Econometrics III - Assignment 2

The Explosive Invertibles

08 Apr 2022

Question 1

1. Analyze the claims regarding the spurious regression problem using a Monte Carlo simulation study.

The code below first simulates both independent random walks

$$Y_t = Y_{t-1} + u_t, \quad u_t \sim \mathcal{N}(0, \sigma_u^2) X_t = X_{t-1} + v_t, \quad v_t \sim \mathcal{N}(0, \sigma_v^2)$$

for $t \in \{1, 2, \ldots, T\}$ with $T \in \{50, 100, 200\}$ separately $n = 10.000$ times and subsequently runs the regression

$$Y_t = \alpha + \beta X_t + \varepsilon_t$$

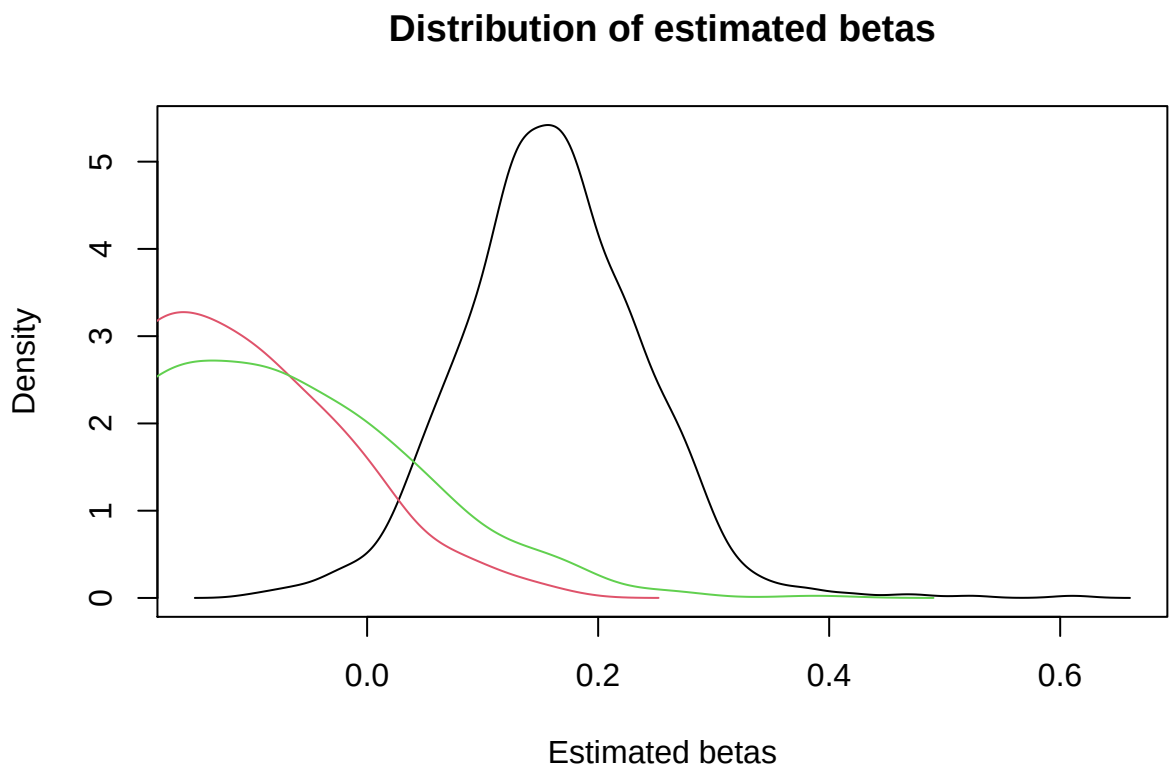
for each simulation round, i.e. $n = 10.000$ times, and stores the regression outputs $\hat{\beta}$, $\hat{\beta}/SE(\hat{\beta})$ and R^2 for each simulation round respectively for each $T \in \{50, 100, 200\}$.

```
# Simulation with x being non-stationary
set.seed(20)
l = c(50, 100, 200)
n = 1000

sigma_u = 1
sigma_v = 1

# Create the list to store dataframes
simulationsX <- list()
simulationsY <- list()


# Generate Random walk
for (j in 1:length(l)){
  t = l[j]
  x <- data.frame(rnorm(n,0,sigma_v))
  y <- data.frame(rnorm(n,0,sigma_u))
  for (i in 1:(t-1)){
    v <- as.numeric(x[i,]) + rnorm(n,0,sigma_v)
    x <- data.frame(x, v)

    u <- as.numeric(y[i,]) + rnorm(n,0,sigma_u)
    y <- data.frame(y, u)
  }
  simulationsX[j] <- list(data.frame(x))
  simulationsY[j] <- list(data.frame(y))
}

# Run regressions & store beta, SE and R^2
```

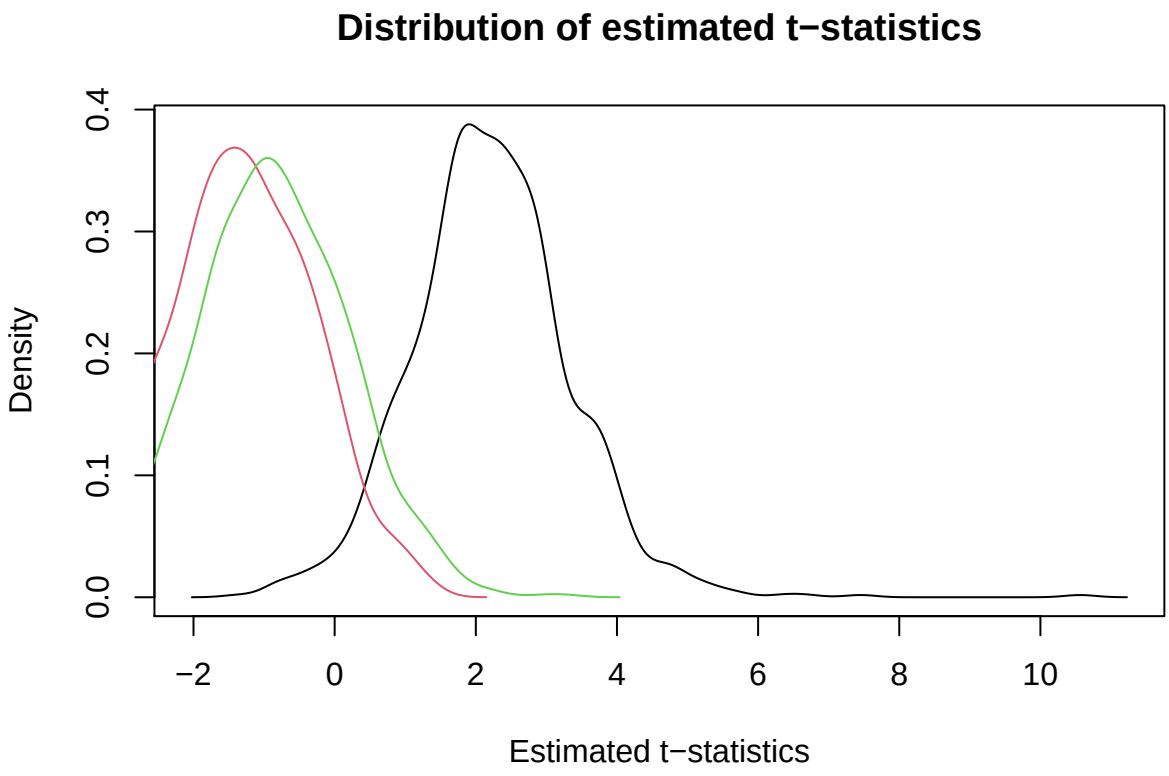
```
list_est <- list()

for (j in 1:length(l)){
  est <- data.frame(matrix(ncol = 3, nrow = n))
  for (i in 1:n){
    reg <- summary(lm(as.numeric(simulationsY[[j]][i,])~ as.numeric(simulationsX[[j]][i,])))
    est[i,1] <- reg$coef[2,1]
    est[i,2] <- reg$coef[2,1] / reg$coef[2,2]
    est[i,3] <- reg$r.squared
  }
  list_est[j] <- list(data.frame(est))
}
```

Below we plot the densities of the simulated estimated betas, t-statistics and R^2 for $T \in \{50, 100, 200\}$.

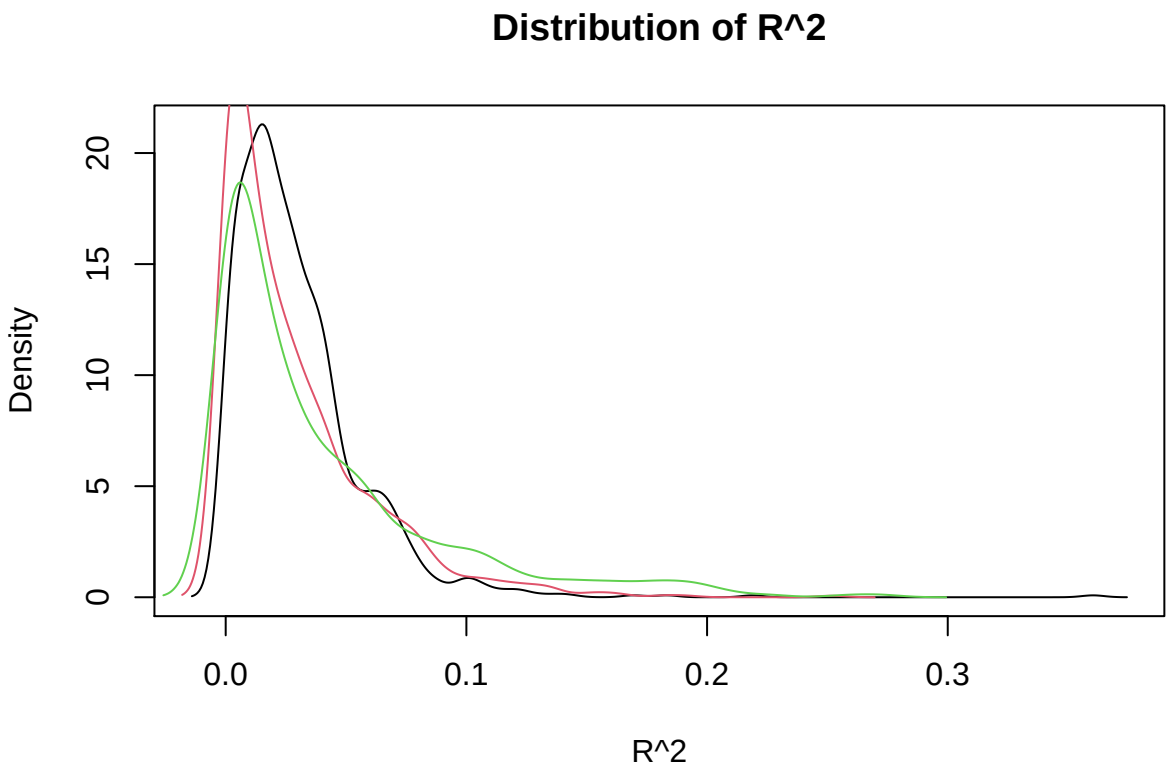
```
# Plot densities
plot(density(list_est[[3]][,1])$x,density(list_est[[3]][,1])$y,type="l",xlab="Estimated betas",ylab="Den
points(density(list_est[[2]][,1])$x, density(list_est[[2]][,1])$y, type="l",col=2)
points(density(list_est[[1]][,1])$x, density(list_est[[1]][,1])$y, type="l",col=3)
```




```
plot(density(list_est[[3]][,2])$x,density(list_est[[3]][,2])$y,type="l",xlab="Estimated t-statistics",yl
points(density(list_est[[2]][,2])$x, density(list_est[[2]][,2])$y, type="l",col=2)
points(density(list_est[[1]][,2])$x, density(list_est[[1]][,2])$y, type="l",col=3)
```




```
plot(density(list_est[[3]][,3])$x,density(list_est[[3]][,3])$y,type="l",xlab="R^2",ylab="Density", main=
points(density(list_est[[2]][,3])$x, density(list_est[[2]][,3])$y, type="l",col=2)
points(density(list_est[[1]][,3])$x, density(list_est[[1]][,3])$y, type="l",col=3)
```



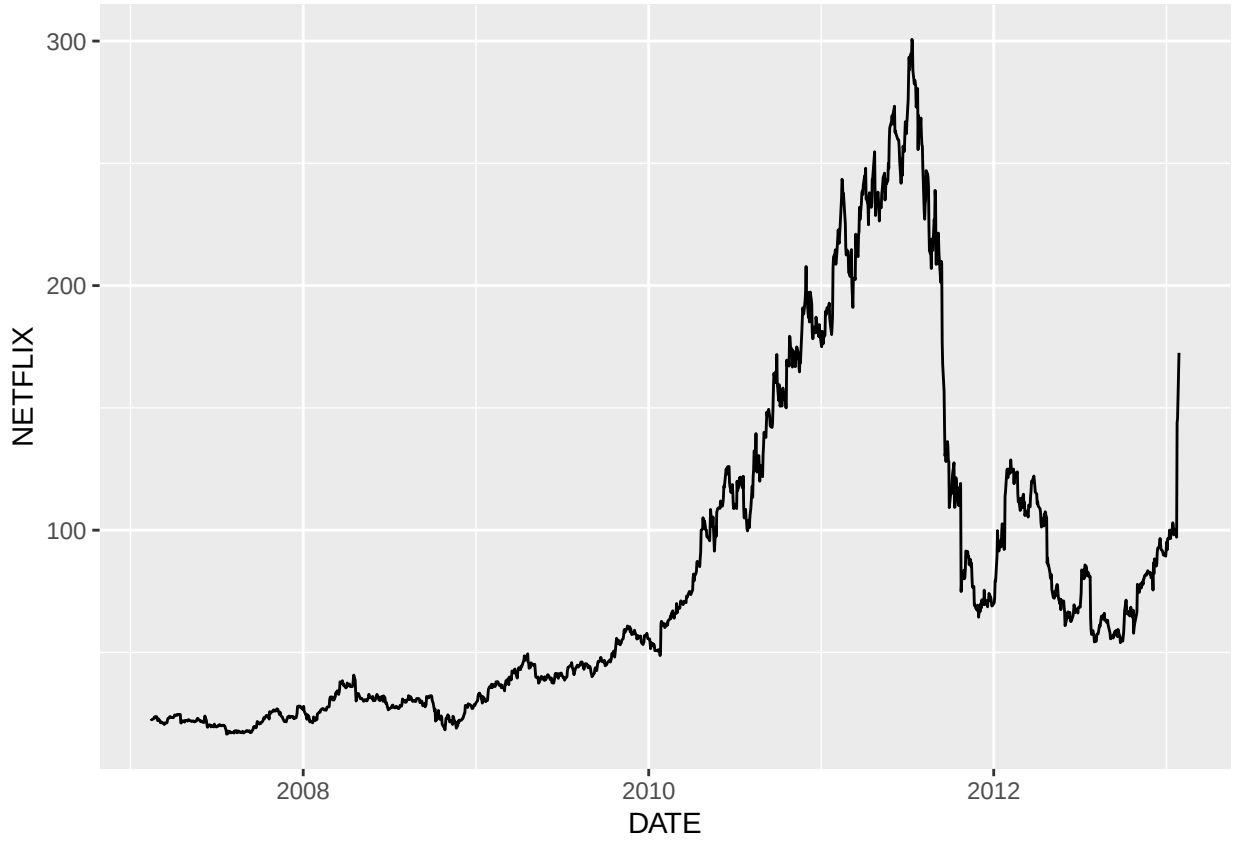
The plotted densities show symptoms of spurious regression since estimated betas converge to a random variable, i.e. estimated betas differ for each $T \in \{50, 100, 200\}$, and densities of obtained R^2's for different $T \in \{50, 100, 200\}$ also do not reveal any sort of convergence. Moreover, estimated t-statistis also differ substantially in their mean scale across each $T \in \{50, 100, 200\}$. These are clear indications of spurious regression which emerges from regressing two independet time series of integration order 1.

2. Provide plots for two stock market time-series at your choice and report 12-period ACF and PACF functions for those two time series. What does the sample ACF tell you about the dynamic properties of these stocks?

```
# Load data
data <- read.csv("data_assign_p3.csv")
data$DATE<- as.Date(data$DATE, format = "%d/%m/%Y")
```

Below we plot the stock price time series for Netflix and Apple.

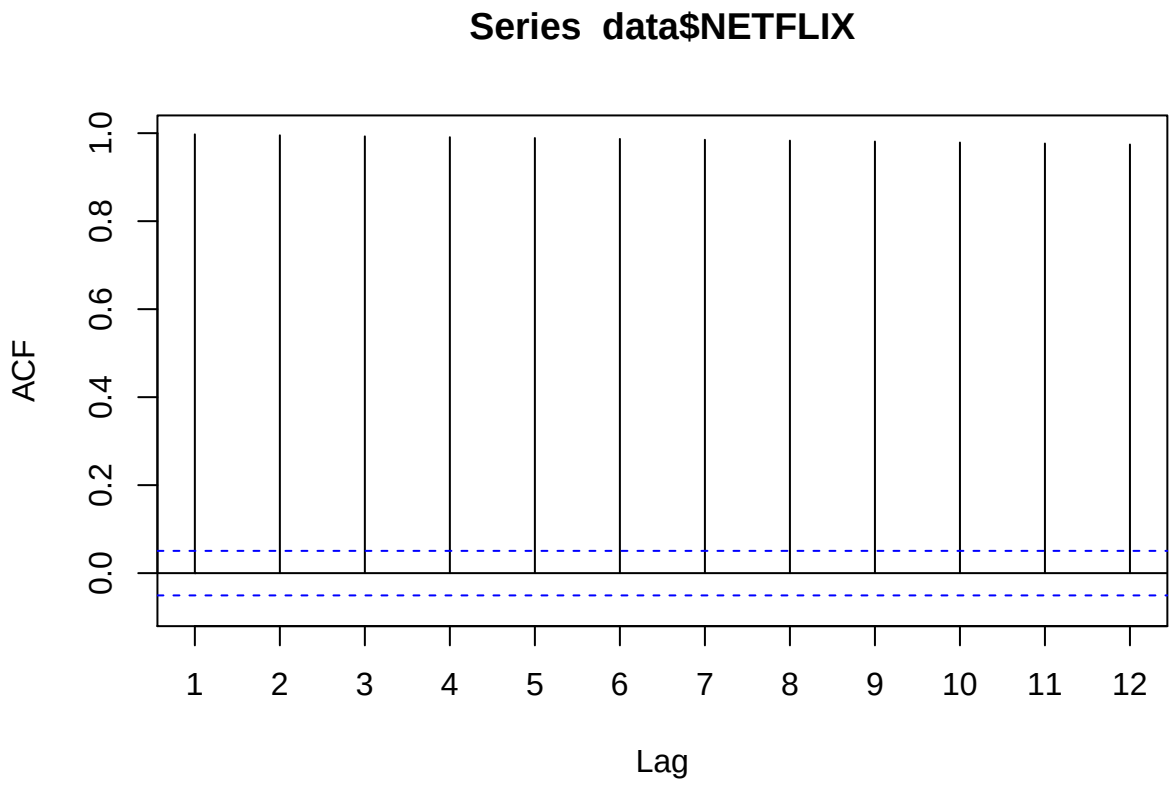
```
ggplot(data, aes(x = DATE, y = NETFLIX)) +
  geom_line()
```



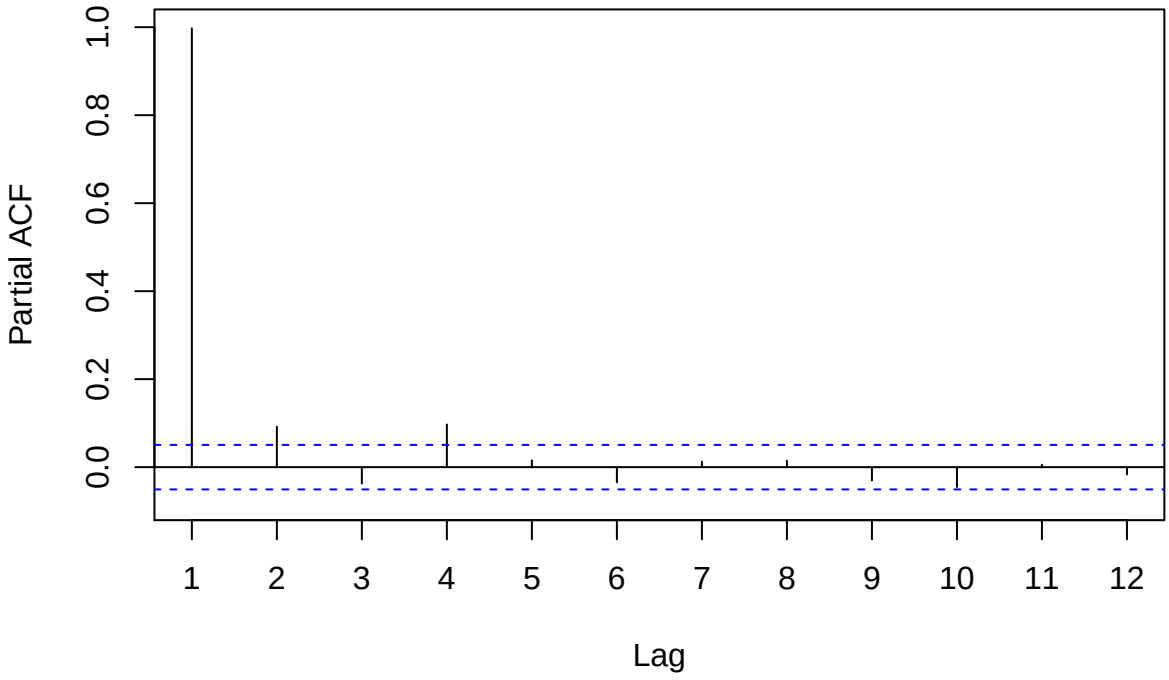

```
ggplot(data, aes(x = DATE, y = APPLE)) +
  geom_line()
```

The plots already raise some suspicion that the time-series may be non-stationary. The issue is investigated further by inspecting the ACF and PACF functions of both stocks which yield further insights about the dynamic properties of the stocks. Note that the functions are plotted alongside 95%-confidence intervals.

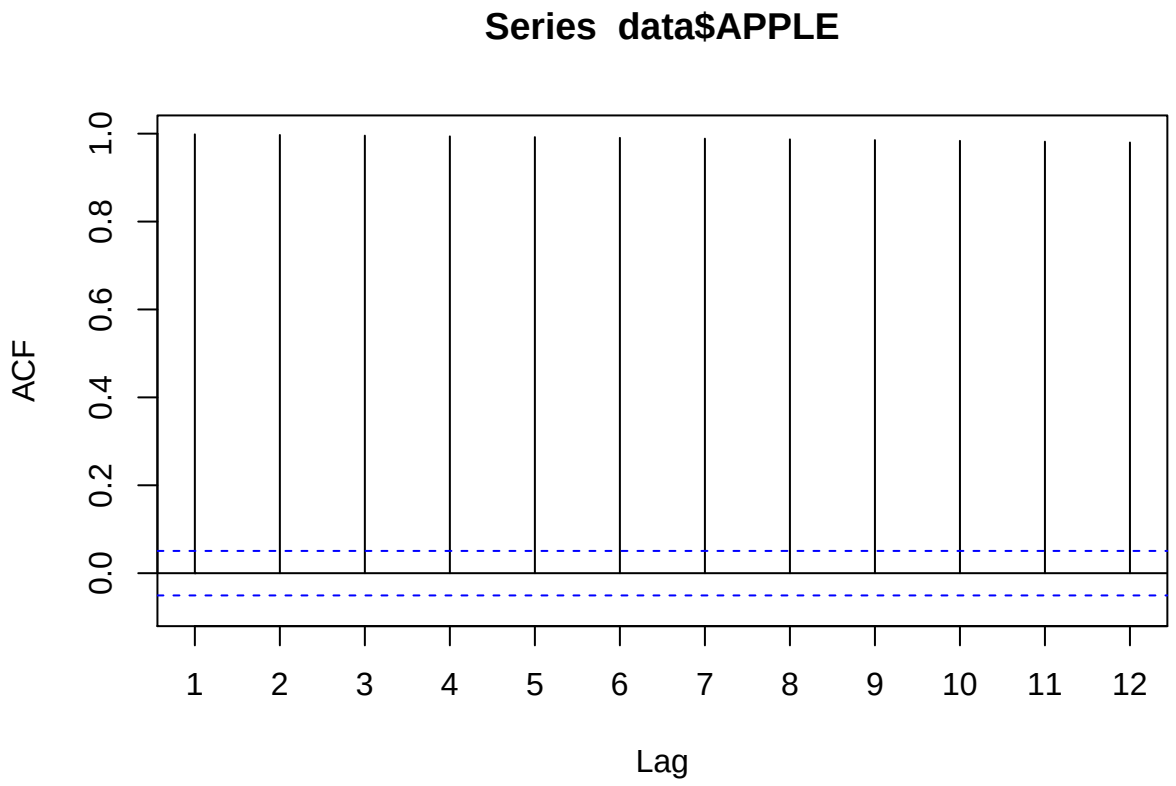
```
Acf(
  data$NETFLIX,
  lag.max = 12
)
```



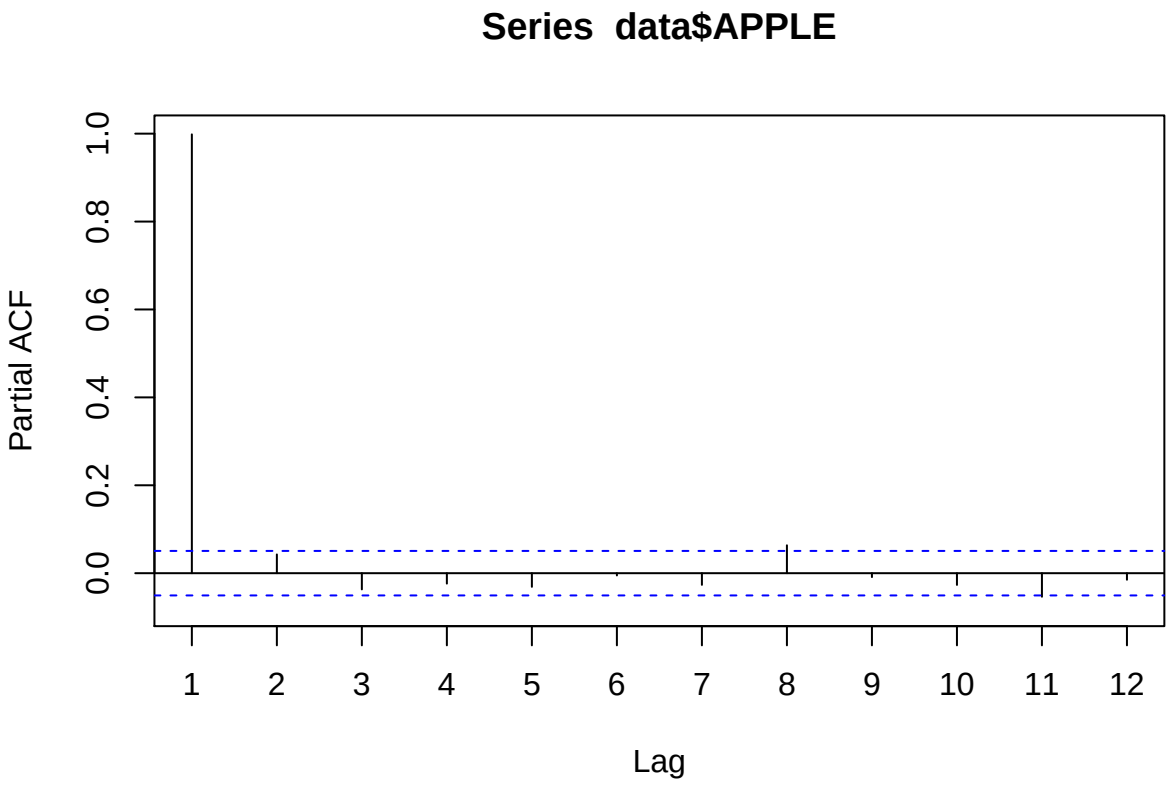

```
Pacf(
  data$NETFLIX,
  lag.max = 12
)
```




```
Acf(
  data$APPLE,
  lag.max = 12
)
```




```
Pacf(
  data$APPLE,
  lag.max = 12
)
```



The insights that can be drawn based on the auto-correlation functions are similar for both stocks. The correlation of the respective stock price with any of the previous trading days' stock prices (here we consider the previous 12 trading days) is very close to 1. This points at the fact that both time series seem to possess significant memory serving as further worry towards alleged non-stationarity. Partial auto-correlation functions clearly show that this observation is mainly driven by very high correlation with the preceding trading day's price for both stocks. Partial correlation with other previous prices differ for both stock considered here. For Netflix, also other closely preceding end-of-day stock prices are significantly positively correlated with the current price, while this is not the case for Apple. Given the big amount of lags (and overall empirical findings about the time series of stock prices), these findings should be treated with caution and are therfore not further discussed here.

3. Perform an ADF unit-root test for all the 10 time series using the general-to-specific approach based on the Schwarz Information Criterion (SIC). Report the values of the ADF test statistics. Is the unit-root hypothesis rejected for any time-series at the 90% confidence level? Did you expect to reject the unit-root hypothesis for some time-series at this confidence level? Justify your answer carefully.

To perform the model selection of the AR(p)-model considered in the ADF unit-root-test for each stock in an efficient way, we use a program that estimates the model for each combination of auto-lag order (with and without drift) and selects the one with the lowest Schwarz Information Criterion (SIC). To do so we first create a matrix which includes all possible combinations for some maximum lag-order that can subsequently be used to estimate every model. Note that we do not consider the model only with drift as well as do not consider a deterministic time trend since we handling finance data.

```
# Set combinatorics where 1 last lag is intercept
max_lag = 6
lag <- c(1:max_lag)
comb_set <- CombSet(lag, m=1:max_lag)
comb = 0
```

```
for (i in 1:max_lag){
  comb = comb + CombN(max_lag,i)
}
lag_struc <- matrix(0,nrow = comb, ncol =max_lag)
lag_max <- rep(0, comb)

# Lag 1
lag_n = 1
for (i in 1:length(comb_set[[lag_n]])){
  lag = comb_set[[lag_n]][i]
  lag_struc[i,lag] = NA
}

# Lag 2 and higher
start = 0
for (n in 2:max_lag){
  lag_n = n
  start = start + length(comb_set[[lag_n-1]][,1])
  for (i in 1:length(comb_set[[lag_n]][,1])){
    for (j in 1:lag_n){
      lag = comb_set[[lag_n]][i,j]
      lag_struc[start + i,lag] = NA
    }
  }
}

# Delete model with only intercept
lag_struc <- lag_struc[-max_lag,]
comb = comb -1
```

The code below estimates the AR(p)-model for each combination and selects the combination with lowest SIC. To calculate the ADF test statistic the selected AR(p)-model needs to be rewritten in the following way for the case that it does not contain holes in its lag structure:

$$X_t = \alpha+\phi_1 X_{t-1}+\cdots+\phi_p X_{t-p}+\epsilon_t \Leftrightarrow \Delta X_t = \alpha+\beta X_{t-1}+\phi_2^*\Delta X_{t-1}+\cdots+\phi_{t-p+1}^*\Delta X_{t-p+1}+\epsilon_t \text{where} \quad \beta = (\phi_1+\cdots+\phi_p-1) \quad \text{ar}$$

so that the ADF test statistic can be calculated as

$$ADF = \frac{\hat{\beta}}{SE(\hat{\beta})} \sim \text{Dickey-Fuller Distribution with(out) drift}$$

However, such a generalized formulation does not hold if the lag-structure contains arbitrary holes for which the program needs to allow. Shortly before submitting this assignment we became aware of this complication and did not find a way to program the re-arrangement in a systemic way. Therefore, we decided to stick to the detected programming mistake which assumes that the AR(p) model for arbitrary lags $\{p_1,\ldots,p_n\}$ with $p_1,\ldots,p_n \in \mathbb{Z}$, i.e. allowing for holes, can be rewritten as follows (which is not the case):

$$X_t = \alpha+\phi_1 X_{t-p_1}+\cdots+\phi_p X_{t-p_n}+\epsilon_t \Leftrightarrow \Delta X_t = \alpha+\beta X_{t-1}+\phi_1^*\Delta X_{t-p_1}+\cdots+\phi_n^*\Delta X_{t-p_n}+\epsilon_t \text{such that} \quad \beta = (\phi_1+\cdots+\phi_n-1)$$

We are fully aware that this is wrong, but did not find the mistake soon enough to do the rearring by hand for each model separately, but sticked to the wrongly programmed solution provided below.

Then the program checks whether the thus obtained ADF test statistic is lower than the 90%-significance level critical value from the Dickey-Fuller Distribution with or without drift (dependent on the selected model)

and returns TRUE if this is the case. We do not further comment the code or its output for each stock but point out that the AR-model estimations are performed using the CSS-estimation method (conditional sum-of-squares) which proved to be more flexible towards the model specication to be estimated.

```
# Estimate all models and pick specification with lowest SIC - APPLE
ts <- data$APPLE
sics <- rep(0, comb)

for (i in 1:comb){
  logl <- arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[i,], method ="CSS")$loglik
  k = sum(is.na(lag_struc[i,]))
  sics[i] = k*log(length(ts)) - 2*logl
}
min(sics)
```

```
## [1] 9951.287
```

```
arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[which.min(sics),], method ="CSS")
```

```
##
## Call:
## arima(x = ts, order = c(max_lag - 1, 0, 0), fixed = lag_struc[which.min(sics),
##     ], method = "CSS")
##
## Coefficients:
##          ar1     ar2  ar3  ar4  ar5  intercept
##       0.8840  0.1165    0    0    0          0
## s.e.  0.0258  0.0258    0    0    0          0
##
## sigma^2 estimated as 44.3:  part log likelihood = -4968.33
```

```
# Perform ADF Test
ts <- ts(ts)
struc <- c(0)
for (i in 1:length(lag_struc[which.min(sics),]-1)){
  if (is.na(lag_struc[which.min(sics),i])){
    struc <- cbind(struc, i)
  }
}
struc <- as.numeric(struc[,-1])
struc <- head(struc,-1)
if(is.na(lag_struc[which.min(sics),i])){
  int = TRUE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc))
} else{
  int = FALSE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc)-1)
}
beta = summary(mod)$coefficients[1,1]
se = summary(mod)$coefficients[1,2]
adf = beta/se
adf
```

```
## [1] 0.8922442
```

```
if(int==TRUE){
  adf< (-1.616)
```

```
}else{
  adf< (-2.568)
}
```

```
## [1] FALSE
```

```
# Estimate all models and pick specification with lowest SIC - EXXON_MOBIL
ts <- data$EXXON_MOBIL
sics <- rep(0, comb)

for (i in 1:comb){
  logl <- arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[i,], method ="CSS")$loglik
  k = sum(is.na(lag_struc[i,]))
  sics[i] = k*log(length(ts)) - 2*logl
}
min(sics)
```

```
## [1] 4961.287
```

```
arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[which.min(sics),], method ="CSS")
```

```
##
## Call:
## arima(x = ts, order = c(max_lag - 1, 0, 0), fixed = lag_struc[which.min(sics),
##     ], method = "CSS")
##
## Coefficients:
##          ar1  ar2     ar3  ar4  ar5  intercept
##       0.8964    0  0.1036    0    0          0
## s.e.  0.0190    0  0.0190    0    0          0
##
## sigma^2 estimated as 1.587:  part log likelihood = -2473.33
```

```
# Perform ADF Test
ts <- ts(ts)
struc <- c(0)
for (i in 1:length(lag_struc[which.min(sics),]-1)){
  if (is.na(lag_struc[which.min(sics),i])){
    struc <- cbind(struc, i)
  }
}
struc <- as.numeric(struc[,-1])
if(is.na(lag_struc[which.min(sics),i])){
  int = TRUE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc))
} else{
  int = FALSE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc)-1)
}
beta = summary(mod)$coefficients[1,1]
se = summary(mod)$coefficients[1,2]
adf = beta / se
adf
```

```
## [1] 0.1134663
```

```
if(int==TRUE){
  adf< (-1.616)
}else{
  adf< (-2.568)
}
```

```
## [1] FALSE
```

```
# Estimate all models and pick specification with lowest SIC - FORD
ts <- data$FORD
sics <- rep(0, comb)

for (i in 1:comb){
  logl <- arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[i,], method ="CSS")$loglik
  k = sum(is.na(lag_struc[i,]))
  sics[i] = k*log(length(ts)) - 2*logl
}
min(sics)
```

```
## [1] 241.035
```

```
arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[which.min(sics),], method ="CSS")
```

```
##
## Call:
## arima(x = ts, order = c(max_lag - 1, 0, 0), fixed = lag_struc[which.min(sics),
##     ], method = "CSS")
##
## Coefficients:
##         ar1  ar2  ar3  ar4  ar5  intercept
##       1e+00    0    0    0    0          0
## s.e.  7e-04    0    0    0    0          0
##
## sigma^2 estimated as 0.06843:  part log likelihood = -116.86
```

```
# Perform ADF Test
ts <- ts(ts)
struc <- c(0)
for (i in 1:length(lag_struc[which.min(sics),]-1)){
  if (is.na(lag_struc[which.min(sics),i])){
    struc <- cbind(struc, i)
  }
}
struc <- as.numeric(struc[,-1])
if(is.na(lag_struc[which.min(sics),i])){
  int = TRUE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc))
} else{
  int = FALSE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc)-1)
}
beta = summary(mod)$coefficients[1,1]
se = summary(mod)$coefficients[1,2]
adf = beta / se
adf
```

```
## [1] 0.06724836
```

```
if(int==TRUE){
  adf< (-1.616)
}else{
  adf< (-2.568)
}
```

```
## [1] FALSE
```

```
# Estimate all models and pick specification with lowest SIC - GEN_ELECTRIC
ts <- data$GEN_ELECTRIC
sics <- rep(0, comb)

for (i in 1:comb){
  logl <- arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[i,], method ="CSS")$loglik
  k = sum(is.na(lag_struc[i,]))
  sics[i] = k*log(length(ts)) - 2*logl
}
min(sics)
```

```
## [1] 2087.649
```

```
arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[which.min(sics),], method ="CSS")
```

```
##
## Call:
## arima(x = ts, order = c(max_lag - 1, 0, 0), fixed = lag_struc[which.min(sics),
##     ], method = "CSS")
##
## Coefficients:
##          ar1  ar2  ar3     ar4  ar5  intercept
##       0.9422    0    0  0.0571    0          0
## s.e.  0.0157    0    0  0.0157    0          0
##
## sigma^2 estimated as 0.2334:  part log likelihood = -1036.51
```

```
# Perform ADF Test
ts <- ts(ts)
struc <- c(0)
for (i in 1:length(lag_struc[which.min(sics),]-1)){
  if (is.na(lag_struc[which.min(sics),i])){
    struc <- cbind(struc, i)
  }
}
struc <- as.numeric(struc[,-1])
if(is.na(lag_struc[which.min(sics),i])){
  int = TRUE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc))
} else{
  int = FALSE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc)-1)
}
beta = summary(mod)$coefficients[1,1]
se = summary(mod)$coefficients[1,2]
adf = beta / se
adf
```

```
## [1] -1.334758
```

```
if(int==TRUE){
  adf< (-1.616)
}else{
  adf< (-2.568)
}
```

```
## [1] FALSE
```

```
# Estimate all models and pick specification with lowest SIC - INTEL
ts <- data$INTEL
sics <- rep(0, comb)

for (i in 1:comb){
  logl <- arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[i,], method ="CSS")$loglik
  k = sum(is.na(lag_struc[i,]))
  sics[i] = k*log(length(ts)) - 2*logl
}
min(sics)
```

```
## [1] 1555.822
```

```
arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[which.min(sics),], method ="CSS")
```

```
##
## Call:
## arima(x = ts, order = c(max_lag - 1, 0, 0), fixed = lag_struc[which.min(sics),
##     ], method = "CSS")
##
## Coefficients:
##          ar1  ar2  ar3  ar4  ar5  intercept
##       0.9998    0    0    0    0          0
## s.e.  0.0005    0    0    0    0          0
##
## sigma^2 estimated as 0.1645:  part log likelihood = -774.25
```

```
# Perform ADF Test
ts <- ts(ts)
struc <- c(0)
for (i in 1:length(lag_struc[which.min(sics),]-1)){
  if (is.na(lag_struc[which.min(sics),i])){
    struc <- cbind(struc, i)
  }
}
struc <- as.numeric(struc[,-1])
if(is.na(lag_struc[which.min(sics),i])){
  int = TRUE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc))
} else{
  int = FALSE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc)-1)
}
beta = summary(mod)$coefficients[1,1]
se = summary(mod)$coefficients[1,2]
adf = beta / se
adf
```

```
## [1] -0.3575228
```

```
if(int==TRUE){
  adf< (-1.616)
}else{
  adf< (-2.568)
}
```

```
## [1] FALSE
```

```
# Estimate all models and pick specification with highest SIC - MICROSOFT
ts <- data$MICROSOFT
sics <- rep(0, comb)

for (i in 1:comb){
  logl <- arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[i,], method ="CSS")$loglik
  k = sum(is.na(lag_struc[i,]))
  sics[i] = k*log(length(ts)) - 2*logl
}
min(sics)
```

```
## [1] 1963.339
```

```
arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[which.min(sics),], method ="CSS")
```

```
##
## Call:
## arima(x = ts, order = c(max_lag - 1, 0, 0), fixed = lag_struc[which.min(sics),
##     ], method = "CSS")
##
## Coefficients:
##          ar1  ar2  ar3  ar4  ar5  intercept
##       0.9998    0    0    0    0          0
## s.e.  0.0004    0    0    0    0          0
##
## sigma^2 estimated as 0.2159:  part log likelihood = -978.01
```

```
# Perform ADF Test
ts <- ts(ts)
struc <- c(0)
for (i in 1:length(lag_struc[which.min(sics),]-1)){
  if (is.na(lag_struc[which.min(sics),i])){
    struc <- cbind(struc, i)
  }
}
struc <- as.numeric(struc[,-1])
if(is.na(lag_struc[which.min(sics),i])){
  int = TRUE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc))
} else{
  int = FALSE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc)-1)
}
beta = summary(mod)$coefficients[1,1]
se = summary(mod)$coefficients[1,2]
adf = beta / se
adf
```

```
## [1] -0.4175273
```

```
if(int==TRUE){
  adf< (-1.616)
}else{
  adf< (-2.568)
}
```

```
## [1] FALSE
```

```
# Estimate all models and pick specification with highest SIC - NETFLIX
ts <- data$NETFLIX
sics <- rep(0, comb)

for (i in 1:comb){
  logl <- arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[i,], method ="CSS")$loglik
  k = sum(is.na(lag_struc[i,]))
  sics[i] = k*log(length(ts)) - 2*logl
}
min(sics)
```

```
## [1] 8576.437
```

```
arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[which.min(sics),], method ="CSS")
```

```
##
## Call:
## arima(x = ts, order = c(max_lag - 1, 0, 0), fixed = lag_struc[which.min(sics),
##     ], method = "CSS")
##
## Coefficients:
##          ar1  ar2  ar3  ar4  ar5  intercept
##       1.0001    0    0    0    0          0
## s.e.  0.0010    0    0    0    0          0
##
## sigma^2 estimated as 17.79:  part log likelihood = -4284.56
```

```
# Perform ADF Test
ts <- ts(ts)
struc <- c(0)
for (i in 1:length(lag_struc[which.min(sics),]-1)){
  if (is.na(lag_struc[which.min(sics),i])){
    struc <- cbind(struc, i)
  }
}
struc <- as.numeric(struc[,-1])
if(is.na(lag_struc[which.min(sics),i])){
  int = TRUE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc))
} else{
  int = FALSE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc)-1)
}
beta = summary(mod)$coefficients[1,1]
se = summary(mod)$coefficients[1,2]
adf = beta / se
adf
```

```
## [1] 0.1364884
```

```
if(int==TRUE){
  adf< (-1.616)
}else{
  adf< (-2.568)
}
```

```
## [1] FALSE
```

```
# Estimate all models and pick specification with highest SIC - NOKIA
ts <- data$NOKIA
sics <- rep(0, comb)

for (i in 1:comb){
  logl <- arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[i,], method ="CSS")$loglik
  k = sum(is.na(lag_struc[i,]))
  sics[i] = k*log(length(ts)) - 2*logl
}
min(sics)
```

```
## [1] 2315.22
```

```
arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[which.min(sics),], method ="CSS")
```

```
##
## Call:
## arima(x = ts, order = c(max_lag - 1, 0, 0), fixed = lag_struc[which.min(sics),
##     ], method = "CSS")
##
## Coefficients:
##          ar1  ar2  ar3     ar4  ar5  intercept
##       0.9333    0    0  0.0658    0          0
## s.e.  0.0155    0    0  0.0155    0          0
##
## sigma^2 estimated as 0.2717:  part log likelihood = -1150.3
```

```
# Perform ADF Test
ts <- ts(ts)
struc <- c(0)
for (i in 1:length(lag_struc[which.min(sics),]-1)){
  if (is.na(lag_struc[which.min(sics),i])){
    struc <- cbind(struc, i)
  }
}
struc <- as.numeric(struc[,-1])
if(is.na(lag_struc[which.min(sics),i])){
  int = TRUE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc))
} else{
  int = FALSE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc)-1)
}
beta = summary(mod)$coefficients[1,1]
se = summary(mod)$coefficients[1,2]
adf = beta / se
adf
```

```
## [1] -1.247401
```

```
if(int==TRUE){
  adf< (-1.616)
}else{
  adf< (-2.568)
}
```

```
## [1] FALSE
```

```
# Estimate all models and pick specification with highest SIC - SP500
ts <- data$SP500
sics <- rep(0, comb)

for (i in 1:comb){
  logl <- arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[i,], method ="CSS")$loglik
  k = sum(is.na(lag_struc[i,]))
  sics[i] = k*log(length(ts)) - 2*logl
}
min(sics)
```

```
## [1] 12651.75
```

```
arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[which.min(sics),], method ="CSS")
```

```
##
## Call:
## arima(x = ts, order = c(max_lag - 1, 0, 0), fixed = lag_struc[which.min(sics),
##     ], method = "CSS")
##
## Coefficients:
##          ar1  ar2    ar3  ar4  ar5  intercept
##       0.9369    0  0.063    0    0          0
## s.e.  0.0190    0  0.019    0    0          0
##
## sigma^2 estimated as 268.4:  part log likelihood = -6318.56
```

```
# Perform ADF Test
ts <- ts(ts)
struc <- c(0)
for (i in 1:length(lag_struc[which.min(sics),]-1)){
  if (is.na(lag_struc[which.min(sics),i])){
    struc <- cbind(struc, i)
  }
}
struc <- as.numeric(struc[,-1])
if(is.na(lag_struc[which.min(sics),i])){
  int = TRUE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc))
} else{
  int = FALSE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc)-1)
}
beta = summary(mod)$coefficients[1,1]
se = summary(mod)$coefficients[1,2]
adf = beta / se
adf
```

```
## [1] -0.1442613
```

```
if(int==TRUE){
  adf< (-1.616)
}else{
  adf< (-2.568)
}
```

```
## [1] FALSE
```

```
# Estimate all models and pick specification with highest SIC - YAHOO
ts <- data$YAHOO
sics <- rep(0, comb)

for (i in 1:comb){
  logl <- arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[i,], method ="CSS")$loglik
  k = sum(is.na(lag_struc[i,]))
  sics[i] = k*log(length(ts)) - 2*logl
}
min(sics)
```

```
## [1] 2507.824
```

```
arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[which.min(sics),], method ="CSS")
```

```
##
## Call:
## arima(x = ts, order = c(max_lag - 1, 0, 0), fixed = lag_struc[which.min(sics),
##     ], method = "CSS")
##
## Coefficients:
##          ar1  ar2  ar3     ar4  ar5  intercept
##       0.9379    0    0  0.0611    0          0
## s.e.  0.0161    0    0  0.0161    0          0
##
## sigma^2 estimated as 0.3089:  part log likelihood = -1246.6
```

```
# Perform ADF Test
ts <- ts(ts)
struc <- c(0)
for (i in 1:length(lag_struc[which.min(sics),]-1)){
  if (is.na(lag_struc[which.min(sics),i])){
    struc <- cbind(struc, i)
  }
}
struc <- as.numeric(struc[,-1])
if(is.na(lag_struc[which.min(sics),i])){
  int = TRUE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc))
} else{
  int = FALSE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc)-1)
}
beta = summary(mod)$coefficients[1,1]
se = summary(mod)$coefficients[1,2]
adf = beta / se
adf
```

```
## [1] -1.302492
```

```
if(int==TRUE){
  adf< (-1.616)
}else{
  adf< (-2.568)
}
```

```
## [1] FALSE
```

Based on our estimations we do not reject the unit-root hypothesis for any of the stock price time series at the 90% confidence interval. Given the big the almost common-wisdom like empirical finding that stock prices follow a random walk, e.g. Fama (1995), this finding is little surprising. Following the popular efficient market hypothesis, any randomly arriving news about the fundamental value of stocks are always reflected in stock prices such that no further structure shall exist in the time series of stock prices the argument loosely speaking goes. We leave this issue here, and do not further elaborate on the related discussion but rather point to the less disuputed general wisdom that stock price time series are integrated of order 1.

4. Assume that both the stocks of Apple and Microsoft follow a random walk process. Produce a 5-day forecast for the stocks of Apple and Microsoft. Add 95% confidence bounds to your forecasts under the assumption of Gaussian innovations. Is there any investment advice you can give on these stocks? Is their value expected to increase or decrease?

Assuming stock price $\{X_t\}$ follows a random walk:

$$X_t = X_{t-1} + \varepsilon_t \quad \varepsilon_t \sim \mathcal{N}(0, \sigma^2)$$

the h-step ahead forecast in period T reads as

$$\hat{X}_{T+h} = X_T$$

with the estimated variance of the corresponding h-step ahead forecast error

$$\hat{\mathbb{V}}(e_{T+h}) = h\hat{\sigma}^2 = h\frac{1}{T-1}\sum_{t=2}^{T}(x_t - x_{t-1})^2$$

Under the Gaussianity assumption, the 95%-confidence intervals for $\hat{X}_{T+h}$ then formally read as follows:

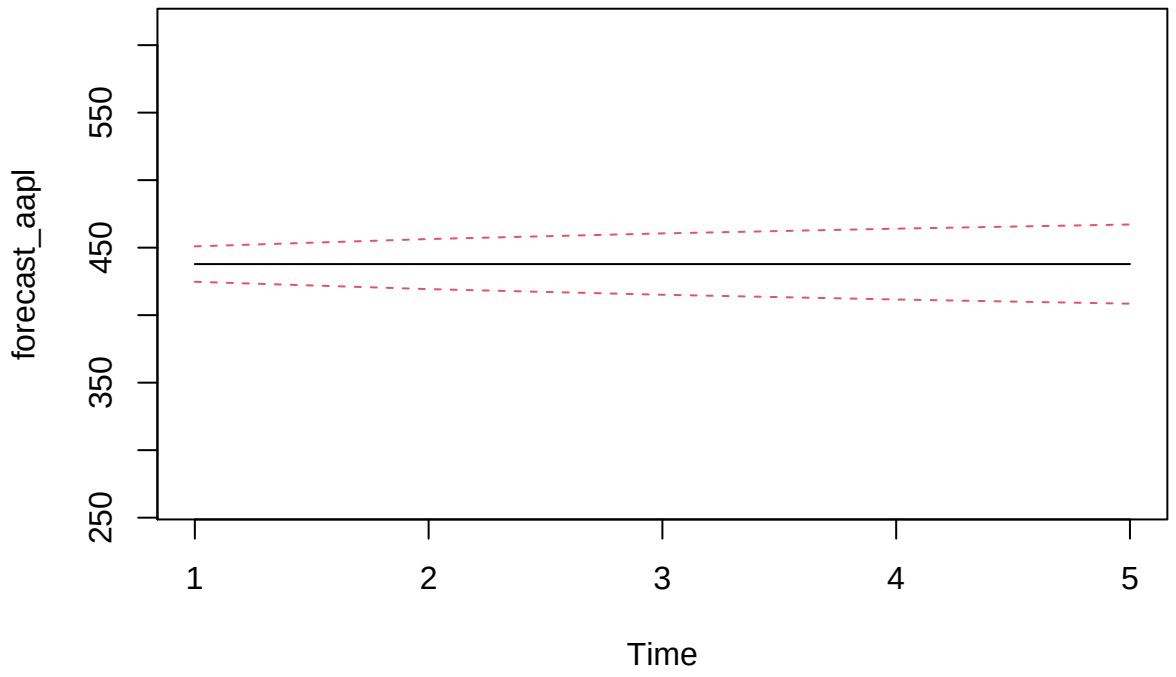
$$\hat{X}_{T+h} \pm 1.96 \cdot \sqrt{\hat{\mathbb{V}}(e_{T+h})}$$

```
# Estimate sigma for APPLE & MICROSOFT
sigma_aapl = sum(diff(data$APPLE)^2) / (length(diff(data$APPLE)))
sigma_msft = sum(diff(data$MICROSOFT)^2) / (length(diff(data$MICROSOFT)))

# Produce forecast
h = 5
forecast_aapl <-  rep( tail(data$APPLE,1),h)
forecast_msft <- rep( tail(data$MICROSOFT,1),h)
forecast_aapl <-  ts(forecast_aapl)
forecast_msft <- ts(forecast_msft)

#Produce confice interval
confup_aapl <- tail(data$APPLE,1) + 1.96 * sqrt(1:h) * sqrt(sigma_aapl)
confdown_aapl <- tail(data$APPLE,1) - 1.96 *sqrt(1:h) * sqrt(sigma_aapl)
confup_aapl <- ts(confup_aapl)
confdown_aapl <- ts(confdown_aapl)
```

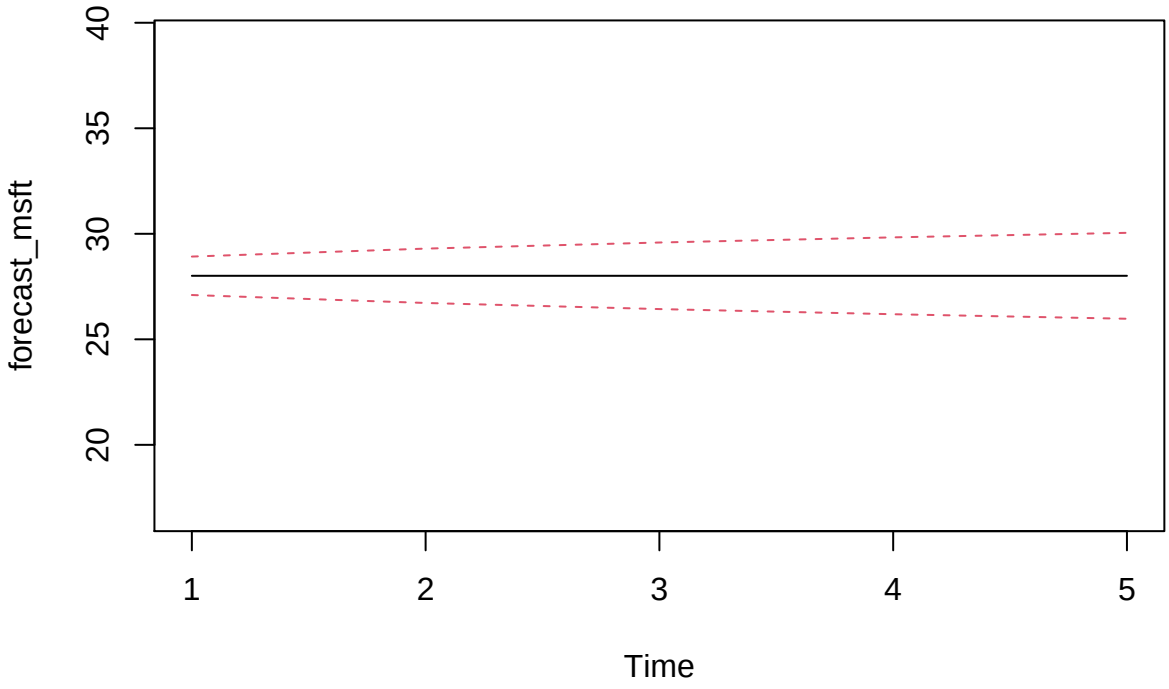
```
confup_msft <- tail(data$MICROSOFT,1) + 1.96 * sqrt(1:h) * sqrt(sigma_msft)
confdown_msft <- tail(data$MICROSOFT,1) - 1.96 * sqrt(1:h) * sqrt(sigma_msft)
confup_msft <- ts(confup_msft)
confdown_msft <- ts(confdown_msft)
```

The forecasts and confidence intervals for both Apple and Microsoft are plotted below.

```
# Generate plot APPLE
ts.plot(forecast_aapl)
points(confup_aapl, type="l", col=2, lty=2)
points(confdown_aapl, type="l", col=2, lty=2)
```




```
# Generate plot MICROSOFT
ts.plot(forecast_msft)
points(confup_msft, type="l", col=2, lty=2)
points(confdown_msft, type="l", col=2, lty=2)
```



Based on these forecasts the only investment advice would be to hold the stock since no movement can be inferred from a random walk. This shows the unpredictability of stock prices implied by the random walk thesis briefly discussed above.

5. Do you find a statistically significant contemporaneous relation between Microsoft and Exxon Mobile stock prices? Do you agree that changes in Microsoft stock prices are largely explained by fluctuations in the stock price of Exxon Mobile? Justify your answer.

To test whether there is a statistically significant contemporaneous relation between Microsoft and Exxon Mobile stock prices we estimate a simple regression model:

$$Microsoft_t = \alpha + \beta Exxon_t + \varepsilon_t$$

```
mod <- lm(data$MICROSOFT ~ data$EXXON_MOBIL)
summary(mod)
```

```
##
## Call:
## lm(formula = data$MICROSOFT ~ data$EXXON_MOBIL)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.2357  -1.7355   0.2203   1.9912   8.3225
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      11.244927   0.712719   15.78   <2e-16 ***
## data$EXXON_MOBIL  0.202856   0.009054   22.40   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.229 on 1497 degrees of freedom
## Multiple R-squared:  0.2511, Adjusted R-squared:  0.2506
## F-statistic: 501.9 on 1 and 1497 DF,  p-value: < 2.2e-16
```

The regression output suggests that there indeed exists some contemporenous relation since the estimated $\hat{\beta}$-coefficient is significantly positive suggesting that the two stocks move in tandem on average, such that one could explain at least some part of the variation in the stock price of Microsoft with that of Exxon Mobile. Under standard Asset Pricing model cross-sectional correlation of stocks may very well be seen as causality since stock may share common risk factors. However, given our conclusion above that both stocks are non-stationary (i.e. we could not reject the null hypothesis of a unit-root) one should be wary of the spurious regression phenomenon analyzed in part 1. Especially if one were to assume that both stock price time series follow a random walk, our simulation results suggest that both the $\hat{\beta}$-coefficient as well as estimated t-statistics converge to a random variable rendering the obtained regression results spurious and not be taken seriously.

Question 2

1. Show that the spurious regression problem does not occur when two I(1) variables are cointegrated using a Monte Carlo simulation study. Furthermore, analyze the claim of superconsistency of the estimator of the static cointegrating regression.

First, we run a Monte Carlo Simulation for both cases, with X being once non-stationary but co-integrated with Y and once stationary (and also cointegrated with Y). The simulation is run for three different T, namely $T = \{50, 100, 200\}$.

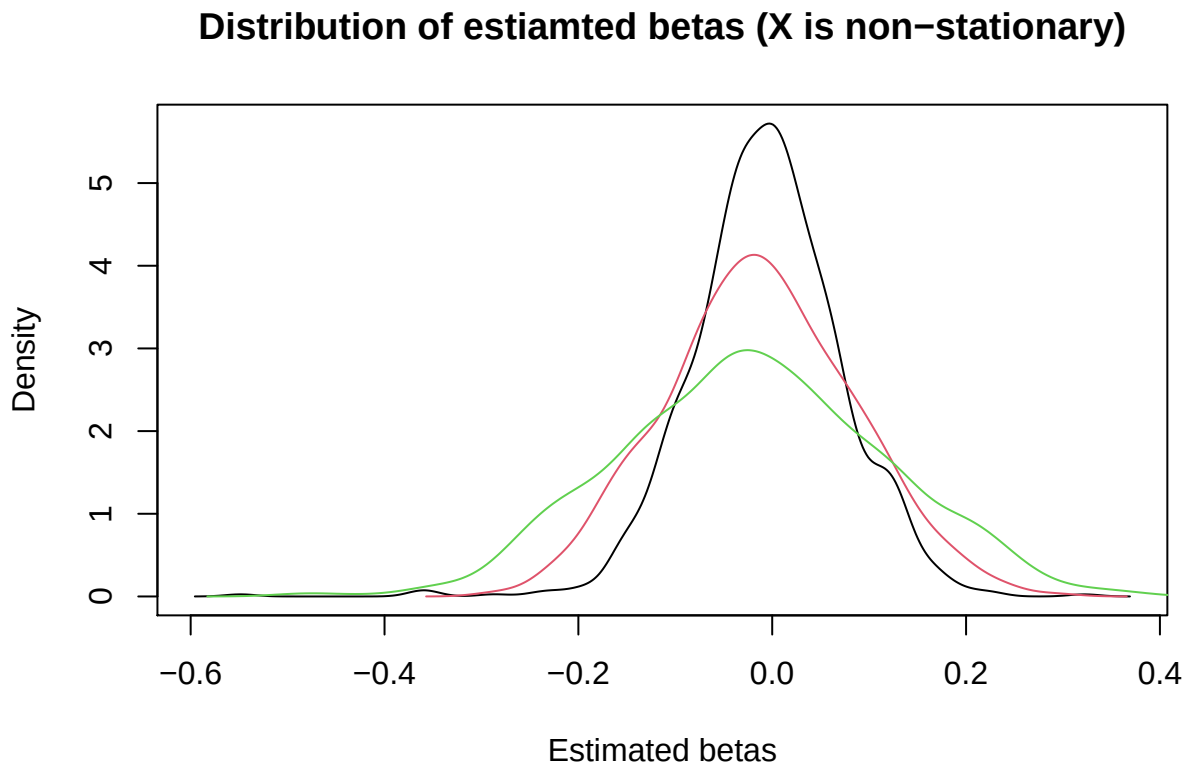
```
# Simulation with x being non-stationary
set.seed(20)
l = c(50, 100, 200)
n = 1000

sigma_u = 1
sigma_v = 1

gamma = 0.8
phi = 1

# Create the list to store dataframes
simulationsX <- list()
simulationsY <- list()


# Generate Random walk
for (j in 1:length(l)){
  t = l[j]
  x <- data.frame(rnorm(n,0,sigma_v))
  y <- data.frame(rnorm(n,0,sigma_u))
  for (i in 1:(t-1)){
    v <- as.numeric(x[i,]) + phi * rnorm(n,0,sigma_v)
    x <- data.frame(x, v)

    u <- as.numeric(x[(i+1),]) + gamma * rnorm(n,0,sigma_u)
    y <- data.frame(y, u)
  }
```

```
  simulationsX[j] <- list(data.frame(x))
  simulationsY[j] <- list(data.frame(y))
}

# Run regressions & store beta, SE and R^2
list_est <- list()

for (j in 1:length(l)){
  est <- data.frame(matrix(ncol = 3, nrow = n))
  for (i in 1:n){
    reg <- summary(lm(as.numeric(simulationsY[[j]][i,])~ as.numeric(simulationsX[[j]][i,])))
    est[i,1] <- reg$coef[2,1]
    est[i,2] <- reg$coef[2,1]/reg$coef[2,2]
    est[i,3] <- reg$r.squared
  }
  list_est[j] <- list(data.frame(est))
}
```

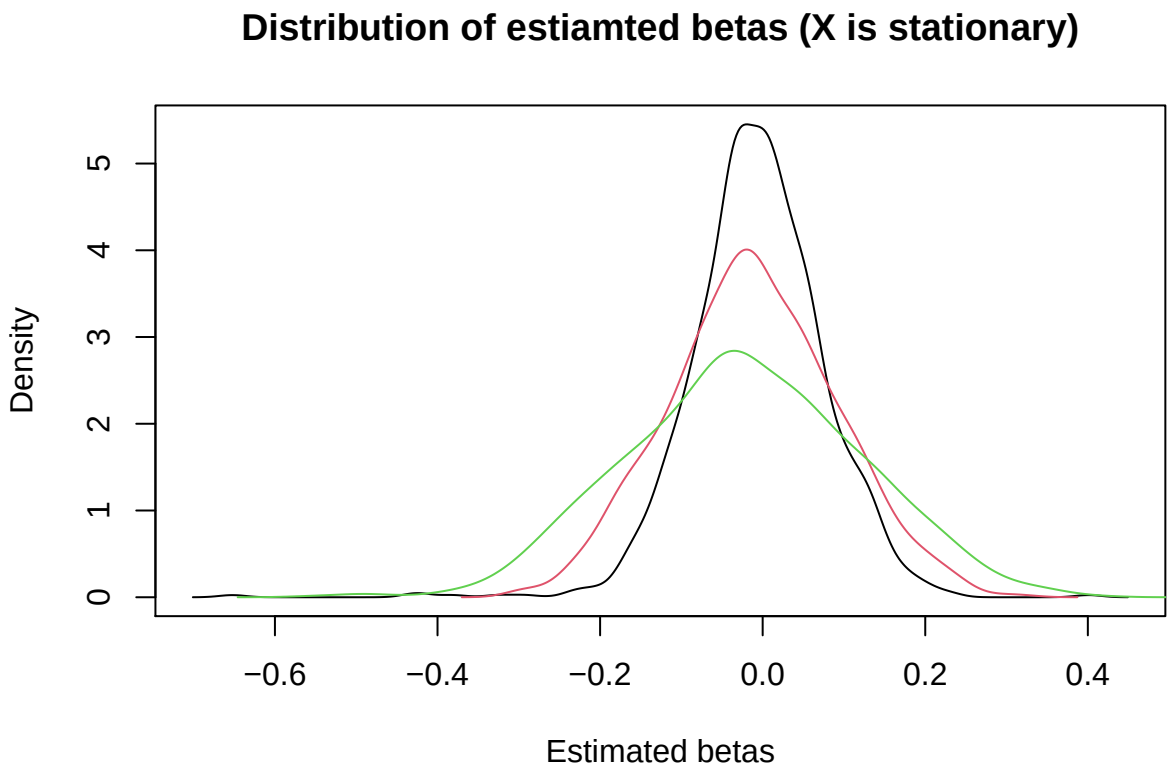
```
# Simulation with x being stationary
set.seed(20)
l = c(50, 100, 200)
n = 1000

sigma_u = 1
sigma_v = 1

gamma = 0.8
phi = 0.8

# Create the list to store dataframes
simulationsX_2 <- list()
simulationsY_2 <- list()


# Generate Random walk
for (j in 1:length(l)){
  t = l[j]
  x <- data.frame(rnorm(n,0,sigma_v))
  y <- data.frame(rnorm(n,0,sigma_u))
  for (i in 1:(t-1)){
    v <- as.numeric(x[i,]) + phi * rnorm(n,0,sigma_v)
    x <- data.frame(x, v)

    u <- as.numeric(x[(i+1),]) + gamma * rnorm(n,0,sigma_u)
    y <- data.frame(y, u)
  }
  simulationsX_2[j] <- list(data.frame(x))
  simulationsY_2[j] <- list(data.frame(y))
}

# Run regressions & store beta, SE and R^2
list2_est <- list()

for (j in 1:length(l)){
```

```
  est <- data.frame(matrix(ncol = 3, nrow = n))
  for (i in 1:n){
    reg <- summary(lm(as.numeric(simulationsY_2[[j]][i,])~ as.numeric(simulationsX_2[[j]][i,])))
    est[i,1] <- reg$coef[2,1]
    est[i,2] <- reg$coef[2,1]/reg$coef[2,2]
    est[i,3] <- reg$r.squared
  }
  list2_est[j] <- list(data.frame(est))
}
```

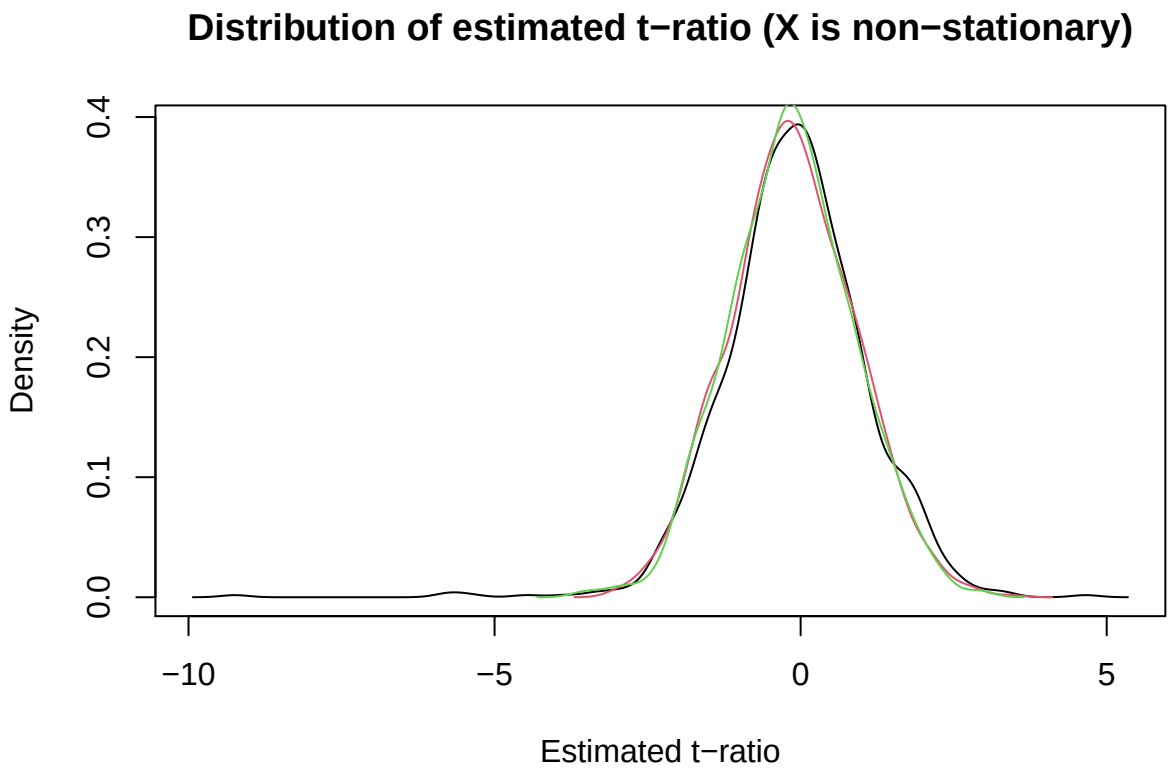
```
# Plot histograms
plot(density(list_est[[3]][,1])$x,density(list_est[[3]][,1])$y,type="l",xlab="Estimated betas",ylab="Den
points(density(list_est[[2]][,1])$x, density(list_est[[2]][,1])$y, type="l",col=2)
points(density(list_est[[1]][,1])$x, density(list_est[[1]][,1])$y, type="l",col=3)
```



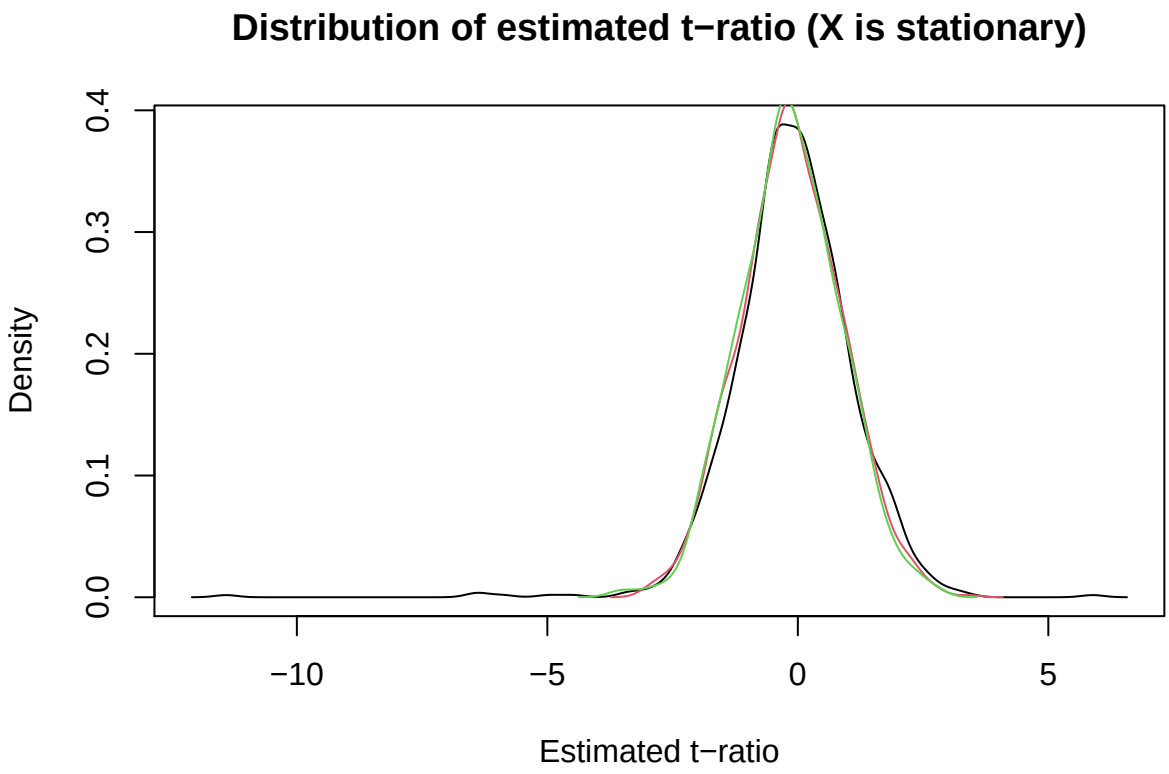

```
plot(density(list2_est[[3]][,1])$x,density(list2_est[[3]][,1])$y,type="l",xlab="Estimated betas",ylab="D
points(density(list2_est[[2]][,1])$x, density(list2_est[[2]][,1])$y, type="l",col=2)
points(density(list2_est[[1]][,1])$x, density(list2_est[[1]][,1])$y, type="l",col=3)
```



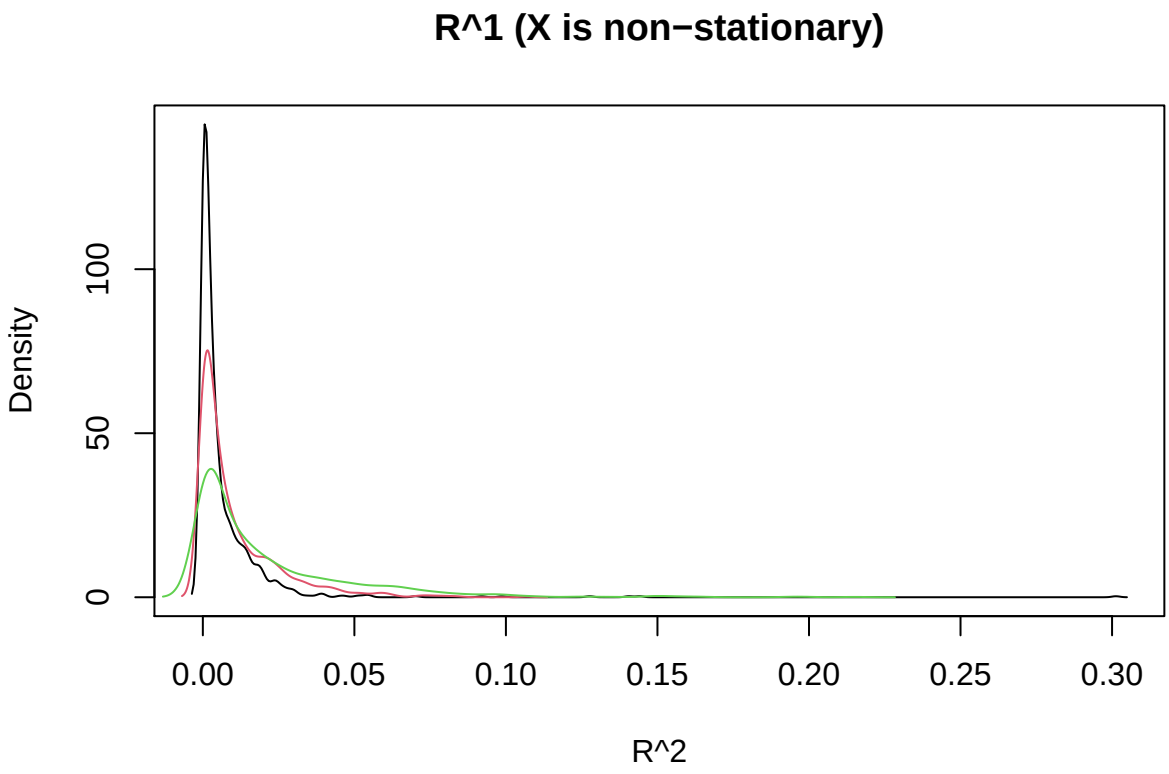

```
plot(density(list_est[[3]][,2])$x,density(list_est[[3]][,2])$y,type="l",xlab="Estimated t-ratio",ylab="
points(density(list_est[[2]][,2])$x, density(list_est[[2]][,2])$y, type="l",col=2)
points(density(list_est[[1]][,2])$x, density(list_est[[1]][,2])$y, type="l",col=3)
```



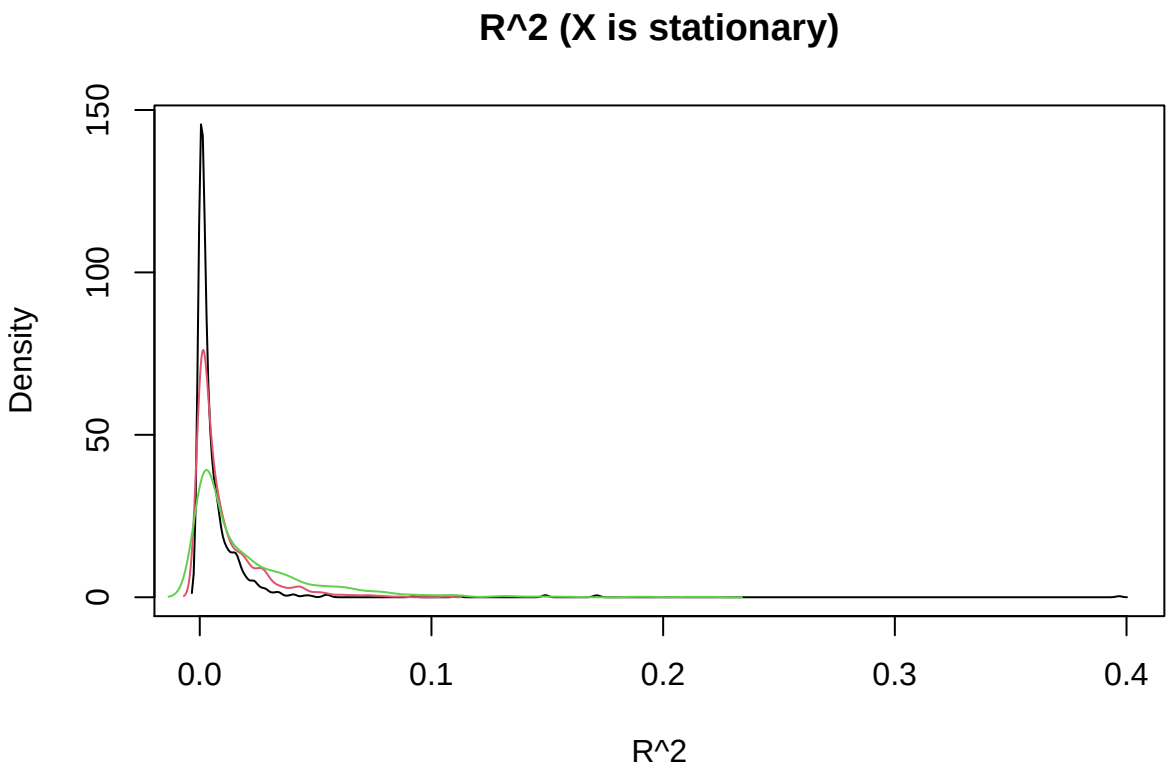

```
plot(density(list2_est[[3]][,2])$x,density(list2_est[[3]][,2])$y,type="l",xlab="Estimated t-ratio",ylab=
points(density(list2_est[[2]][,2])$x, density(list2_est[[2]][,2])$y, type="l",col=2)
points(density(list2_est[[1]][,2])$x, density(list2_est[[1]][,2])$y, type="l",col=3)
```




```
plot(density(list_est[[3]][,3])$x,density(list_est[[3]][,3])$y,type="l",xlab="R^2",ylab="Density", col=
points(density(list_est[[2]][,3])$x, density(list_est[[2]][,3])$y, type="l",col=2)
points(density(list_est[[1]][,3])$x, density(list_est[[1]][,3])$y, type="l",col=3)
```




```
plot(density(list2_est[[3]][,3])$x,density(list2_est[[3]][,3])$y,type="l",xlab="R^2",ylab="Density", co
points(density(list2_est[[2]][,3])$x, density(list2_est[[2]][,3])$y, type="l",col=2)
points(density(list2_est[[1]][,3])$x, density(list2_est[[1]][,3])$y, type="l",col=3)
```



Based on the plots, we can see that β goes against zero for large samples in both cases (X being either non-stationary or stationary). Further, for both cases the t-statistic is normal and R^2 tends against zero for large samples. We cannot observe any randomness across the different beta, t-ratio and R^2 estimations. This shows that the spurious regression problem does not occur when two non-stationary processes are co-integrated.

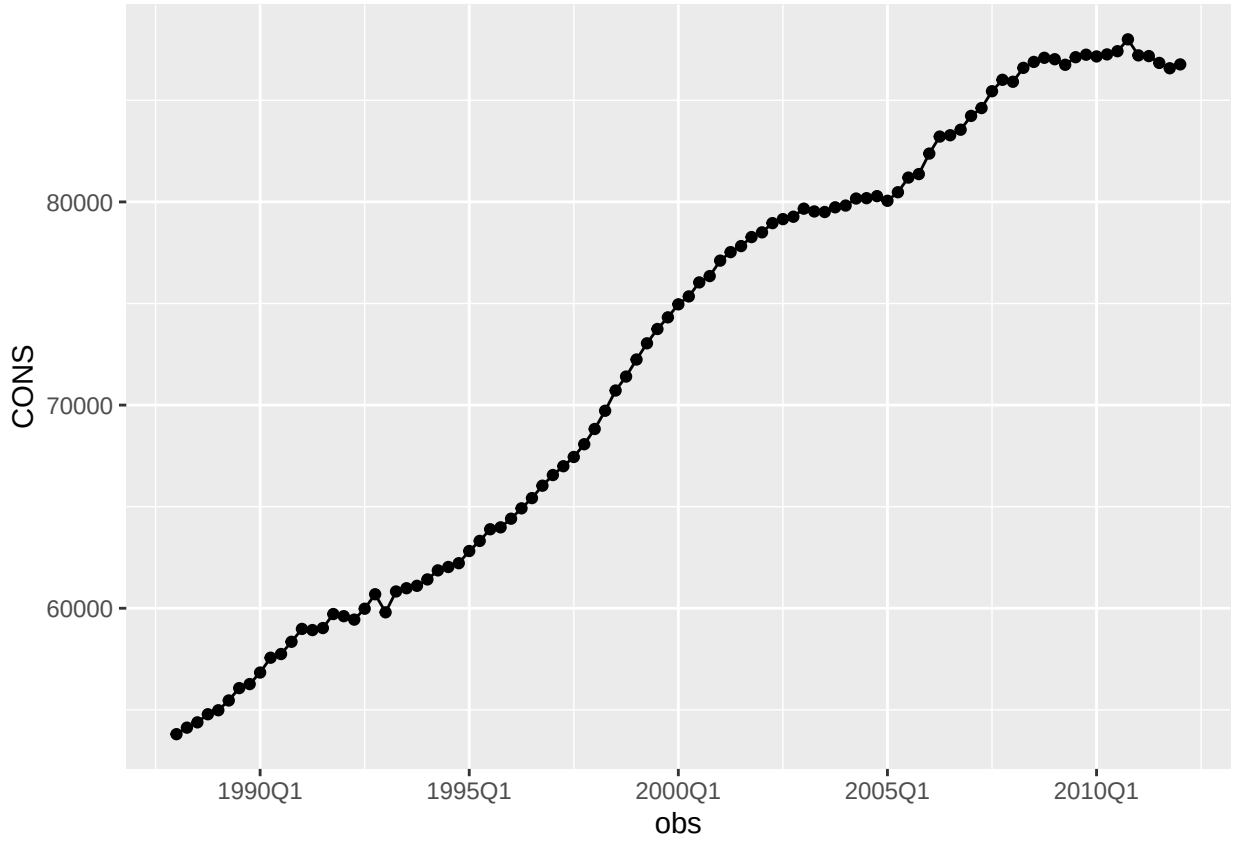
The higher our T becomes (note that the black line refers to $T = 200$, the red line refers to $T = 100$ and the green line refers to $T = 50$), the lower the distribution of the estimated β as well as the distribution of the R^2. The distribution of the t-statistic is approximately the same for every T.

2. Plot both the aggregate consumption and aggregate income time-series (these series are called cons and inc respectively in the csv file). Compute and report 12-period ACF and PACF functions for each series and comment on their shape

```
# Load data
data <- read.csv("data_assign_p4.csv")
```

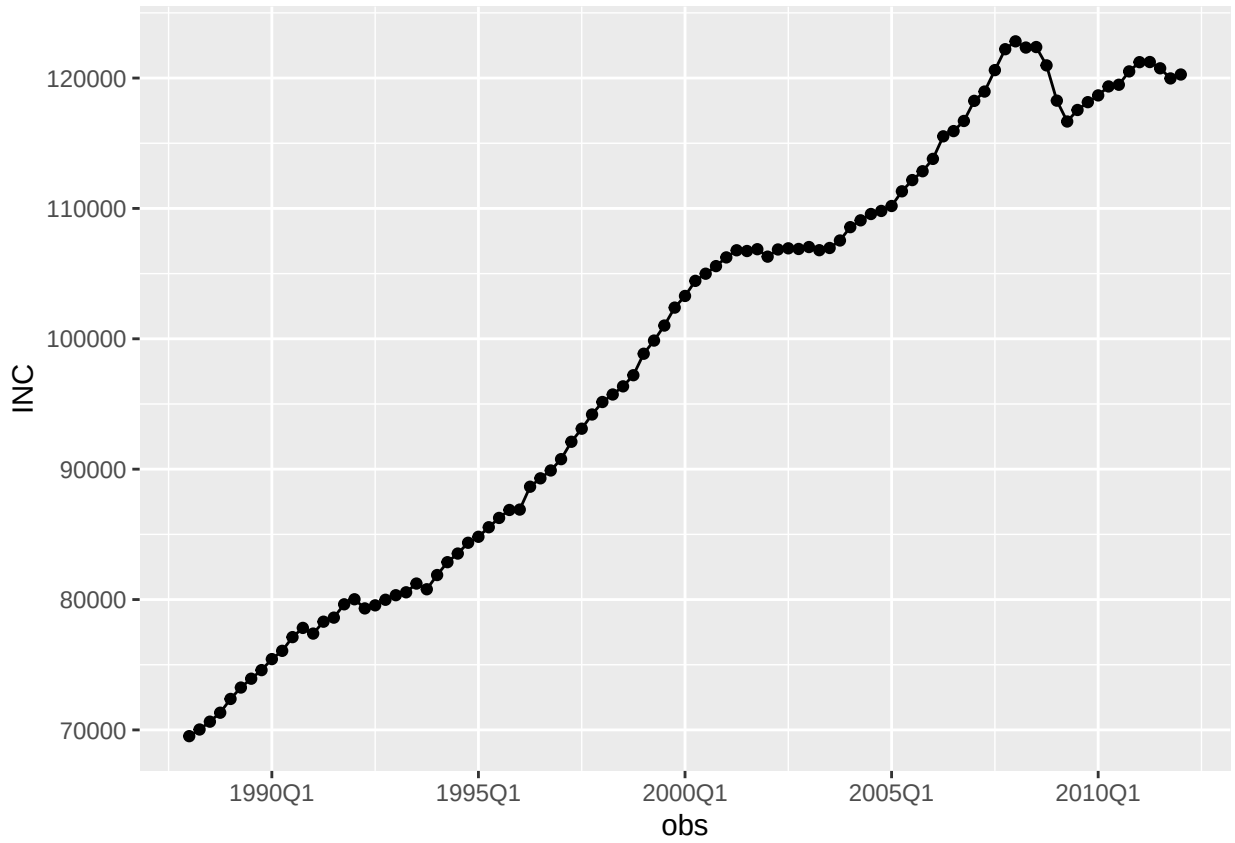
```
fmt <- "%YQ%q"
data$obs <- as.yearqtr(data$obs, format = fmt)
```

```
# Plot of consumption
ggplot(data, aes(obs, CONS)) +
  geom_point() +
  geom_line() +
  scale_x_yearqtr(format = fmt)
```



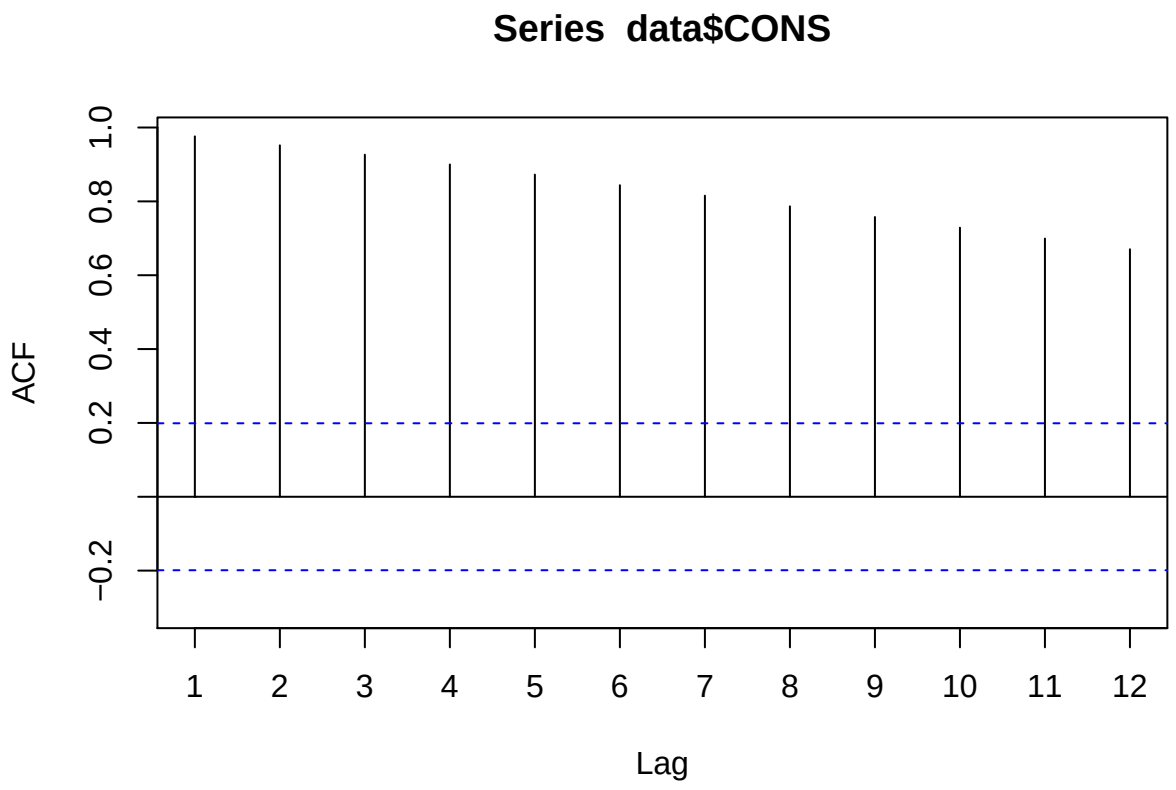
The graph shows that consumption steadily increaside since 1990.

```
# Plot of income
ggplot(data, aes(obs, INC)) +
  geom_point() +
  geom_line() +
  scale_x_yearqtr(format = fmt)
```

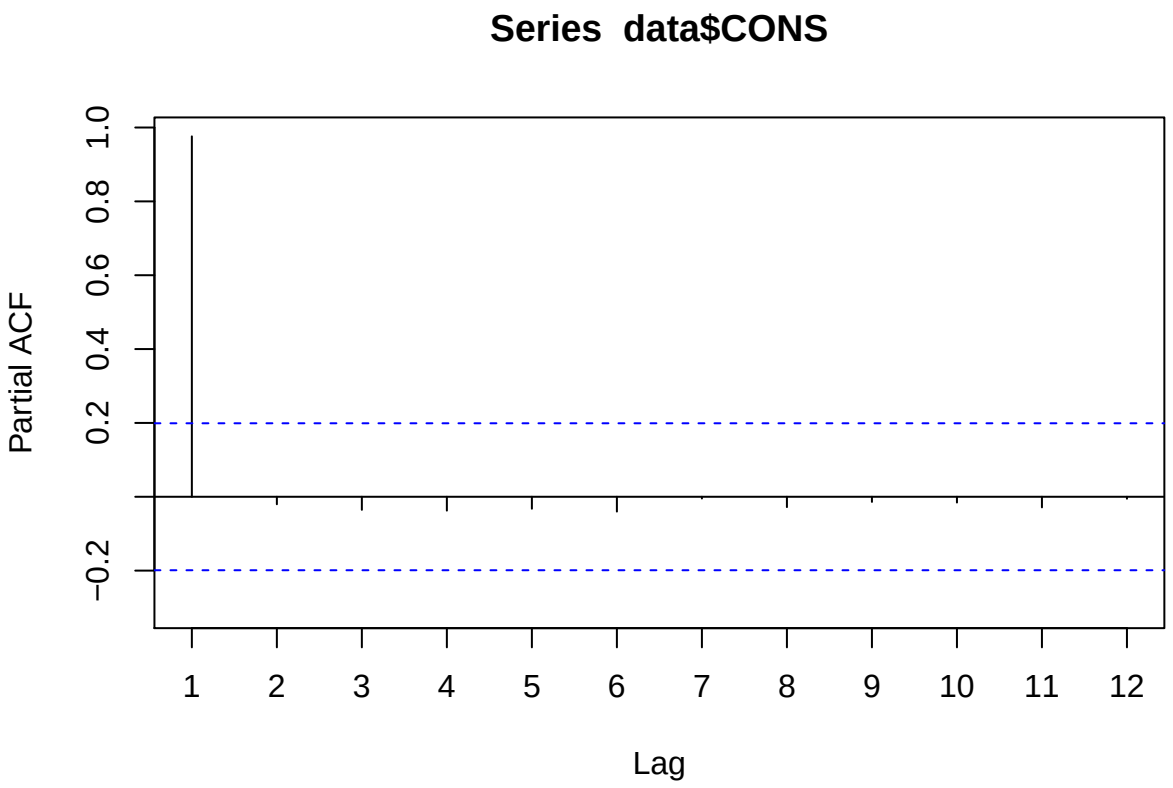


The graph shows that income steadily increased since 1990. However, around 2008 we can observe a small fall which is followed by increase again.

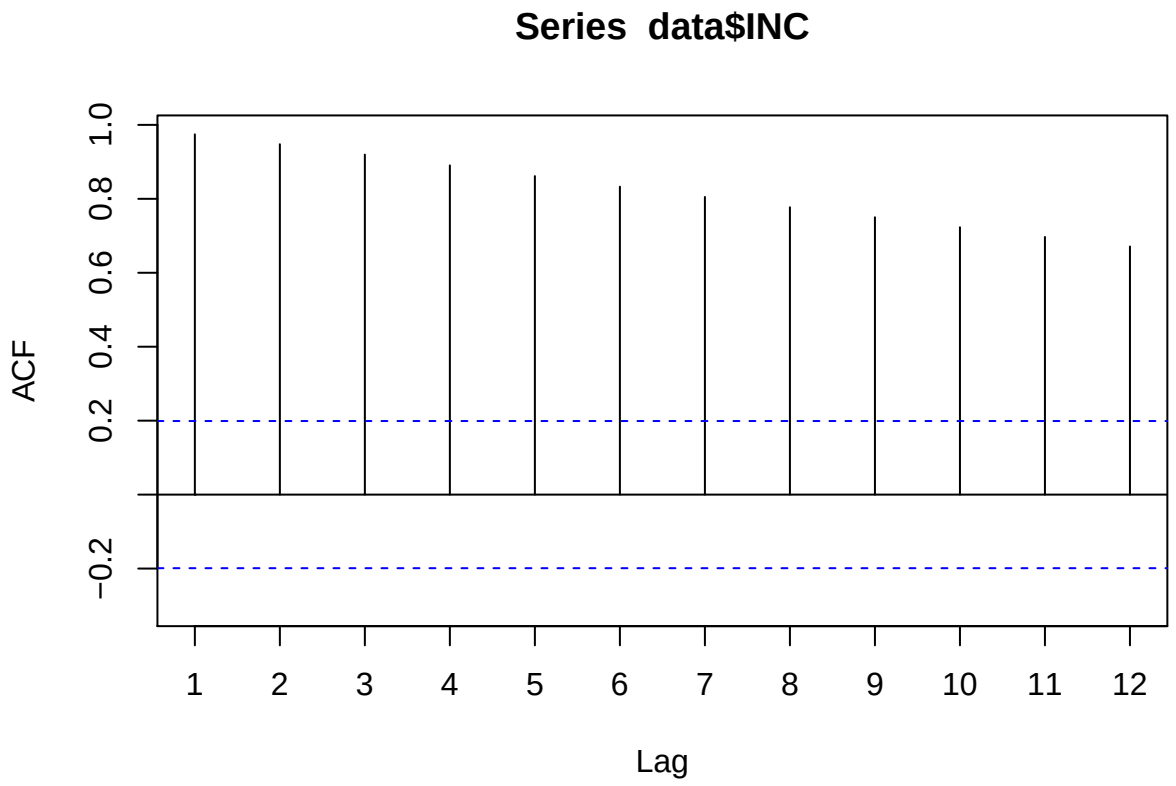
```
Acf(
  data$CONS,
  lag.max = 12
)
```



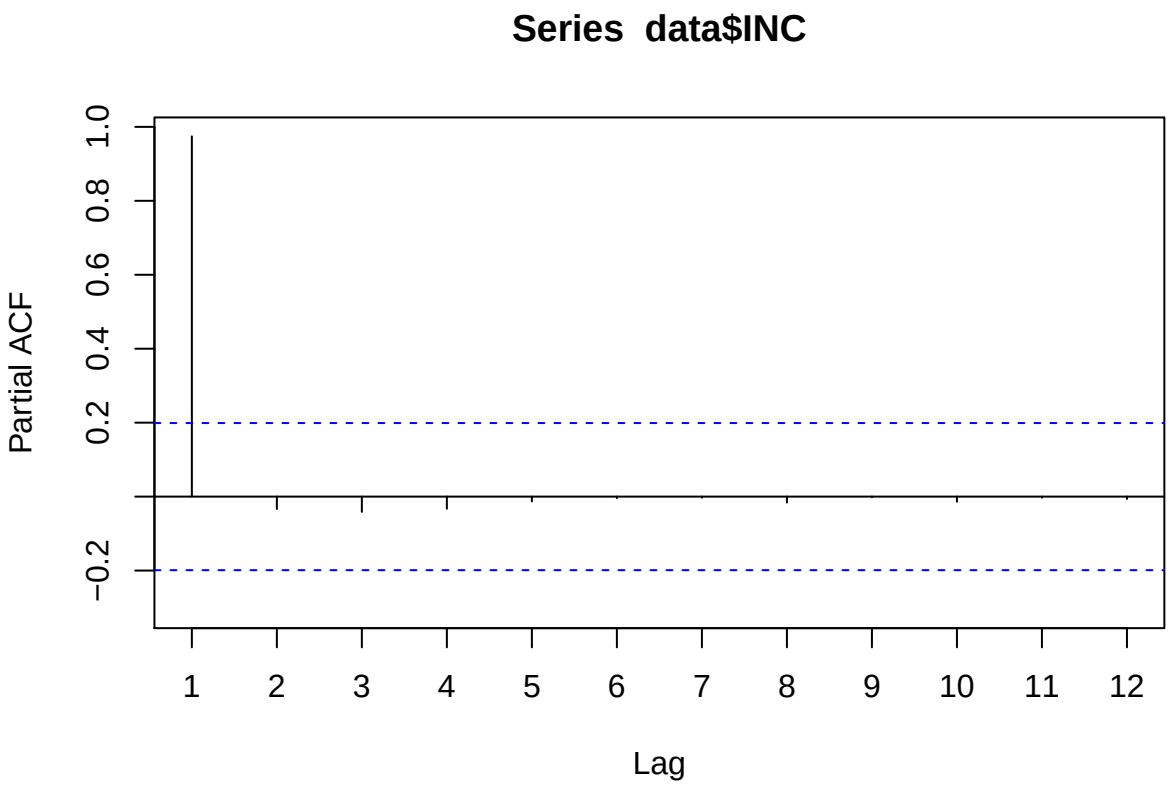

```
Pacf(
  data$CONS,
  lag.max = 12
)
```




```
Acf(
  data$INC,
  lag.max = 12
)
```




```
Pacf(
  data$INC,
  lag.max = 12
)
```



The ACF is the autocorrelation function and measures the autocorrelation between X_t and $X_{t-h} \forall h$, in our case for $h = \{1, 2, ..., 12\}$. The autocorrelation is an indicator for memory: the higher the correlation, the higher the time dependence. Hence, the ACF also is a good first indicator for the selection of the lags of our time series model (corresponding to the Box-Jenkins approach). The partial autocorrelation function (PACF) goes one step further, controlling for other lags. According to the graphs of both time series, we would have to include all 12 lags. However, according to the PACF graph, only the first lag is significant which shows that we need to control for other lags. Note that the confidence interval corresponds to 95%.

3. Perform an ADF unit-root test on each series using the general-to-specific approach and report the values of the test statistics. Is the unit-root hypothesis rejected in any of them?

```
# Set combinatorics where 1 last lag is intercept
max_lag = 6
lag <- c(1:max_lag)
comb_set <- CombSet(lag, m=1:max_lag)
comb = 0
for (i in 1:max_lag){
  comb = comb + CombN(max_lag,i)
}
lag_struc <- matrix(0,nrow = comb, ncol =max_lag)
lag_max <- rep(0, comb)

# Lag 1
lag_n = 1
for (i in 1:length(comb_set[[lag_n]])){
  lag = comb_set[[lag_n]][i]
  lag_struc[i,lag] = NA
}
```

```
# Lag 2 and higher
start = 0
for (n in 2:max_lag){
  lag_n = n
  start = start + length(comb_set[[lag_n-1]][,1])
  for (i in 1:length(comb_set[[lag_n]][,1])){
    for (j in 1:lag_n){
      lag = comb_set[[lag_n]][i,j]
      lag_struc[start + i,lag] = NA
    }
  }
}

# Delete model with only intercept
lag_struc <- lag_struc[-max_lag,]
comb = comb -1
```

```
# Estimate all models and pick specification with lowest SIC - CONS
ts <- data$CONS
sics <- rep(0, comb)

for (i in 1:comb){
  logl <- arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[i,], method ="CSS")$loglik
  k = sum(is.na(lag_struc[i,]))
  sics[i] = k*log(length(ts)) - 2*logl
}
min(sics)
```

```
## [1] 1407.379
```

```
arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[which.min(sics),], method ="CSS")
```

```
##
## Call:
## arima(x = ts, order = c(max_lag - 1, 0, 0), fixed = lag_struc[which.min(sics),
##     ], method = "CSS")
##
## Coefficients:
##          ar1  ar2     ar3      ar4  ar5  intercept
##       1.1345    0  0.3202  -0.4536    0          0
## s.e.  0.0604    0  0.1312   0.0985    0          0
##
## sigma^2 estimated as 101691:  part log likelihood = -696.83
```

```
# Perform ADF Test
ts <- ts(ts)
struc <- c(0)
for (i in 1:length(lag_struc[which.min(sics),]-1)){
  if (is.na(lag_struc[which.min(sics),i])){
    struc <- cbind(struc, i)
  }
}
struc <- as.numeric(struc[,-1])
#struc <- head(struc,-1)
if(is.na(lag_struc[which.min(sics),i])){
```

```
  int = TRUE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc))
} else{
  int = FALSE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc)-1)
}
beta = summary(mod)$coefficients[1,1]
se = summary(mod)$coefficients[1,2]
adf = beta/se
adf
```

```
## [1] 1.505432
```

```
if(int==TRUE){
  adf< (-1.616)
}else{
  adf< (-2.568)
}
```

```
## [1] FALSE
```

We use the same code as in the previous questions. For consumption the specification with the lowest SIC is an AR(4) model, without the second lag. The estimated augmented Dickey-Fuller statistic is around 1.51, which is above the critical value of 10%. Hence, we cannot reject the H_0 which is an indicator that we have a unit root.

```
# Estimate all models and pick specification with lowest SIC - INC
ts <- data$INC
sics <- rep(0, (comb-1))

for (i in 1:(comb-1)){ # there is an issue with (1,1,1,1,1,1,)
  logl <- arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[i,], method ="CSS")$loglik
  k = sum(is.na(lag_struc[i,]))
  sics[i] = k*log(length(ts)) - 2*logl
}
min(sics)
```

```
## [1] 1526.973
```

```
arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[which.min(sics),], method ="CSS")
```

```
##
## Call:
## arima(x = ts, order = c(max_lag - 1, 0, 0), fixed = lag_struc[which.min(sics),
##     ], method = "CSS")
##
## Coefficients:
##          ar1      ar2  ar3  ar4  ar5  intercept
##       1.5393  -0.5371    0    0    0          0
## s.e.  0.0874   0.0878    0    0    0          0
##
## sigma^2 estimated as 365780:  part log likelihood = -758.91
```

```
# Perform ADF Test
ts <- ts(ts)
struc <- c(0)
for (i in 1:length(lag_struc[which.min(sics),]-1)){
```

```
  if (is.na(lag_struc[which.min(sics),i])){
    struc <- cbind(struc, i)
  }
}
struc <- as.numeric(struc[,-1])
#struc <- head(struc,-1)
if(is.na(lag_struc[which.min(sics),i])){
  int = TRUE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc))
} else{
  int = FALSE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc)-1)
}
beta = summary(mod)$coefficients[1,1]
se = summary(mod)$coefficients[1,2]
adf = beta/se
adf
```

```
## [1] 2.569735
```

```
if(int==TRUE){
  adf< (-1.616)
}else{
  adf< (-2.568)
}
```

```
## [1] FALSE
```

For income the specification with the lowest SIC is an AR(2) model. The estimated augmented Dickey-Fuller statistic is around 2.57, which is above the critical value of 10%. Hence, we cannot reject the H_0 which is an indicator that we have a unit root.

4. Perform an ADF unit-root test on the first difference of each series using the general-to specific approach and report the values of the test statistics. Is the unit-root hypothesis rejected in any of them? What do you conclude about the order of integration of these time series?

To perform an ADF unit-root test on the first difference, we calculate first the first-differences of each time series and then apply the same procedure as before.

```
# Calculate first differences of income and consumption data
data2 <- data %>%
  mutate_at(vars(INC:CONS), list(~ .x - lag(.x)))
```

```
# Estimate all models and pick specification with lowest SIC - CONS
ts <- data2$CONS
sics <- rep(0, comb)

for (i in 1:comb){
  logl <- arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[i,], method ="CSS")$loglik
  k = sum(is.na(lag_struc[i,]))
  sics[i] = k*log(length(ts)) - 2*logl
}
min(sics)
```

```
## [1] 1390.808
```

```
arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[which.min(sics),], method ="CSS")
```

```
##
## Call:
## arima(x = ts, order = c(max_lag - 1, 0, 0), fixed = lag_struc[which.min(sics),
##     ], method = "CSS")
##
## Coefficients:
##       ar1     ar2    ar3     ar4  ar5  intercept
##         0  0.1972  0.493  0.2025    0          0
## s.e.    0  0.0863  0.083  0.0863    0          0
##
## sigma^2 estimated as 99384:  part log likelihood = -688.54
```

```
# Perform ADF Test
ts <- ts(ts)
struc <- c(0)
for (i in 1:length(lag_struc[which.min(sics),]-1)){
  if (is.na(lag_struc[which.min(sics),i])){
    struc <- cbind(struc, i)
  }
}
struc <- as.numeric(struc[,-1])
#struc <- head(struc,-1)
if(is.na(lag_struc[which.min(sics),i])){
  int = TRUE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc))
} else{
  int = FALSE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc)-1)
}
beta = summary(mod)$coefficients[1,1]
se = summary(mod)$coefficients[1,2]
adf = beta/se
adf
```

```
## [1] -4.746404
```

```
if(int==TRUE){
  adf< (-1.616)
}else{
  adf< (-2.568)
}
```

```
## [1] TRUE
```

For consumption, the specification with the lowest SIC is an AR(4) model, with the first lag exluded. The test statistic is around -4.746. As it is below the critical value of 1%, we can reject the H_0. Hence, we can conclude that the order of integration of consumption is 1 as the order of integration of its first difference is zero.

```
# Estimate all models and pick specification with lowest SIC - INC
ts <- data2$INC
sics <- rep(0, comb)

for (i in 1:comb){
  logl <- arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[i,], method ="CSS")$loglik
  k = sum(is.na(lag_struc[i,]))
  sics[i] = k*log(length(ts)) - 2*logl
```

```
}
min(sics)
```

```
## [1] 1510.412
```

```
arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[which.min(sics),], method ="CSS")
```

```
##
## Call:
## arima(x = ts, order = c(max_lag - 1, 0, 0), fixed = lag_struc[which.min(sics),
##     ], method = "CSS")
##
## Coefficients:
##          ar1  ar2  ar3  ar4  ar5  intercept
##       0.5082    0    0    0    0   510.1836
## s.e.  0.0878    0    0    0    0   124.9493
##
## sigma^2 estimated as 362316:  part log likelihood = -750.63
```

```
# Perform ADF Test
ts <- ts(ts)
struc <- c(0)
for (i in 1:length(lag_struc[which.min(sics),]-1)){
  if (is.na(lag_struc[which.min(sics),i])){
    struc <- cbind(struc, i)
  }
}
struc <- as.numeric(struc[,-1])
#struc <- head(struc,-1)
if(is.na(lag_struc[which.min(sics),i])){
  int = TRUE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc))
} else{
  int = FALSE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc)-1)
}
beta = summary(mod)$coefficients[1,1]
se = summary(mod)$coefficients[1,2]
adf = beta/se
adf
```

```
## [1] 2.815002
```

```
if(int==TRUE){
  adf< (-1.616)
}else{
  adf< (-2.568)
}
```

```
## [1] FALSE
```

For income, the specification with the lowest SIC is an AR(1) model. The test statistic is around 2.82. As it is above the critical value of 10%, we cannot reject the H_0. Hence, we can conclude that the order of integration of income must be above 1. Given that we are searching for co-integration and that in economic data an integration order above 1 is very seldom, there could be a mistake. One possible source could be the programming mistake we elaborated on in the previous question. In case the order of integration is indeed higher, we would have to continue taking differences until we can reject the H_0 to find the order of

integration.

5. Assuming both series are I(1), test for cointegration between consumption and income by regressing consumption on income and performing a unit-root test on the residuals. Report the estimated regression coefficients. Plot the regression residuals. Use the Schwartz Information Criterion (SIC) to determine the number of ADF lags in your unit-root residual test. Report the cointegration test statistic. Do you reject cointegration?

First, we run a static regression of consumption on income $C_t = \delta + \lambda I_t + Z_t$ (note that C_t is consumption and I_t is income in t):

```
# Regress Consumption on Income to get get estimated regression coefficients
test <- lm(data$CONS ~ data$INC)
lambda <- test$coefficients[2]
delta <- test$coefficients[1]
lambda
```

```
##  data$INC
## 0.6651404
```

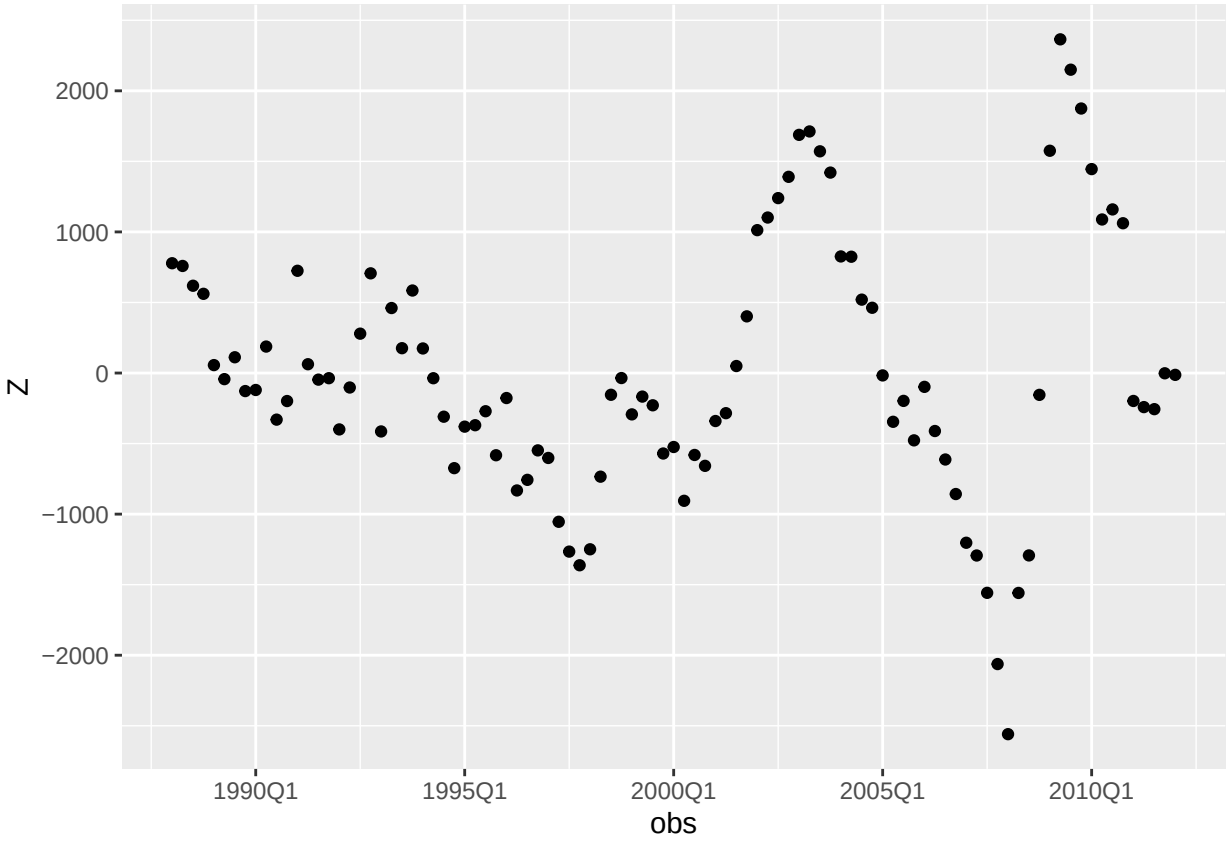
```
delta
```

```
## (Intercept)
##    6783.373
```

Our estimated regression coefficients are $\lambda \approx 0.665$ and $\delta \approx 6783.37$. Next, we calculate the regression residuals $\hat{Z}_t$ using our estimated regression coefficients and plot them. The estimated regression residuals have a mean approximately around zero. However, their variance increases with time.

```
# Calculate regression residuals
data$Z <- data$CONS - delta - lambda*data$INC
# Add also to data2
data2$Z <- data$Z
```

```
# Plot regression residuals
ggplot(data, aes(obs, Z)) +
  geom_point() +
  scale_x_yearqtr(format = fmt)
```



Next, we estimate to determine the number of ADF lags in our unit-root residual and conduct a DF cointegration test. The DF test for no-cointegration between consumption and income makes use of the t-ratio test statistic $DFNC = \frac{\hat{\phi}-1}{SE(\hat{\phi})}$. $\hat{\phi}$ is the estimated parameter from an AR(1) model $\hat{Z}_t = \phi \hat{Z_{t-1}} + \epsilon_t$. The null hypothesis is $H_0 : \phi = 1$ against $H_1 : |\phi| < 1$. Note that we have to use other critical values. For two I(1) series without a trend for which we test the nullhypothesis of non-cointegration the critical values are -3.9001 at 1%, -3.337 at 5% and -3.0462 at 10% (MacKinnon, 1991, p. 9).

```
# Estimate all models and pick specification with lowest SIC - INC
ts <- data$Z
sics <- rep(0, comb)

for (i in 1:comb){
  logl <- arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[i,], method ="CSS")$loglik
  k = sum(is.na(lag_struc[i,]))
  sics[i] = k*log(length(ts)) - 2*logl
}
min(sics)
```

```
## [1] 1450.084
```

```
arima(x=ts, order = c(max_lag -1,0,0), fixed = lag_struc[which.min(sics),], method ="CSS")
```

```
##
## Call:
## arima(x = ts, order = c(max_lag - 1, 0, 0), fixed = lag_struc[which.min(sics),
##     ], method = "CSS")
##
```

```
## Coefficients:
##          ar1  ar2  ar3      ar4  ar5  intercept
##       0.9816    0    0  -0.1813    0          0
## s.e.  0.0538    0    0   0.0534    0          0
##
## sigma^2 estimated as 165566:  part log likelihood = -720.47
```

```
# Perform ADF Test
ts <- ts(ts)
struc <- c(0)
for (i in 1:length(lag_struc[which.min(sics),]-1)){
  if (is.na(lag_struc[which.min(sics),i])){
    struc <- cbind(struc, i)
  }
}
struc <- as.numeric(struc[,-1])
# struc <- head(struc,-1)
if(is.na(lag_struc[which.min(sics),i])){
  int = TRUE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc))
} else{
  int = FALSE
  mod <- dynlm(diff(ts) ~ L(ts,1) + L(diff(ts),struc)-1)
}
phi = summary(mod)$coefficients[1,1]
se = summary(mod)$coefficients[1,2]
dfnc = (phi-1)/se
dfnc
```

```
## [1] -21.04336
```

```
if(int==TRUE){
  adf< (-3.0462)
}else{
  adf< (-3.9001)
}
```

```
## [1] FALSE
```

According to SIC the number of ADF lags in our unit-root residual test are 5, excluding the second and the third lag. Our co-integration test statistic is around -21.04. As the statistic is below the critical value of 1%, we can reject our null hypothesis which means that $\hat{Z}_t \sim I(0)$ and that we have co-integration between consumption and income.

6. Estimate an error correction model for consumption using the estimated residuals from the cointegration regression above. Use a general-to-specific modeling approach for the short-run dynamics. Report the estimated model. Report and interpret the short-run and long-run multipliers. Report and interpret the error correction coefficient.

For the general-to-specific modeling approach, we start estimating an error correction model with four legs for the first-difference of consumption and income and exclude insignificant lags step by step.

```
# Transform data into times series format
tsCONS <- ts(data$CONS, start= c(1988, 1), end = c(2012, 1), frequency = 4)
tsINC <- ts(data$INC, start= c(1988, 1), end = c(2012, 1), frequency = 4)
tsZ <- ts(data$Z, start= c(1988, 1), end = c(2012, 1), frequency = 4)
```

```
ecm <- dynlm(diff(tsCONS) ~ L(tsZ, 1) + L(diff(tsINC), 0:4) + L(diff(tsCONS), 1:4))
summary(ecm)
```

```
##
## Time series regression with "ts" data:
## Start = 1989(2), End = 2012(1)
##
## Call:
## dynlm(formula = diff(tsCONS) ~ L(tsZ, 1) + L(diff(tsINC), 0:4) +
##     L(diff(tsCONS), 1:4))
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -869.8 -132.8    7.5  136.6  523.2
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           171.45902   69.34030   2.473  0.01550 *
## L(tsZ, 1)              -0.17933    0.05508  -3.256  0.00165 **
## L(diff(tsINC), 0:4)0    0.22738    0.05374   4.231 6.08e-05 ***
## L(diff(tsINC), 0:4)1   -0.02516    0.06317  -0.398  0.69142
## L(diff(tsINC), 0:4)2   -0.09401    0.06342  -1.482  0.14212
## L(diff(tsINC), 0:4)3   -0.02143    0.06222  -0.344  0.73145
## L(diff(tsINC), 0:4)4   -0.05484    0.06022  -0.911  0.36523
## L(diff(tsCONS), 1:4)1  -0.13332    0.10497  -1.270  0.20772
## L(diff(tsCONS), 1:4)2   0.11022    0.09537   1.156  0.25120
## L(diff(tsCONS), 1:4)3   0.31521    0.09866   3.195  0.00199 **
## L(diff(tsCONS), 1:4)4   0.14211    0.10541   1.348  0.18138
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 282.5 on 81 degrees of freedom
## Multiple R-squared:  0.4637, Adjusted R-squared:  0.3975
## F-statistic: 7.004 on 10 and 81 DF,  p-value: 8.096e-08
```

```
ecm <- dynlm(diff(tsCONS) ~ L(tsZ, 1) + L(diff(tsINC), 0:3) + L(diff(tsCONS), 1:3))
summary(ecm)
```

```
##
## Time series regression with "ts" data:
## Start = 1989(1), End = 2012(1)
##
## Call:
## dynlm(formula = diff(tsCONS) ~ L(tsZ, 1) + L(diff(tsINC), 0:3) +
##     L(diff(tsCONS), 1:3))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -946.28 -143.09   22.47  158.42  528.04
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           175.64973   62.46386   2.812  0.00613 **
## L(tsZ, 1)              -0.16425    0.04873  -3.371  0.00114 **
```

```
## L(diff(tsINC), 0:3)0    0.21972    0.05332   4.121 8.81e-05 ***
## L(diff(tsINC), 0:3)1   -0.01115    0.06217  -0.179  0.85815
## L(diff(tsINC), 0:3)2   -0.10981    0.06013  -1.826  0.07137 .
## L(diff(tsINC), 0:3)3   -0.01314    0.05985  -0.220  0.82677
## L(diff(tsCONS), 1:3)1  -0.10145    0.09355  -1.084  0.28126
## L(diff(tsCONS), 1:3)2   0.12630    0.09366   1.348  0.18113
## L(diff(tsCONS), 1:3)3   0.31382    0.09834   3.191  0.00199 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 281.7 on 84 degrees of freedom
## Multiple R-squared:  0.4479, Adjusted R-squared:  0.3953
## F-statistic: 8.519 on 8 and 84 DF,  p-value: 2.029e-08
```

```
ecm <- dynlm(diff(tsCONS) ~ L(tsZ, 1) + L(diff(tsINC), 0:2) + L(diff(tsCONS), 1:3))
summary(ecm)
```

```
##
## Time series regression with "ts" data:
## Start = 1989(1), End = 2012(1)
##
## Call:
## dynlm(formula = diff(tsCONS) ~ L(tsZ, 1) + L(diff(tsINC), 0:2) +
##     L(diff(tsCONS), 1:3))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -935.71 -147.50   22.73  161.49  524.98
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)           171.307180  58.916284   2.908 0.004644 **
## L(tsZ, 1)              -0.159177   0.042665  -3.731 0.000344 ***
## L(diff(tsINC), 0:2)0    0.218235   0.052591   4.150 7.87e-05 ***
## L(diff(tsINC), 0:2)1   -0.009654   0.061455  -0.157 0.875539
## L(diff(tsINC), 0:2)2   -0.111867   0.059056  -1.894 0.061592 .
## L(diff(tsCONS), 1:3)1  -0.102625   0.092868  -1.105 0.272252
## L(diff(tsCONS), 1:3)2   0.126255   0.093136   1.356 0.178817
## L(diff(tsCONS), 1:3)3   0.310178   0.096379   3.218 0.001826 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 280.2 on 85 degrees of freedom
## Multiple R-squared:  0.4476, Adjusted R-squared:  0.4021
## F-statistic: 9.839 on 7 and 85 DF,  p-value: 6.264e-09
```

```
ecm <- dynlm(diff(tsCONS) ~ L(tsZ, 1) + L(diff(tsINC), 0:1) + L(diff(tsCONS), c(1,3)))
summary(ecm)
```

```
##
## Time series regression with "ts" data:
## Start = 1989(1), End = 2012(1)
##
## Call:
## dynlm(formula = diff(tsCONS) ~ L(tsZ, 1) + L(diff(tsINC), 0:1) +
```

```
##     L(diff(tsCONS), c(1, 3)))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -912.19 -165.38   40.21  176.03  573.55
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               178.23302   53.24288   3.348 0.001206 **
## L(tsZ, 1)                  -0.13735    0.03925  -3.500 0.000737 ***
## L(diff(tsINC), 0:1)0        0.20582    0.05299   3.884 0.000200 ***
## L(diff(tsINC), 0:1)1       -0.02418    0.05883  -0.411 0.682044
## L(diff(tsCONS), c(1, 3))1  -0.10017    0.09422  -1.063 0.290657
## L(diff(tsCONS), c(1, 3))3   0.28338    0.09609   2.949 0.004091 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 284.3 on 87 degrees of freedom
## Multiple R-squared:  0.4179, Adjusted R-squared:  0.3845
## F-statistic: 12.49 on 5 and 87 DF,  p-value: 3.816e-09
```

```
ecm <- dynlm(diff(tsCONS) ~ L(tsZ, 1) +L(diff(tsINC), 0) + L(diff(tsCONS), c(1,3)))
summary(ecm)
```

```
##
## Time series regression with "ts" data:
## Start = 1989(1), End = 2012(1)
##
## Call:
## dynlm(formula = diff(tsCONS) ~ L(tsZ, 1) + L(diff(tsINC), 0) +
##     L(diff(tsCONS), c(1, 3)))
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -924.2 -157.1   33.4  167.7  584.2
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               171.24166   50.21433   3.410 0.000982 ***
## L(tsZ, 1)                  -0.13003    0.03481  -3.736 0.000332 ***
## L(diff(tsINC), 0)           0.19511    0.04592   4.249 5.32e-05 ***
## L(diff(tsCONS), c(1, 3))1  -0.11087    0.09012  -1.230 0.221914
## L(diff(tsCONS), c(1, 3))3   0.29361    0.09237   3.179 0.002042 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 282.9 on 88 degrees of freedom
## Multiple R-squared:  0.4168, Adjusted R-squared:  0.3903
## F-statistic: 15.72 on 4 and 88 DF,  p-value: 9.621e-10
```

```
ecm <- dynlm(diff(tsCONS) ~ L(tsZ, 1) +L(diff(tsINC), 0) + L(diff(tsCONS), 3))
summary(ecm)
```

```
##
## Time series regression with "ts" data:
```

```
## Start = 1989(1), End = 2012(1)
##
## Call:
## dynlm(formula = diff(tsCONS) ~ L(tsZ, 1) + L(diff(tsINC), 0) +
##     L(diff(tsCONS), 3))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -963.35 -163.93   43.08  157.17  635.71
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        146.43379   46.11937   3.175 0.002058 **
## L(tsZ, 1)           -0.12151    0.03421  -3.552 0.000614 ***
## L(diff(tsINC), 0)    0.17725    0.04369   4.057 0.000106 ***
## L(diff(tsCONS), 3)   0.28253    0.09219   3.065 0.002885 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 283.7 on 89 degrees of freedom
## Multiple R-squared:  0.4067, Adjusted R-squared:  0.3867
## F-statistic: 20.34 on 3 and 89 DF,  p-value: 3.99e-10
```

Our final model is

$$\Delta Y_t = \gamma Z_{t-1} + \beta_0 \Delta X_t + \phi_3 \Delta Y_{t-3} + \epsilon_t$$

In the long-run equilibrium, we have $Y_t = \bar{Y}, X_t = \bar{X}_t$, $\Delta\bar{Y} = 0$ and $\Delta\bar{X} = 0$. Hence, our equation becomes

$$0 = \gamma Z_{t-1} \Rightarrow 0 = \bar{Y} - \delta - \lambda\bar{X}$$

This can be rearranged to

$$\bar{Y} = \delta + \lambda\bar{X}$$

which is why our long-run multiplier is $\lambda = 0.6651404$ (as we derived in IV.5).

Our short-run multiplier measures the expected impact on ΔY_t of a unit increase in ΔX_t. Therefore, our short-run multiplier equals 0.17725.

The error correction coefficient corresponds to γ in our model and equals -0.12151. It determines the adjustment speed to the long-run equilibrium which is implied by X_{t-1}. As it is below zero, we can conclude that correction to deviations from long-run equilibrium exists. However, as γ is between zero and minus one, we only have partial error correction.

7. How strong is the correction to equilibrium? Is there over-shooting? Do you find evidence of Granger causality? Justify your answer.

We have no over-shooting as the error correction coefficient is between zero and minus one. For overshooting, we would need $-2 < \gamma < -1$. However, with $\gamma = -0.12151$, the error correction coefficient is quite close to zero which means that the adjustment is rather slow.

We have Granger Causality when a time-series $\{X_t\}$ causes a time-series $\{Y_t\}$ which means that past values of $\{X_t\}$ provide statistically significant information on future values of $\{Y_t\}$. As our coefficient for ΔX_t is significant there might by Granger Causality. However, we cannot conclude definitely whether the significance stems from X_t or X_{t-1} as $\Delta X_t = X_{t-1} - X_t$.

8. At the peak of the recession, during the 2nd quarter of 2009, you are asked to forecast the value of consumption for the 3rd quarter of 2009. Do you expect aggregate consumption to raise or fall? Report the predicted change in the value of consumption. How much of this change in consumption is due to

the correction mechanism alone? Report your point forecast for consumption for the 3rd quarter of 2009.

We expect aggregate consumption to raise as the coefficients for ΔY_{t-3} and ΔX_t are higher stronger than the error correction coefficient. However, it also depends on the lags. If they are positive then the outcome should corresponds to our expectations. If they are negative, then aggregate consumption might also fall.

```
i = 86
pred_delta <- ecm$coefficients[1] + ecm$coefficients[2]*data2$Z[i-1] +  ecm$coefficients[3]*data2$INC[i]
as.numeric(pred_delta)
```

```
## [1] -246.1442
```

The predicted change is -246.1442.

```
i = 86
pred_delta_lr <- ecm$coefficients[2]*data2$Z[i-1]
as.numeric(pred_delta_lr)
```

```
## [1] -191.3947
```

The change in consumption due to the correction mechanism alone is around -191.3947.

```
i = 86
pred_delta_sr <- ecm$coefficients[1] + ecm$coefficients[3]*data2$INC[i] + ecm$coefficients[4]*data2$CONS
as.numeric(pred_delta_sr)
```

```
## [1] -54.7495
```

```
j=86
pred <- data$CONS[j] + pred_delta
as.numeric(pred)
```

```
## [1] 86502.66
```

The point forecast for consumption for the 3rd quarter of 2009 is 86502.66.

References

Fama, E. F. (1995). Random walks in stock market prices. Financial analysts journal, 51(1), 75-80.

MacKinnon, J. G. (1991). Critical values for cointegration tests. In Eds., Long-Run Economic Relationship: Readings in Cointegration.