_new <- DL.1step.out[1+lag_p2_new,lag_p2_new:1]

if(lag_p1_new < lag_p2_new){
  phi1_new <- c(phi1_new,rep(0,lag_p2_new-lag_p1_new))
  }

if(lag_p1_new > lag_p2_new){
  phi2_new <- c(phi2_new,rep(0,lag_p1_new-lag_p2_new))
  }
```

Step 6: Define the new objective function $Q(\tau;\phi_1,\phi_2)$ as

$$Q(\tau;\phi_1,\phi_2)=\frac{1}{T-p+1}\left(\sum_{t=p}^{\lfloor T\tau\rfloor}(X_t-\phi_1'Z_t)^2+\sum_{t=\lfloor T\tau\rfloor+1}^{T}(X_t-\phi_2'Z_t)^2\right).$$

```
dist <- 20
Ttau2 <- (dist):(T-dist)
M2 <- length(Ttau2)
Q <- rep(0,M2)
for(i in 1:M2){
  X_bf_loop2 <- X[1:Ttau2[i]]
  X_af_loop2 <- X[(Ttau2[i]+1):T]

  Z_bf_loop2 <- matrix(0,nrow=Ttau2[i],ncol=lag_new_max)
  for(t in 1:Ttau2[i]){
    for(j in 1:lag_new_max){
      Z_bf_loop2[t,j] <- ifelse(j<t,X_bf_loop2[t-j],0)
    }
    }
  Z_af_loop2 <- matrix(0,nrow=T-Ttau2[i],ncol=lag_new_max)
  for(t in 1:T-Ttau2[i]){
    for(j in 1:lag_new_max){
      Z_af_loop2[t,j] <- ifelse(j<t,X_af_loop2[t-j],0)
    }
    }
  Q[i] <- sum((X_bf_loop2-Z_bf_loop2%*%matrix(phi1_new,ncol=1))^2)/(T-2*lag_new_max+1)+
    sum((X_af_loop2-Z_af_loop2%*%matrix(phi2_new,ncol=1))^2)/(T-2*lag_new_max+1)
}
Q[1:10]
```

```
##  [1] 0.8957298 0.8961555 0.8938648 0.8936198 0.8936925 0.8921768 0.8953838
##  [8] 0.8919007 0.8906099 0.8906992
```

Step 7: Update the estimator for the change point parameter as:

$$[T\tilde{\tau}]=\text{argmin}_{\tau\in(0,1)}Q(\tau;\hat{\phi}_1,\hat{\phi}_2).$$

```r
# optimal estimator
Min_data2 <- data.frame(cbind(Ttau2,Q))
Ttau_tilde <- Min_data2$Ttau2[which(Min_data2$Q==min(Min_data2$Q))]

Ttau_tilde
```

```
## [1] 712
```

Constructing Confidence Interval

In order to construct the $100(1-\alpha)\%$ confidence interval, we need to follow the steps described below. Note that the $100(1-\alpha)\%$ confidence interval for the change point can be defined as

$$([T\tilde{\tau}] - [q_U/\hat{L}] - 1, [T\tilde{\tau}] - [q_L/\hat{L}] + 1),$$

where (q_L, q_U) are the $(1-\alpha/2)$-th lower and upper quantiles of the random variable $\text{argmax}\{W(s) - |s|/2\}$ assuming $W(s)$ is a standard Wiener process defined on $[0, \infty)$. Additionally, note that $\hat{L}$ is defined in the following code, and $[a]$ is the integer part of "a".

The steps are as follows:

1. Update all the elements of time series based on the estimated change point; i.e. $[T\tilde{\tau}]$.
2. Construct covariance matrices.
3. Estimate the jump sizes for before and after the estimated change point.
4. Estimate variances σ's from limits.
5. Calculate $\hat{L}(.)$.
6. Estimate the related parameters following Bai (1997)'s paper.
7. Construct the relevant quantiles for the confidence interval.

```r
# updating all the elements based on Ttau_tilde
X_bf_new2 <- X[1:Ttau_tilde]
X_af_new2 <- X[(Ttau_tilde+1):T]

lag_p1_new2 <- as.vector(VARselect(X_bf_new2)$selection[1])
lag_p2_new2 <- as.vector(VARselect(X_af_new2)$selection[1])
lag_new2_max <- max(lag_p1_new2,lag_p2_new2)

gamma.hat <- gamma_hat(X_bf_new2)
gamma.0.hat <- gamma.hat[1]
gamma.n.hat <- gamma.hat[-1]
DL.1step.out <- DL(X_bf_new2-mean(X_bf_new2),gamma.0.hat,gamma.n.hat)
phi1_new2 <- DL.1step.out[1+lag_p1_new2,lag_p1_new2:1]

gamma.hat <- gamma_hat(X_af_new2)
gamma.0.hat <- gamma.hat[1]
gamma.n.hat <- gamma.hat[-1]
DL.1step.out <- DL(X_af_new2-mean(X_af_new2),gamma.0.hat,gamma.n.hat)
phi2_new2 <- DL.1step.out[1+lag_p2_new2,lag_p2_new2:1]
```

```
if(lag_p1_new2 < lag_p2_new2){
  phi1_new2 <- c(phi1_new2,rep(0,lag_p2_new2-lag_p1_new2))
  }

if(lag_p1_new2 > lag_p2_new2){
  phi2_new2 <- c(phi2_new2,rep(0,lag_p1_new2-lag_p2_new2))
  }

Z_bf_new2 <- matrix(0,nrow=Ttau_tilde,ncol=lag_new2_max)
for(t in 1:Ttau_tilde){
  for(j in 1:lag_new2_max){
    Z_bf_new2[t,j] <- ifelse(j<t,X_bf_new2[t-j],0)
  }
  }
Z_af_new2 <- matrix(0,nrow=T-Ttau_tilde,ncol=lag_new2_max)
for(t in 1:T-Ttau_tilde){
  for(j in 1:lag_new2_max){
    Z_af_new2[t,j] <- ifelse(j<t,X_af_new2[t-j],0)
  }
}

# covariance matrices
SIGMA1 <- cov(Z_bf_new2)
SIGMA2 <- cov(Z_af_new2)

# jump size (xi2)
eta_star <- phi1_new2 - phi2_new2
xi2_bf <- sqrt(sum(eta_star^2))
xi2_af <- sqrt(sum(eta_star^2))

# estimation of sigma's from limits
sigma1_2 <- as.vector(xi2_bf^(-2)*(matrix(eta_star,nrow=1)%*%SIGMA1%*%
                                   t(matrix(eta_star,nrow=1))))
sigma2_2 <- as.vector(xi2_af^(-2)*(matrix(eta_star,nrow=1)%*%SIGMA2%*%
                                   t(matrix(eta_star,nrow=1))))

e_bf <- t(X_bf_new2 - Z_bf_new2%*%phi1_new2)
e_af <- t(X_af_new2 - Z_af_new2%*%phi2_new2)

dd_1 <- as.vector((t(Z_bf_new2%*%eta_star)*e_bf))
dd_1 <- dd_1[(lag_new2_max+1):length(dd_1)]

dd_2 <- as.vector((t(Z_af_new2%*%eta_star)*e_af))
dd_2 <- dd_2[(lag_new2_max+1):length(dd_2)]

sigma1_star_2 <- xi2_bf^(-2)*(var(dd_1))
sigma2_star_2 <- xi2_af^(-2)*(var(dd_2))

# confidence interval
L_hat <- as.vector((sigma1_2^2)*(1/sigma1_star_2)*(xi2_bf^2))

# following parameters come from Bai's paper
phi_G <- 1*sigma2_star_2/sigma1_star_2
```

```
xi_G <- as.vector(sigma2_2/sigma1_2)

a_n <- 0.5*(xi_G/phi_G)*(1+(xi_G/phi_G))
b_n <- 0.5+(xi_G/phi_G)
c_n <- (phi_G*(phi_G+2*xi_G))/(xi_G*(phi_G+xi_G))
d_n <- (phi_G+2*xi_G)^2/((phi_G+xi_G)*xi_G)

a_p <- (phi_G+xi_G)/2
b_p <- (2*phi_G+xi_G)/(2*sqrt(phi_G))
c_p <- (xi_G*(2*phi_G+xi_G))/((phi_G+xi_G)*phi_G)
d_p <- (2*phi_G+xi_G)^2/((phi_G+xi_G)*phi_G)

Q_L70 <- quantile_low(a_n,b_n,c_n,d_n,xi_G,phi_G,0.15)
Q_U70 <- quantile_up(a_p,b_p,c_p,d_p,xi_G,phi_G,0.85)

L70 <- Ttau_tilde-floor(Q_U70/L_hat)-1 # lower bound of 70% CI
U70 <- Ttau_tilde-floor(Q_L70/L_hat)+1 # upper bound of 70% CI

Q_L80 <- quantile_low(a_n,b_n,c_n,d_n,xi_G,phi_G,0.1)
Q_U80 <- quantile_up(a_p,b_p,c_p,d_p,xi_G,phi_G,0.9)

L80 <- Ttau_tilde-floor(Q_U80/L_hat)-1 # lower bound of 80% CI
U80 <- Ttau_tilde-floor(Q_L80/L_hat)+1 # upper bound of 80% CI

Q_L90 <- quantile_low(a_n,b_n,c_n,d_n,xi_G,phi_G,0.05)
Q_U90 <- quantile_up(a_p,b_p,c_p,d_p,xi_G,phi_G,0.95)

L90 <- Ttau_tilde-floor(Q_U90/L_hat)-1 # lower bound of 90% CI
U90 <- Ttau_tilde-floor(Q_L90/L_hat)+1 # upper bound of 90% CI

Q_L95 <- quantile_low(a_n,b_n,c_n,d_n,xi_G,phi_G,0.025)
Q_U95 <- quantile_up(a_p,b_p,c_p,d_p,xi_G,phi_G,0.975)

L95 <- Ttau_tilde-floor(Q_U95/L_hat)-1 # lower bound of 95% CI
U95 <- Ttau_tilde-floor(Q_L95/L_hat)+1 # upper bound of 95% CI

Q_L99 <- quantile_low(a_n,b_n,c_n,d_n,xi_G,phi_G,0.005)
Q_U99 <- quantile_up(a_p,b_p,c_p,d_p,xi_G,phi_G,0.995)

L99 <- Ttau_tilde-floor(Q_U99/L_hat)-1 # lower bound of 99% CI
U99 <- Ttau_tilde-floor(Q_L99/L_hat)+1 # upper bound of 99% CI
```

```
## [1] "The 95% confidence interval is"
```

```
## [1] 574 753
```

References

- Bai, J. (1997), Estimation of a change point in multiple regression models. *Review of Economics and Statistics* 79, 551-563.