Econometrics II - Problem 6

William Radaic Peron

September 3, 2020

Part 2: Unit roots

As has been discussed during the lecture, when the population model is a *random walk*:

$$Y_t = 1 * Y_{t-1} + \varepsilon_t, \quad \varepsilon \sim wn(0, \sigma^2),$$

it happens that the series *is not ergodic* (nor is it stationary). Therefore, the usual asymptotic properties *do not hold*, as all innovations have *permanent effects*.

$$Y_t = c + \delta t + \sum_{i=1}^{t} \varepsilon_i$$

The random walk, defined above, is an example of the more general class of *unit root processes*:

$$Y_t = c + \delta t + u_t,$$

u_t has an ARMA(p,q) representation:

$$\Phi_p(L) u_t = \Theta_q(L) \varepsilon_t, \quad \varepsilon \sim wn(0, \sigma^2)$$

Suppose that one of the roots of $\Theta_p(L)$ is equal to 1:

$$\Theta_p(L) = (1 - [1]L)(1 - \lambda_2 L)...(1 - \lambda_p L)$$

$$(1 - L) u_t = (1 - \lambda_2 L)^{-1} ... (1 - \lambda_p L)^{-1} \Theta_q(L) \varepsilon_t =: \Psi(L) \varepsilon_t$$

We can now rewrite this as:

$$(1 - L) Y_t = (1 - L)c + (1 - L)\delta t + (1 - L) u_t$$

$$(1 - L) Y_t = \delta + \Psi(L) \varepsilon_t$$

Defining ΔY_t, we have:

$$\Delta Y_t := (1 - L) Y_t = Y_t - Y_{t-1} = \delta + \Psi(L) \varepsilon_t$$

With this concept, we can define ARIMA(p,d,q) processes:

$$\Phi_p(L)(1 - L)^d Y_t = c + \Theta_q(L) \varepsilon_t, \quad \varepsilon_t \sim wn(0, \sigma^2)$$

Hypothesis testing

It turns out that testing for unit root presence presents some challenges. Under the null ($a_1 = 1$), its distribution *is not standard* and does not present the usual asymptotic properties. We can circumvent this issue with the use of numeric methods, such as the *Monte Carlo simulation*. It will now be employed, following Dickey and Fuller (1979).

First, some notation:

We begin with a simple model:

$$Y_t = a_1 y_{t-1} + \varepsilon_t$$

$$\Delta Y_t = \gamma y_{t-1} + \varepsilon_t, \quad \gamma := a_1 - 1$$

Dickey and Fuller constructed the following regression equations:

$$\Delta y_t = \gamma y_{t-1} + \varepsilon_t \tag{1}$$

$$\Delta y_t = a_0 + \gamma y_{t-1} + \varepsilon_t \tag{2}$$

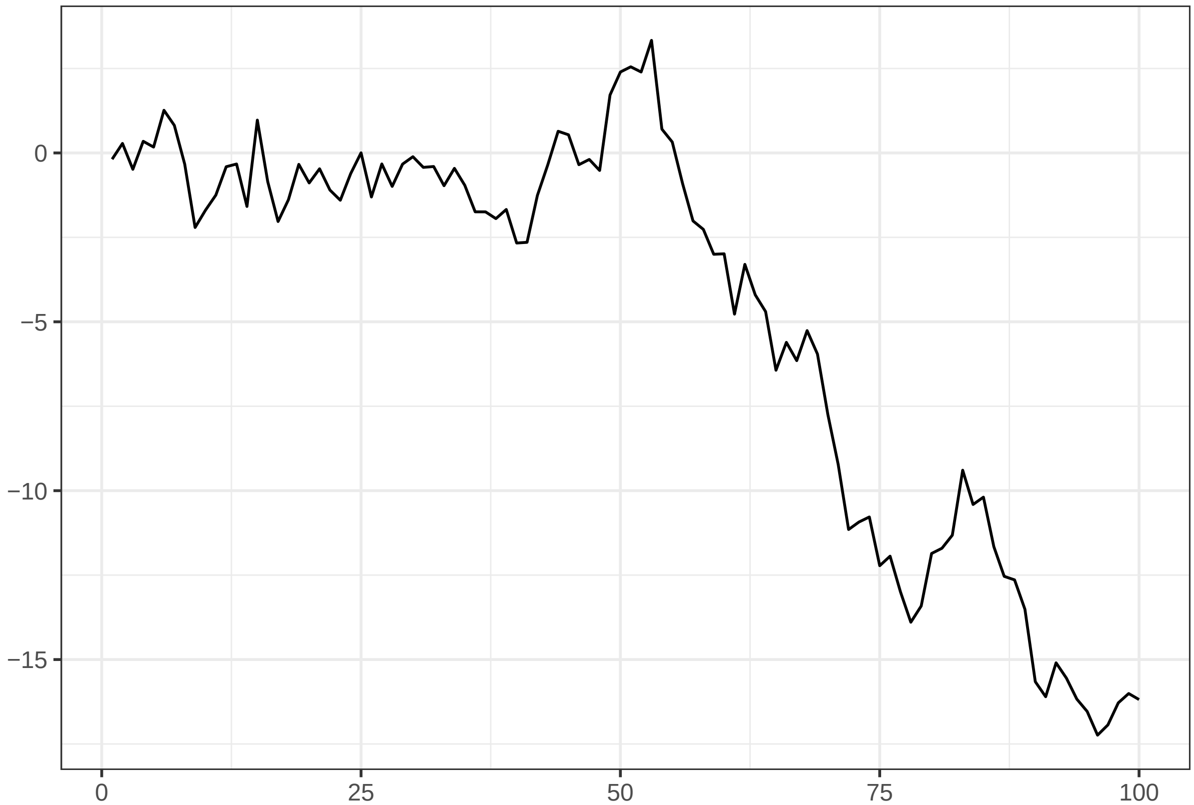
$$\Delta y_t = a_0 + \gamma y_{t-1} + a_2 t + \varepsilon_t \tag{3}$$

(1) has no intercept and represents a simple random walk. (2) includes an intercept. (3) also adds a deterministic time trend.

Note that the critical values of the t-statistics are *different* between these regressions. This will now be shown with our Monte Carlo simulation.

For equation (1):

```
set.seed(76345)

# length of ts
T <- 100

# Loops
S <- 10000

e <- rnorm(T, 0, 1)

# As y_{t+1} = y_t + \varepsilon_t, we can write this as an MA(\infty) model. y_t = \sum \varepsilon_i.

y <- cumsum(e)

y <- vector()

y[1] = e[1]

for (i in (2:T)) {

    y[i] <- y[(i-1)] + e[i]

}

y <- as.ts(y)

autoplot(y) + theme_bw()
```

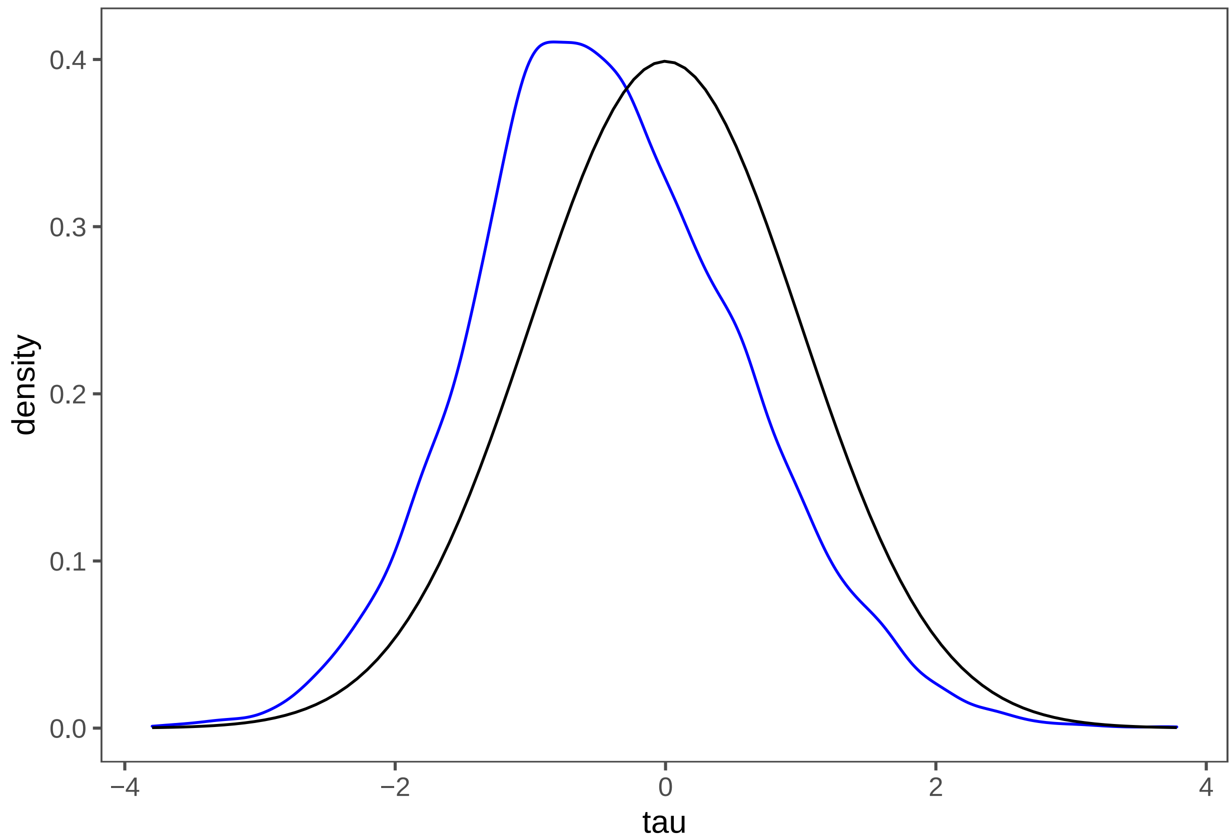



```
# Taking the first difference

y_diff <- diff(y)

# Regression of I(1) model

reg <- dynlm(y_diff ~ 0 + L(y,1)) # no lag (x1 = 0)

reg <-summary(reg)

reg$coefficients[1,3] # t value
```

```
## [1] 1.110447
```

```
# Loop

tau <- vector()

for (i in 1:S) {

  e <- rnorm(T,0,1)
  y <- cumsum(e)
  y <- as.ts(y)

  y_diff <- diff(y)

  reg <- summary(dynlm(y_diff ~ 0 + L(y, 1)))
```

```r
  tau[i] <- reg$coefficients[1,3]

}

tau.df <- data.frame(tau)

ggplot(data = tau.df, aes(x = tau)) + geom_density(color = "blue") + stat_function(fun = dnorm, n = 101
```




```r
jarque.bera.test(tau) # We reject H0 at the 1% significance level.
```

```
##
##  Jarque Bera Test
##
## data:  tau
## X-squared = 101.33, df = 2, p-value < 2.2e-16
```

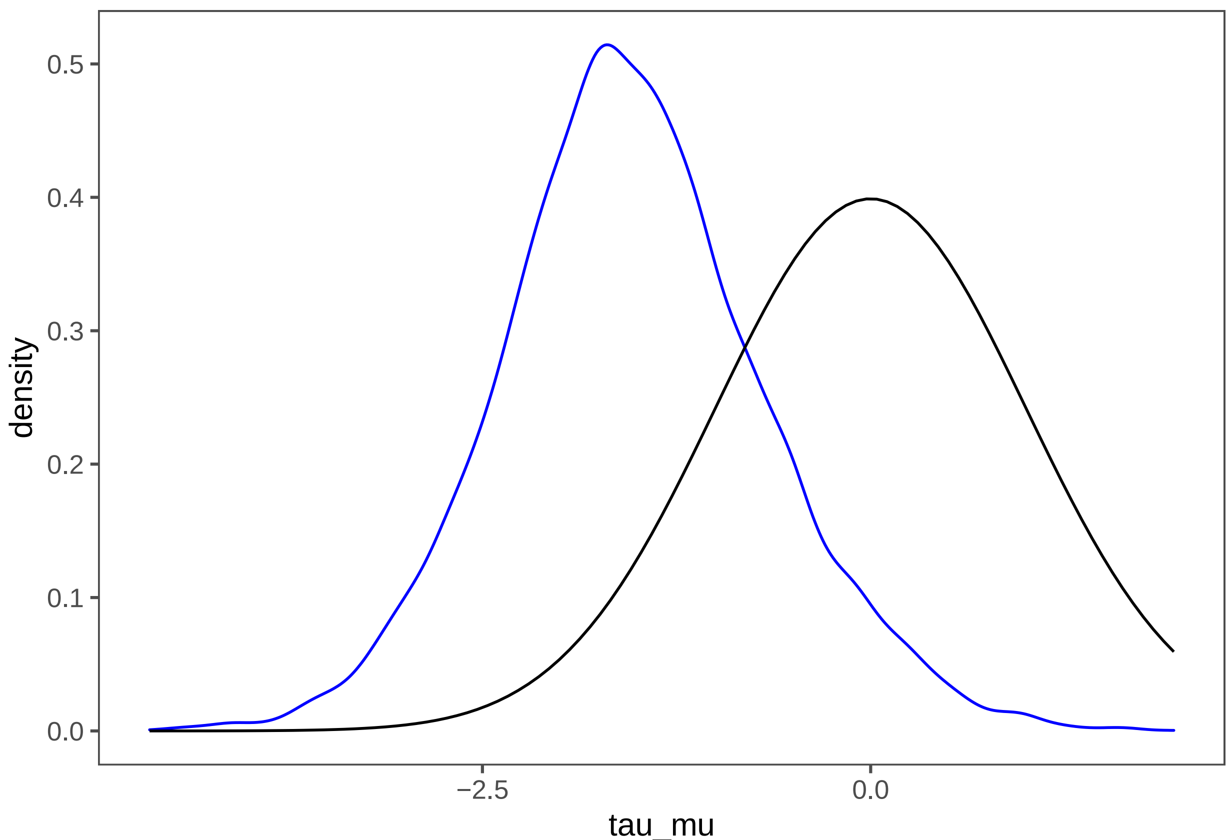
```r
tau.ci <- quantile(tau, c(0.01, 0.05, 0.1))

tau.ci
```

```
##        1%        5%       10%
## -2.591810 -1.959085 -1.634022
```

For equation (2) – with an intercept:

```r
# Loop

tau_mu <- vector()
```

```r
for (i in 1:S) {

  e <- rnorm(T,0,1)
  y <- cumsum(e)
  y <- as.ts(y)

  y_diff <- diff(y)

  reg <- summary(dynlm(y_diff ~ 1 + L(y, 1)))

  tau_mu[i] <- reg$coefficients[2,3]

}

tau_mu.df <- data.frame(tau_mu)

ggplot(data = tau_mu.df, aes(x = tau_mu)) + geom_density(color = "blue") + stat_function(fun = dnorm, n
```




```r
jarque.bera.test(tau_mu) # We reject H0 at the 1% significance level.
```

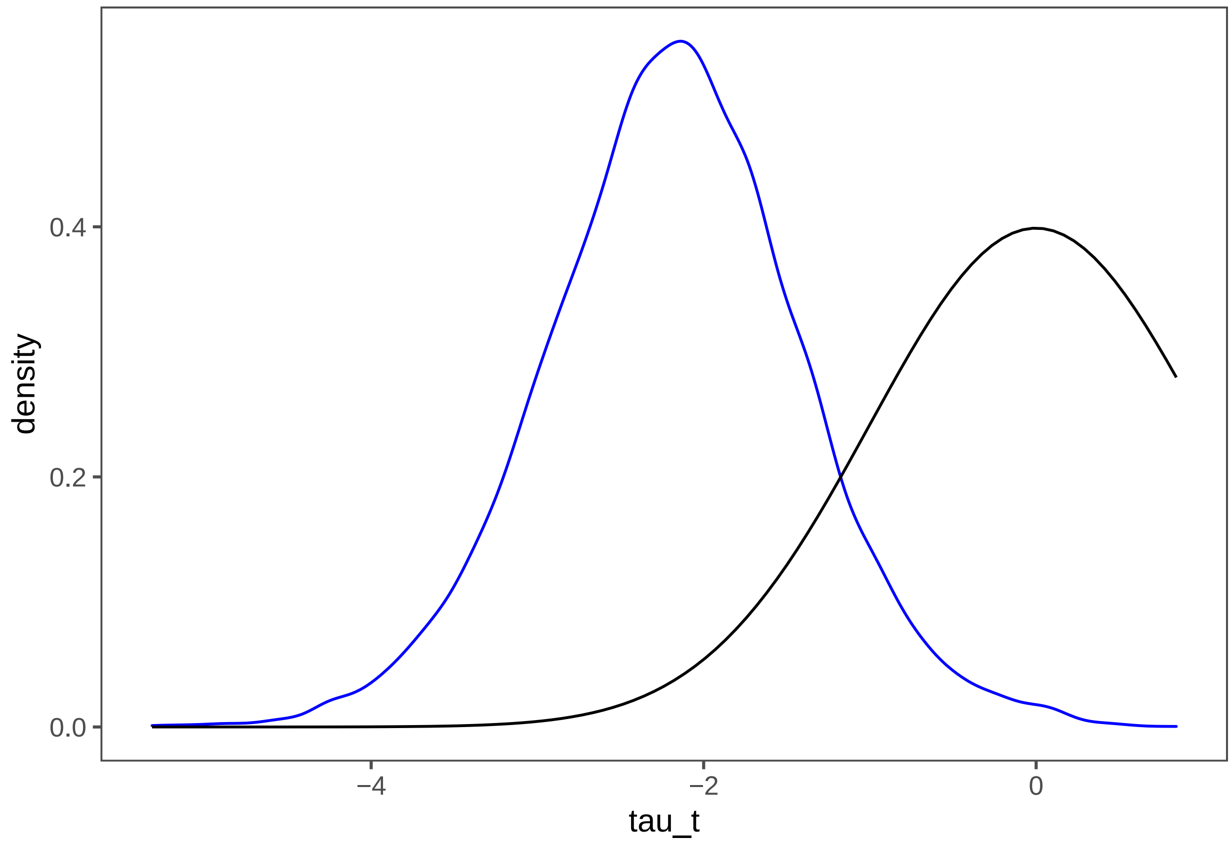
```
##
##  Jarque Bera Test
##
## data:  tau_mu
## X-squared = 96.894, df = 2, p-value < 2.2e-16
```

```r
tau_mu.ci <- quantile(tau_mu, c(0.01, 0.05, 0.1))

tau_mu.ci
```

```
##        1%        5%       10%
## -3.494281 -2.895757 -2.577903
```

And finally, for equation (3) – with an intercept and a deterministic time trend:

```r
# Loop

tau_t <- vector()
time <- c(1:T)

for (i in 1:S) {

  e <- rnorm(T,0,1)
  y <- cumsum(e)
  y <- as.ts(y)

  y_diff <- diff(y)

  reg <- summary(dynlm(y_diff ~ 1 + L(y, 1) + time[-1])) # removed 1 dimension for no. of obs.

  tau_t[i] <- reg$coefficients[2,3]

}

tau_t.df <- data.frame(tau_t)

ggplot(data = tau_t.df, aes(x = tau_t)) + geom_density(color = "blue") + stat_function(fun = dnorm, n =
```




```
jarque.bera.test(tau_t) # We reject H0 at the 1% significance level.
```

```
##
##  Jarque Bera Test
##
## data:  tau_t
## X-squared = 58.223, df = 2, p-value = 2.275e-13
```

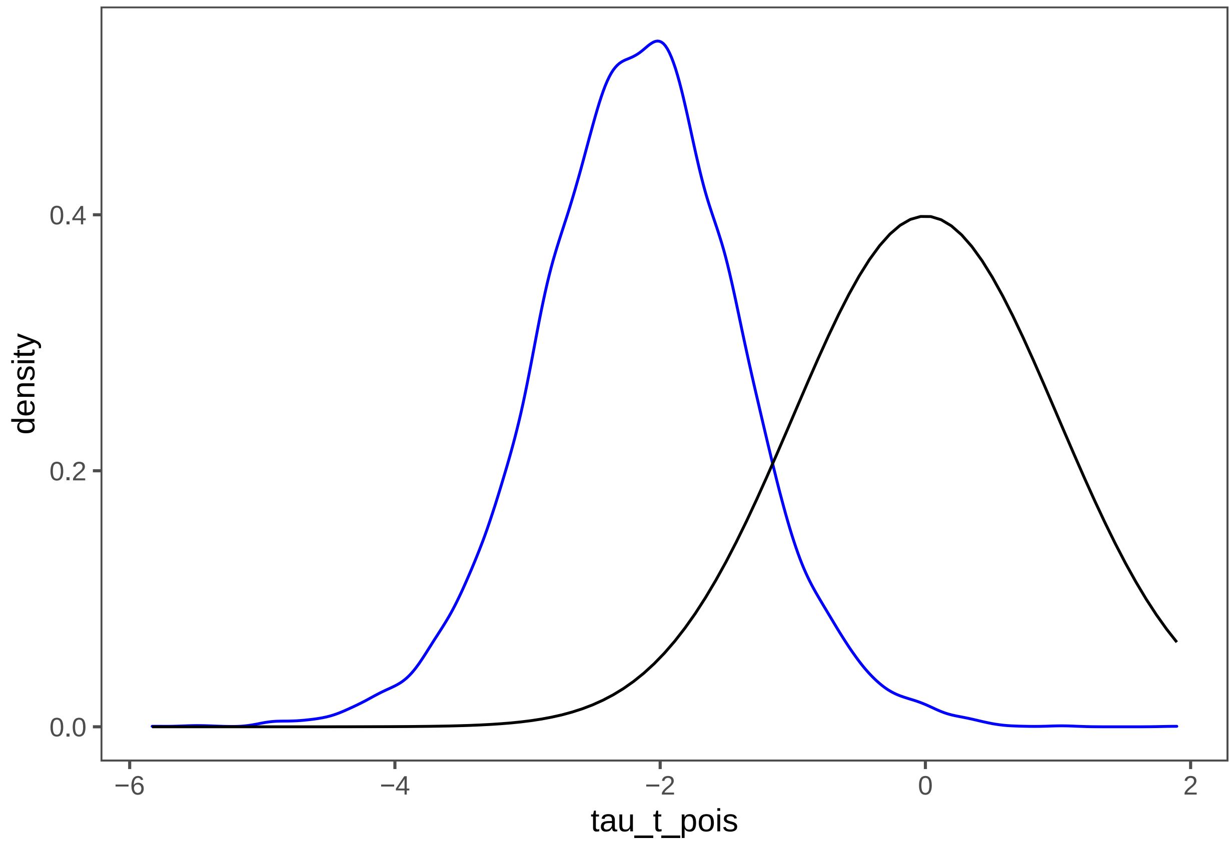
```
tau_t.ci <- quantile(tau_t, c(0.01, 0.05, 0.1))

tau_t.ci
```

```
##        1%        5%       10%
## -4.065067 -3.462447 -3.158227
```

Let's now change the distributions of the errors for equation (3):

```
# Loop

tau_t_pois <- vector()
time <- c(1:T)

for (i in 1:S) {

  e <- rpois(T,1)
  y <- cumsum(e)
  y <- as.ts(y)
```

```r
  y_diff <- diff(y)

  reg <- summary(dynlm(y_diff ~ 1 + L(y, 1) + time[-1])) # removed 1 dimension for no. of obs.

  tau_t_pois[i] <- reg$coefficients[2,3]

}

tau_t_pois.df <- data.frame(tau_t_pois)

ggplot(data = tau_t_pois.df, aes(x = tau_t_pois)) + geom_density(color = "blue") + stat_function(fun = d
```




```r
jarque.bera.test(tau_t_pois) # We reject H0 at the 1% significance level.
```

```
##
##  Jarque Bera Test
##
## data:  tau_t_pois
## X-squared = 120.61, df = 2, p-value < 2.2e-16
```

```r
tau_t_pois.ci <- quantile(tau_t_pois, c(0.01, 0.05, 0.1))

tau_t_pois.ci
```

```
##        1%        5%       10%
## -4.085753 -3.436365 -3.127525
```

Part 3: Applying the Dickey-Fuller test for GDP

Loading the data from the previous problem:

```r
pib <- get_sidra(6612, variable = 9318, category = 90707, period = "all")
```

```
## Considering all categories once 'classific' was set to 'all' (default)
```

```r
pib_limpo <- pib[(pib$`Setores e subsetores (Código)` == 90707),]

pib <- pib_limpo

pib_limpo2 <- pib[,c(5,13)]

pib <- pib_limpo2

names(pib)[1] <- "t"
names(pib)[2] <- "v"

pib$t <- as.numeric(pib$t)

names(pib)
```

```
## [1] "t" "v"
```

```r
head(pib)
```

```
##          t        v
## 18  199601 170920.0
## 40  199602 176708.8
## 62  199603 189844.3
## 84  199604 184112.9
## 106 199701 176732.2
## 128 199702 185109.5
```

```r
tail(pib)
```

```
##           t        v
## 2042 201901 292647.6
## 2064 201902 297748.9
## 2086 201903 305150.9
## 2108 201904 302108.7
## 2130 202001 291912.5
## 2152 202002 263699.7
```

```r
pib <- ts(pib$v)
```

```r
# Choosing the correct model with auto.arima

aa_pib <- auto.arima(pib, stepwise = F)

summary(aa_pib)
```

```
## Series: pib
## ARIMA(2,1,2) with drift
##
## Coefficients:
##           ar1      ar2     ma1    ma2     drift
```

```
##       -0.0092  -0.9764  0.2721  0.956  864.9855
## s.e.   0.0233   0.0171  0.0455  0.117  537.6284
##
## sigma^2 estimated as 23455415:  log likelihood=-961.03
## AIC=1934.06   AICc=1934.99   BIC=1949.51
##
## Training set error measures:
##                     ME    RMSE      MAE        MPE     MAPE      MASE       ACF1
## Training set 64.56963 4692.48 3219.806 0.05416274 1.322955 0.5203698 0.01746157
```

auto.arima yields an ARIMA(2,1,2) model with a drift parameter:

$$\Delta y_t = a_0 + a_1 \Delta y_{t-1} + a_2 \Delta y_{t-2} + a_3 \varepsilon_{t-1} + a_4 \varepsilon_{t-2} + \varepsilon_t$$

Let's decompose this process. First, we'll perform the Dickey-Fuller test on the GDP ts.

```
adf.test(pib)
```

```
## Warning in adf.test(pib): p-value greater than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  pib
## Dickey-Fuller = 0.12028, Lag order = 4, p-value = 0.99
## alternative hypothesis: stationary
```

As we have not been able to reject H_0, it follows that the ts includes (at least one) unit root. Now, let's find the optimal model for the regression.

```
max_p <- 5

max_q <- 5

max_d <- 2

models1 <- vector("list", (max_p + 1) * (max_q + 1))

models2 <- vector("list", (max_p + 1) * (max_q + 1))

# Updating the model

fit1 <- vector("list", (max_p + 1) * (max_q + 1))

fit2 <- vector("list", (max_p + 1) * (max_q + 1))

model_info1 <- data.frame(matrix(NA, nrow = ((max_p + 1) * (max_q + 1)), ncol = 3))




# Updating the model

for (u in 0:max_q) {

for (j in 0:max_p) {
```

```r
 fit1[[(((max_p+1)*j)+u + 1)]] <- Arima(pib, order = c(j,1,u))

 model_info1[(((max_p+1)*j)+u + 1), 1:2] <- c(fit1[[(((max_p+1)*j)+u + 1)]]$aic, fit1[[(((max_p+1)*j)+u




}
}

names(model_info1) <- c("AIC", "BIC", "void")



which.min(model_info1$AIC)
```

```
## [1] 23
```

```r
which.min(model_info1$BIC)
```

```
## [1] 15
```

```r
fit1[[which.min(model_info1$AIC)]]
```

```
## Series: pib
## ARIMA(3,1,4)
##
## Coefficients:
##           ar1      ar2      ar3     ma1     ma2     ma3     ma4
##       -0.9688  -0.9834  -0.9685  1.4659  1.3755  1.2705  0.5008
## s.e.   0.0305   0.0210   0.0270  0.1344  0.1682  0.1821  0.1484
##
## sigma^2 estimated as 21987482:  log likelihood=-956.26
## AIC=1928.53   AICc=1930.17   BIC=1949.13
```

```r
fit1[[which.min(model_info1$BIC)]]
```

```
## Series: pib
## ARIMA(2,1,2)
##
## Coefficients:
##           ar1      ar2     ma1     ma2
##       -0.0084  -0.9757  0.2769  0.9673
## s.e.   0.0236   0.0175  0.0408  0.1130
##
## sigma^2 estimated as 23692560:  log likelihood=-962.3
## AIC=1934.6   AICc=1935.26   BIC=1947.47
```

```r
# For I(2)

model_info2 <- data.frame(matrix(NA, nrow = max_d*((max_p + 1) * (max_q + 1)), ncol = 3))

for (u in 0:max_q) {

for (j in 0:max_p) {
```

```r
 fit2[[(((max_p+1)*j)+u + 1)]] <- Arima(pib, order = c(j,2,u))

 model_info2[(((max_p+1)*j)+u + 1), 1:2] <- c(fit2[[(((max_p+1)*j)+u + 1)]]$aic, fit2[[(((max_p+1)*j)+u

}
}

names(model_info2) <- c("AIC", "BIC", "void")

which.min(model_info2$AIC)
```

```
## [1] 24
```

```r
which.min(model_info2$BIC)
```

```
## [1] 16
```

```r
fit2[[which.min(model_info2$AIC)]]
```

```
## Series: pib
## ARIMA(3,2,5)
##
## Coefficients:
##           ar1      ar2      ar3     ma1     ma2      ma3      ma4      ma5
##       -0.9700  -0.9835  -0.9674  0.4748  0.0162  -0.0121  -0.6534  -0.3766
## s.e.   0.0315   0.0218   0.0277  0.1511  0.1211   0.1508   0.1090   0.1793
##
## sigma^2 estimated as 22148502:  log likelihood=-946.73
## AIC=1911.47   AICc=1913.56   BIC=1934.55
```

```r
fit2[[which.min(model_info2$BIC)]]
```

```
## Series: pib
## ARIMA(2,2,3)
##
## Coefficients:
##           ar1      ar2      ma1     ma2      ma3
##       -0.0054  -0.9795  -0.6754  0.6720  -0.7903
## s.e.   0.0248   0.0162   0.1121  0.0786   0.1482
##
## sigma^2 estimated as 23858414:  log likelihood=-951.82
## AIC=1915.63   AICc=1916.58   BIC=1931.02
```