ECON7350: Applied Econometrics for Macroeconomics and Finance

Tutorial 5: Trends and Cycles

At the end of this tutorial you should be able to:

- construct an adequate set of ADF specifications for unit root testing;
- carry out ADF tests for a unit root and interpret the results;
- construct an adequate set of general ARIMA(p, d, q) models.

Problems with Solutions

The specification for a general ARIMA(p, d, q) model is

$$\Delta^d y_t = \delta_t + \sum_{j=1}^{p} a_j \Delta^d y_{t-j} + \sum_{j=1}^{q} b_j \epsilon_t + \epsilon_t,$$

where δ_t is a general *deterministic term*.

- If the process has no deterministic terms, then $\delta_t = 0$.
- If the process includes a constant only, then $\delta_t = a_0$.
- If there is a constant and a trend, then $\delta_t = a_0 + \delta t$.

The file `usdata.csv` contains 209 observations on:

- $y_t \equiv$ log real per capita GDP (`GDP`); and
- $r_t \equiv$ the overnight Federal Funds Rate for the US (`FFR`).

1. For y_t:
   (a) Plot the observed time series and comment on potential trends.

Solution For this tutorial, we load the following useful packages.

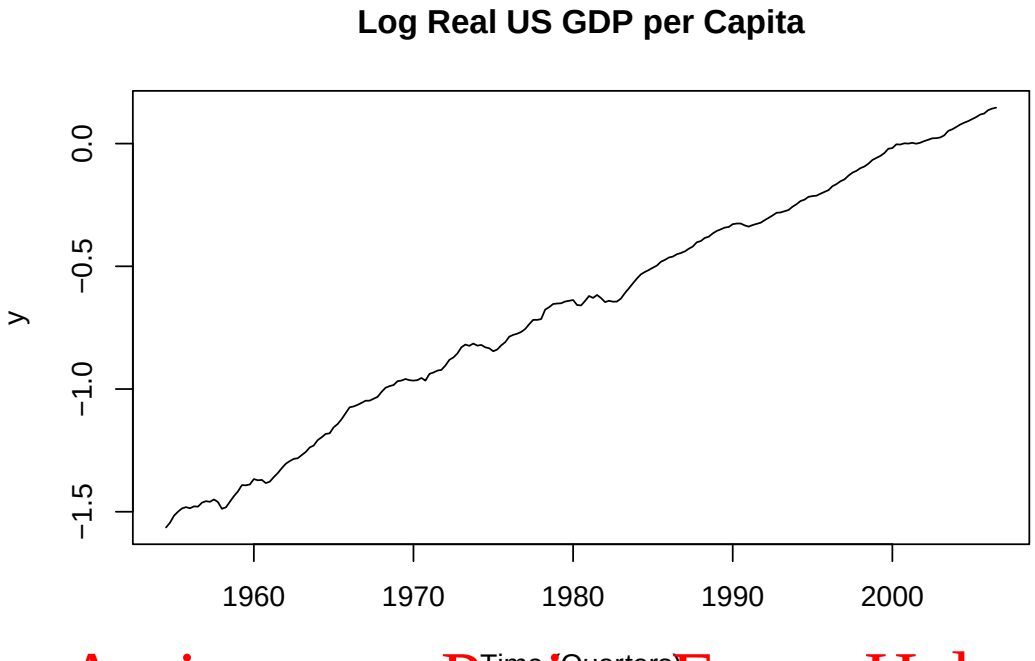
```
library(forecast)
library(dplyr)
library(zoo)
library(aTSA)
```

Next, load the data and plot log real US GDP per capita. Clearly, per capita GDP levels are increasing in the US over the sample period, so we suspect a trend exists in the data generating process.


```
mydata <- read.delim("usdata.csv", header = TRUE,  sep = ",")

dates <- as.yearqtr(mydata$obs)
y <- mydata$GDP
r <- mydata$FFR

plot(dates, y, type = "l", xlab = "Time (Quarters)",
     main = "Log Real US GDP per Capita")
```



(b) Construct an adequate set of ADF regression models.

Solution ADF regression are specified by lag length p along with the inclusion / exclusion of a constant and trend. Note that a trend is only included if a constant is included (i.e., we do not consider specifications with a trend, but no constant). With that in mind, we implement the familiar approach.

```
TT <- length(y)
ADF_est <- list()
ic <- matrix( nrow = 30, ncol = 5 )
colnames(ic) <- c("cons", "trend", "p", "aic", "bic")
i <- 0
for (const in 0:1)
{
  for (p in 0:9)
  {
    i <- i + 1
    ADF_est[[i]] <- Arima(diff(y), xreg = y[-TT],
                          order = c(p, 0, 0),
                          include.mean = as.logical(const),
                          include.drift = F)
```

```
    ic[i,] <- c(const, 0, p, ADF_est[[i]]$aic,
                             ADF_est[[i]]$bic)
  }

  if (const)
  {
    # only add a specification with trend if there is a
    # constant (i.e., exclude no constant with trend)
    for (p in 0:9)
    {
      i <- i + 1
      ADF_est[[i]] <- Arima(diff(y), xreg = y[-TT],
                            order = c(p, 0, 0),
                            include.mean = as.logical(const),
                            include.drift = T)
      ic[i,] <- c(const, 1, p, ADF_est[[i]]$aic,
                               ADF_est[[i]]$bic)
    }
  }
}

ic_aic <- ic[order(ic[,4]),][1:10,]
ic_bic <- ic[order(ic[,5]),][1:10,]

print(ic_aic)
```



```
##       cons trend p       aic       bic
##  [1,]    1     1 2 -1384.766 -1364.741
##  [2,]    1     1 1 -1383.057 -1366.369
##  [3,]    1     1 3 -1382.777 -1359.414
##  [4,]    1     1 4 -1380.779 -1354.079
##  [5,]    1     1 5 -1380.217 -1350.179
##  [6,]    1     0 1 -1378.683 -1365.333
##  [7,]    1     0 2 -1378.681 -1361.993
##  [8,]    1     1 6 -1378.575 -1345.199
##  [9,]    1     0 3 -1377.156 -1357.130
## [10,]    1     1 7 -1376.638 -1339.925
```

```
print(ic_bic)
```

```
##       cons trend p       aic       bic
##  [1,]    1     1 1 -1383.057 -1366.369
##  [2,]    1     0 1 -1378.683 -1365.333
##  [3,]    1     1 2 -1384.766 -1364.741
##  [4,]    1     0 2 -1378.681 -1361.993
##  [5,]    1     1 3 -1382.777 -1359.414
```

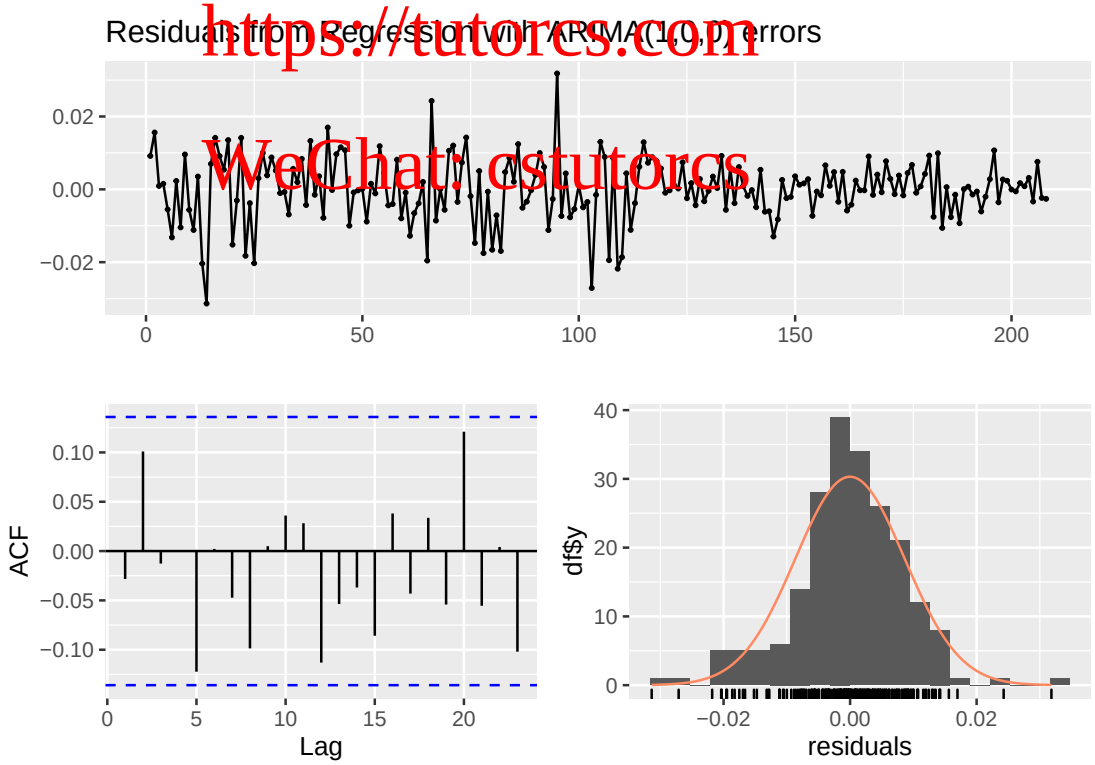
```
##  [6,]    1      0 3 -1377.156 -1357.130
##  [7,]    0      0 2 -1367.494 -1354.144
##  [8,]    1      1 4 -1380.779 -1354.079
##  [9,]    0      0 1 -1362.530 -1352.518
## [10,]    1      0 4 -1375.519 -1352.156
```

The AIC prefers specifications with both a constant and trend as well as lag lengths in the range $p = 1, \ldots, 5$. The BIC ranking includes specifications with both a constant and trend as well as lags $p = 1, \ldots, 3$. However, it also includes specifications with a constant only and lags $p = 1, \ldots, 2$. Putting this information together, we select the top five specifications preferred by the BIC.

```
adq_set <- as.matrix(arrange(as.data.frame(ic_bic[1:5,]),
                    const, trend, p))
adq_idx <- match(data.frame(t(adq_set[, 1:3])),
                 data.frame(t(ic[, 1:3])))
```

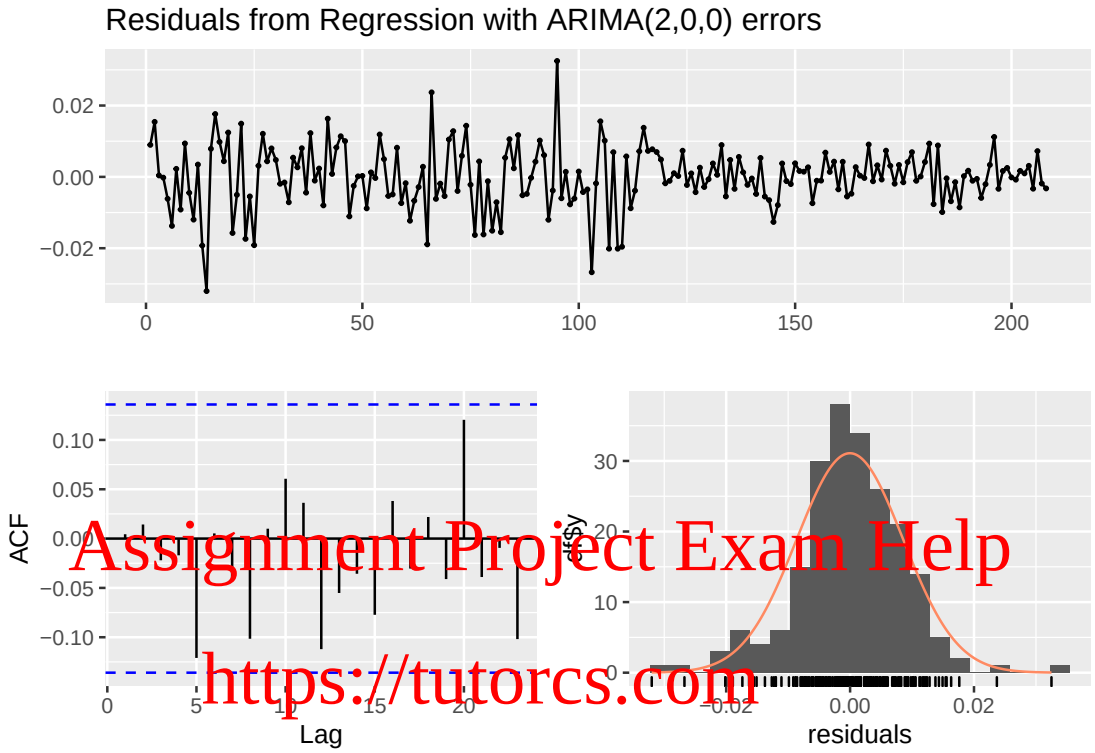
Finally, we check the residuals for all specifications in the adequate set.

```
for (i in 1:length(adq_idx))
{
  checkresiduals(ADF_est[[adq_idx[i]]])
}
```

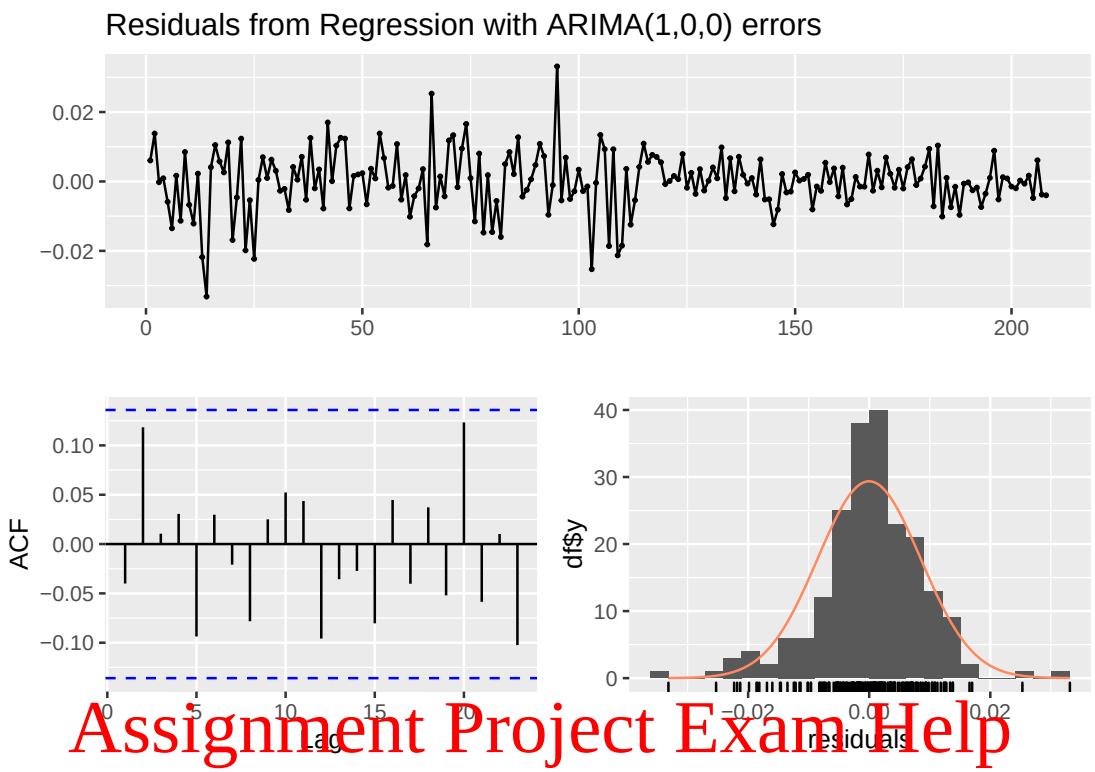




```
##
##  Ljung-Box test
##
```

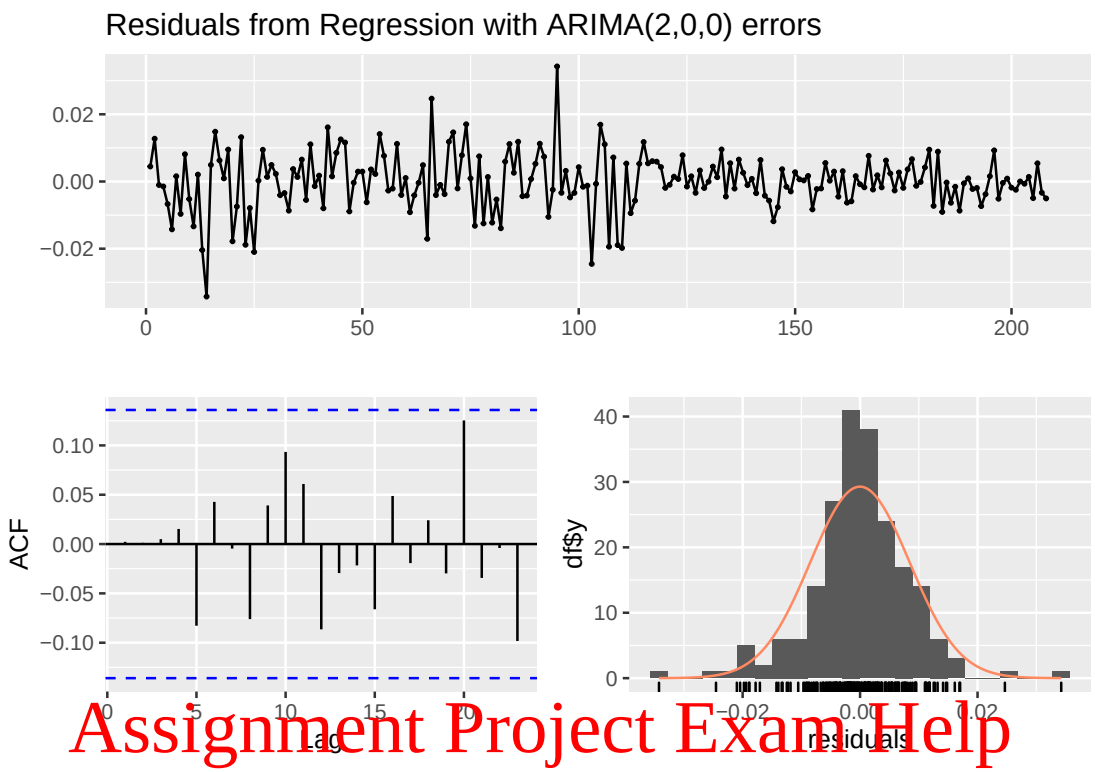
```
## data:  Residuals from Regression with ARIMA(1,0,0) errors
## Q* = 8.4811, df = 7, p-value = 0.2921
##
## Model df: 3.   Total lags used: 10
```



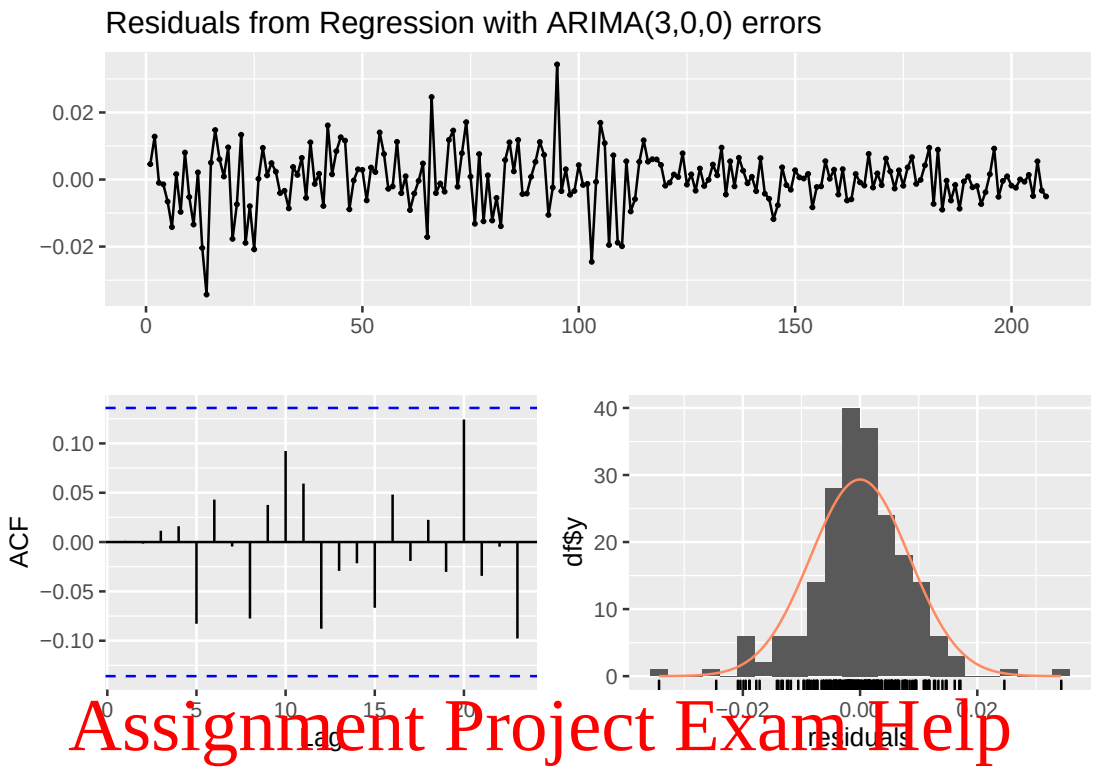

```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(2,0,0) errors
## Q* = 6.822, df = 6, p-value = 0.3376
##
## Model df: 4.   Total lags used: 10
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,0,0) errors
## Q* = 7.7825, df = 6, p-value = 0.2545
##
## Model df: 4.   Total lags used: 10
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(2,0,0) errors
## Q* = 5.4569, df = 5, p-value = 0.3627
##
## Model df: 5.   Total lags used: 10
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(3,0,0) errors
## Q* = 5.466, df = 4, p-value = 0.2427
##
## Model df: 6.   Total lags used: 10
```

As no obvious problems jump out from the residuals analysis, we proceed with the adequate set constructed above.

(c) Implement the ADF test for a stochastic trend and draw inference regarding the integration properties of y_t.

Solution Use the `adf.test` function from the `aTSA` package.

```
adf.test(y)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
```

```
## [1,]   0 -10.33    0.01
## [2,]   1  -5.31    0.01
## [3,]   2  -4.05    0.01
## [4,]   3  -3.79    0.01
## [5,]   4  -3.54    0.01
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0 -1.347   0.575
## [2,]   1 -0.931   0.721
## [3,]   2 -0.637   0.824
## [4,]   3 -0.648   0.820
## [5,]   4 -0.659   0.816
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -1.99   0.577
## [2,]   1 -2.45   0.385
## [3,]   2 -2.54   0.347
## [4,]   3 -2.44   0.390
## [5,]   4 -2.36   0.424
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```


Since only "Type 2" and "Type 3" specifications are in our adequate set, we ignore the output related to "Type 1" specifications.Consequently, for all specifications in our adequate set, the null of unit root cannot be rejected. We conclude that the process is *not empirically distinguishable* from an integrated process with a drift (and possibly linear growth).

(d) Repeat parts (a)-(c) for the differenced series Δy_t.

Solution The procedure is nearly identical to that used in constructing an adequate set for y, but replacing it with Δy instead. In R, this is done by simply replacing `y` with `diff(y)`.

```
TT <- length(diff(y))
ADF_est_diff <- list()
ic_diff <- matrix( nrow = 30, ncol = 5 )
colnames(ic_diff) <- c("cons", "trend", "p", "aic", "bic")
i <- 0
for (const in 0:1)
{
  for (p in 0:9)
  {
```

```
    i <- i + 1
    ADF_est_diff[[i]] <- Arima(diff(diff(y)),
                               xreg = diff(y)[-TT],
                               order = c(p, 0, 0),
                               include.mean = as.logical(const),
                               include.drift = F)
    ic_diff[i,] <- c(const, 0, p, ADF_est_diff[[i]]$aic,
                                  ADF_est_diff[[i]]$bic)
  }

  if (const)
  {
    # only add a specification with trend if there is a
    # constant (i.e., exclude no constant with trend)
    for (p in 0:9)
    {
      i <- i + 1
      ADF_est_diff[[i]] <- Arima(diff(diff(y)),
                                 xreg = diff(y)[-TT],
                                 order = c(p, 0, 0),
                                 include.mean = as.logical(const),
                                 include.drift = T)
      ic_diff[i,] <- c(const, 1, p, ADF_est_diff[[i]]$aic,
                                    ADF_est_diff[[i]]$bic)
    }
  }
}

ic_aic_diff <- ic_diff[order(ic_diff[,4]),][1:10,]
ic_bic_diff <- ic_diff[order(ic_diff[,5]),][1:10,]

print(ic_aic_diff)
```



```
##       cons trend p       aic       bic
##  [1,]    1     0 1 -1374.441 -1361.110
##  [2,]    1     0 0 -1373.402 -1363.404
##  [3,]    1     1 1 -1372.788 -1356.125
##  [4,]    1     0 2 -1372.466 -1355.803
##  [5,]    1     1 0 -1371.969 -1358.638
##  [6,]    1     1 2 -1370.838 -1350.842
##  [7,]    1     0 3 -1370.571 -1350.574
##  [8,]    1     0 5 -1369.103 -1342.441
##  [9,]    1     1 3 -1368.922 -1345.593
## [10,]    1     0 4 -1368.730 -1345.400
```

```
print(ic_bic_diff)
```

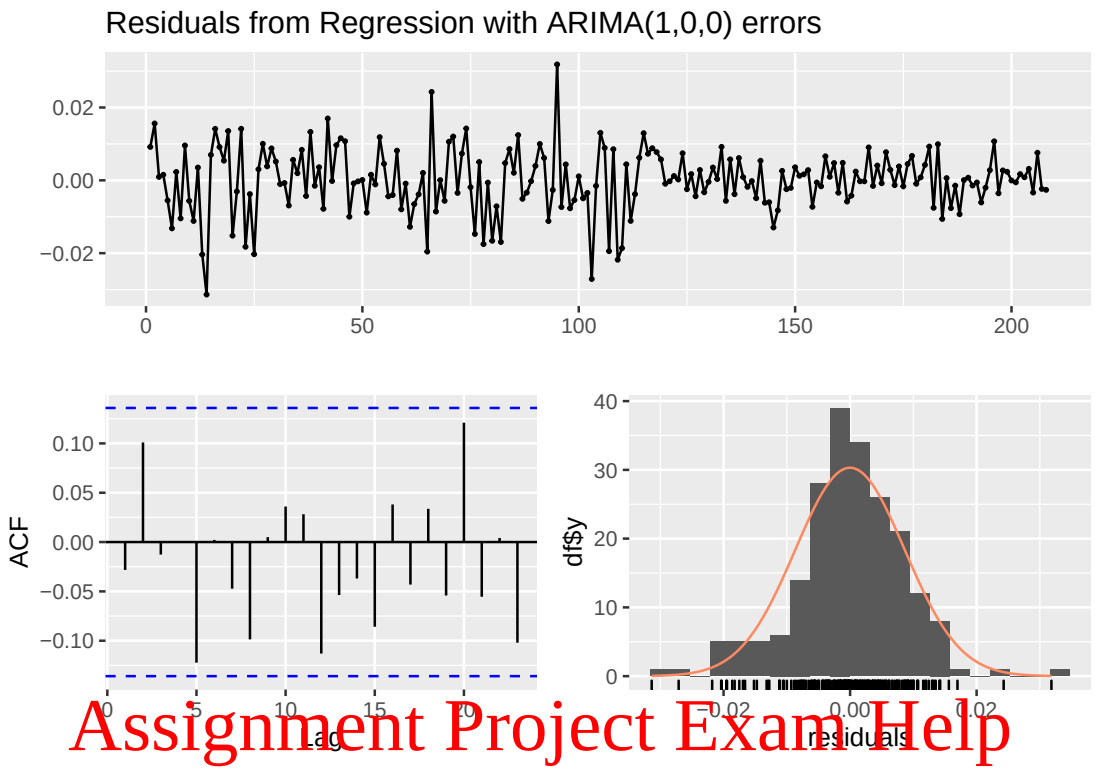
```
##       cons trend p       aic       bic
##  [1,]    1     0 0 -1373.402 -1363.404
##  [2,]    1     0 1 -1374.441 -1361.110
##  [3,]    1     1 0 -1371.969 -1358.638
##  [4,]    1     1 1 -1372.788 -1356.125
##  [5,]    1     0 2 -1372.466 -1355.803
##  [6,]    1     1 2 -1370.838 -1350.842
##  [7,]    1     0 3 -1370.571 -1350.574
##  [8,]    1     1 3 -1368.922 -1345.593
##  [9,]    1     0 4 -1368.730 -1345.400
## [10,]    1     0 5 -1369.103 -1342.441
```

Note that in this case the top five specifications ranked by the AIC is the same as the top five ranked by the BIC. Since they agree, we chose these five to form the adequate set: $p = 0, 1, 2$ with constant and no trend along with $p = 0, 1$ with both a constant and trend.

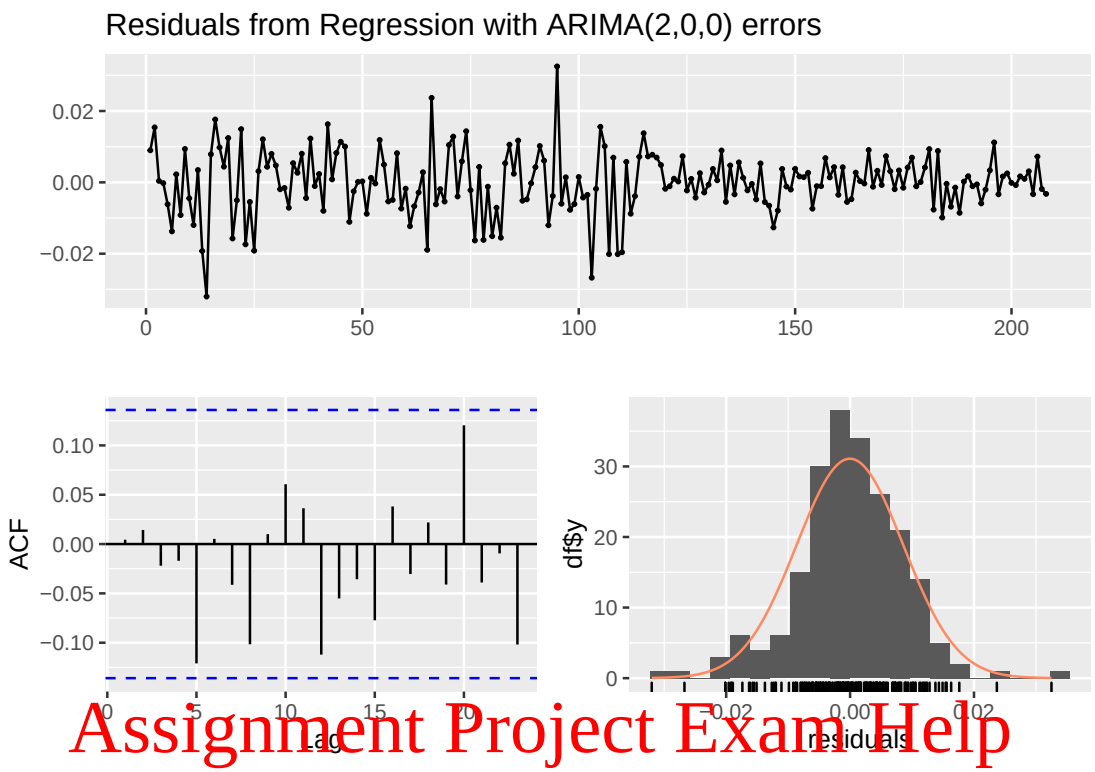

```
adq_set_diff <- as.matrix(arrange(as.data.frame(
                          ic_bic_diff[1:5,]), const, trend, p))
adq_idx_diff <- match(data.frame(t(adq_set_diff[, 1:3])),
                      data.frame(t(ic_diff[, 1:3])))
```


Finally, we check the residuals for all specifications in the adequate set.

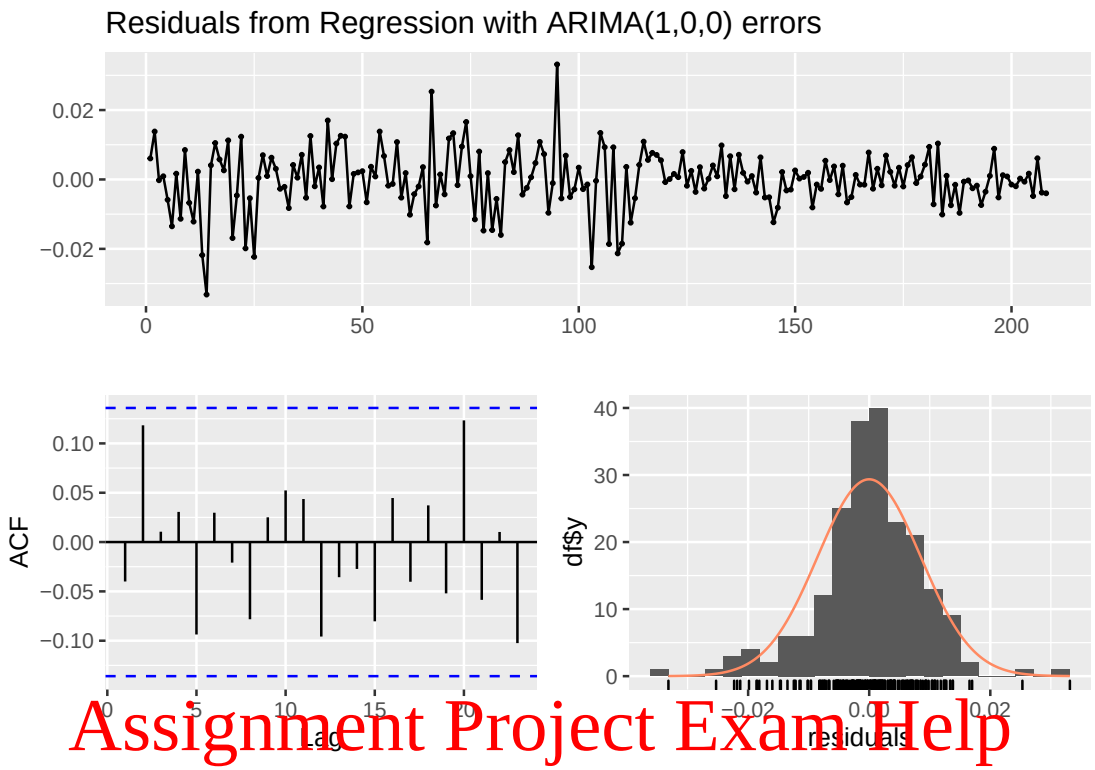

```
for (i in 1:length(adq_idx))
{
  checkresiduals(ADF_est[[adq_idx[i]]])
}
```



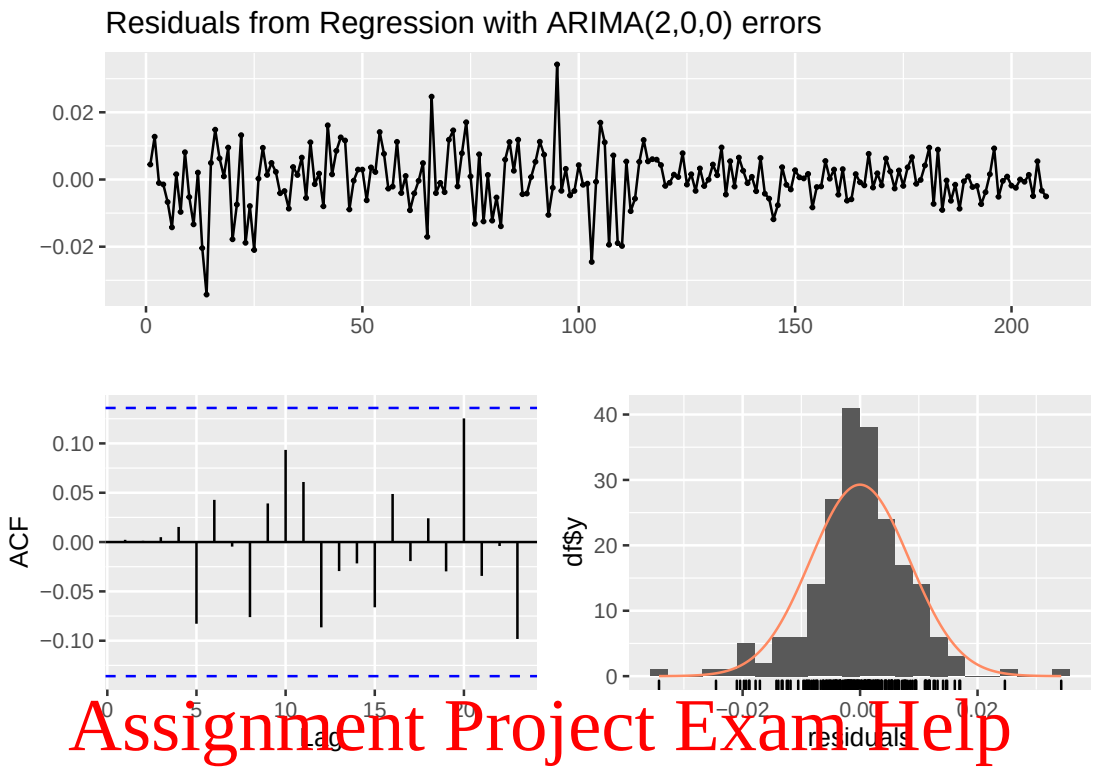

```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,0,0) errors
## Q* = 8.4811, df = 7, p-value = 0.2921
##
## Model df: 3.   Total lags used: 10
```



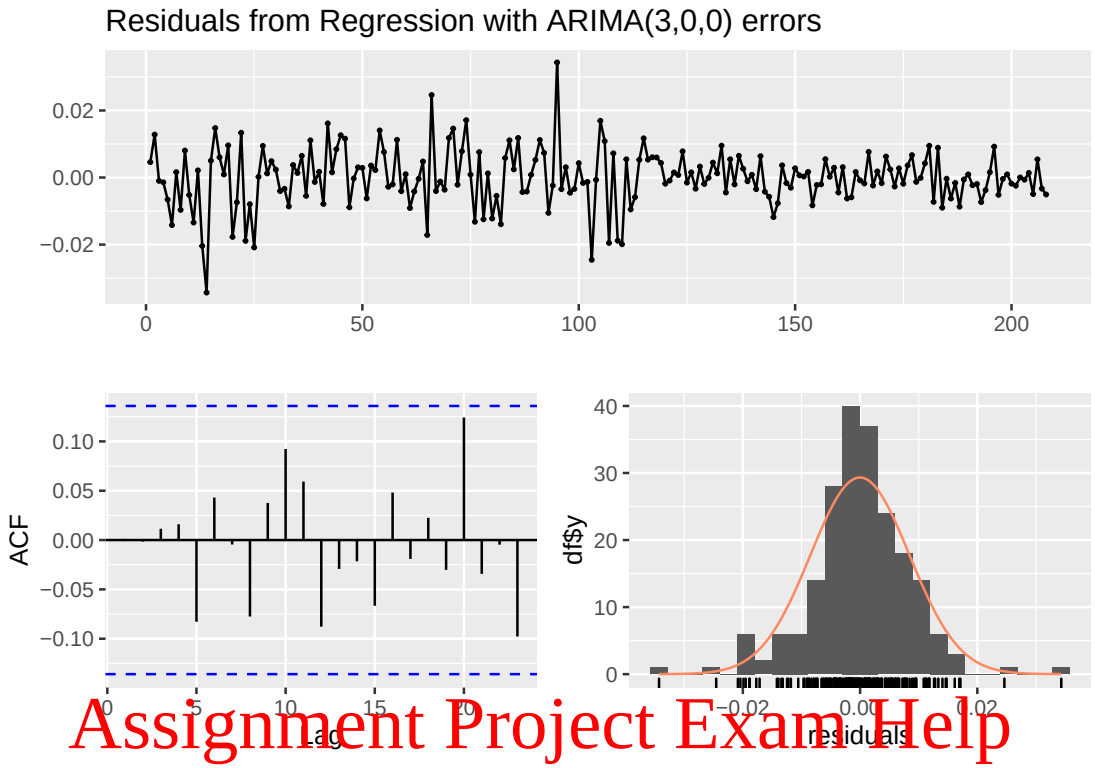

```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(2,0,0) errors
## Q* = 6.822, df = 6, p-value = 0.3376
##
## Model df: 4.   Total lags used: 10
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,0,0) errors
## Q* = 7.7825, df = 6, p-value = 0.2545
##
## Model df: 4.   Total lags used: 10
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(2,0,0) errors
## Q* = 5.4569, df = 5, p-value = 0.3627
##
## Model df: 5.   Total lags used: 10
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(3,0,0) errors
## Q* = 5.466, df = 4, p-value = 0.2427
##
## Model df: 6.   Total lags used: 10
```


As no obvious problems jump out from the residuals analysis, we proceed with ADF test using specifications in the adequate set.

```
adf.test(diff(y))
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -7.13    0.01
## [2,]   1 -5.05    0.01
## [3,]   2 -4.30    0.01
## [4,]   3 -3.73    0.01
## [5,]   4 -3.42    0.01
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0 -10.58    0.01
```

```
## [2,]   1  -7.88    0.01
## [3,]   2  -7.21    0.01
## [4,]   3  -6.67    0.01
## [5,]   4  -6.66    0.01
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0 -10.60    0.01
## [2,]   1  -7.88    0.01
## [3,]   2  -7.21    0.01
## [4,]   3  -6.67    0.01
## [5,]   4  -6.68    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

A unit root is rejected at very small significance level for all specifications. Hence, the differenced process is empirically distinguishable from an integrated process.

(e) Interpret the overall findings in parts (d) and (d).

Solution Since $\{y_t\}$ is not empirically distinguishable from an integrated process, but $\{\Delta y_t\}$ is, we conclude that $\{y_t\}$ is not empirically distinguishable from an I(1) process. Remember, however, that we *did not* find evidence of $\{y_t\}$ being I(1) exactly!

(f) Construct an adequate set of ARIMA(p, d, q) models using information criteria and residuals analysis.

Solution We will consider models for $p = 0, .., 3$; $q = 0, ..., 3$; $d = 0, 1$ and either with or without constant and/or trend terms. There are 96 models to estimate altogether. We use the `Arima` function in a nested `for` loop to automate the estimation. There are two caveats to deal with.

- The `Arima` function with $d = 1$ will only specify an intercept when setting the `include.drift = T` option. Since we want to incude a linear growth term (i.e. t as a regressor) in the differenced specification, we need to pass it as an exogenous variable to `Arima` through the `xreg` option. However, with $d = 1$, `Arima` will also difference whatever data we pass this way, so we need to *cummulatively sum* t before passing it.

- Some specifications will be so bad that MLE will run into numerical problems and return an error. We want to ignore these specifications in the least disruptive way possible. The way to do it is to embed `Arima` as an argument to the `try` function with the `silent = T` option. This in general a very useful programming technique when automating a large number of steps that may potentially cause errors.

```
TT <- length(y)
ARIMA_est <- list()
ic_arima <- matrix( nrow = (2 * 2 + 2) * 4 ^ 2, ncol = 7 )
colnames(ic_arima) <- c("d", "cons", "trend", "p", "q", "aic",
                                                        "bic")
i <- 0
for (d in 0:1)
{
  for (const in 0:1)
  {
    for (p in 0:3)
    {
      for (q in 0:3)
      {
        i <- i + 1
        d1 <- as.logical(d)
        c1 <- as.logical(const)

        try(silent = T, expr =
        {
        ARIMA_est[[i]] <- Arima(y, order = c(p, d, q),
                                include.constant = c1)

        ic_arima[i,] <- c(d, const, 0, p, q,
                          ARIMA_est[[i]]$aic,
                          ARIMA_est[[i]]$bic)
        })

        if (const)
        {
          # only add a specification with trend if there is a
          # constant (i.e., exclude no constant with trend)
          i <- i + 1

          if (d1)
          {
            x <- c(0,cumsum(1:(TT - 1)))
          }
          else
```

```
          {
            x <- NULL
          }

          try(silent = T, expr =
          {
          ARIMA_est[[i]] <- Arima(y, order = c(p, d, q),
                                  xreg = x,
                                  include.constant = c1,
                                  include.drift = T)

          ic_arima[i,] <- c(d, const, 1, p, q,
                            ARIMA_est[[i]]$aic,
                            ARIMA_est[[i]]$bic)
          })
        }
      }
    }
  }
}

ic_aic_arima <- ic_arima[order(ic_arima[,6]),][1:10,]
ic_bic_arima <- ic_arima[order(ic_arima[,7]),][1:10,]

print(ic_aic_arima)
```



```
##       d cons trend p q       aic       bic
##  [1,] 0    1     1 3 0 -1385.796 -1365.742
##  [2,] 0    1     1 2 1 -1385.080 -1365.026
##  [3,] 0    1     1 2 0 -1384.489 -1367.778
##  [4,] 0    1     1 1 2 -1383.951 -1363.897
##  [5,] 0    1     1 2 2 -1383.938 -1360.542
##  [6,] 0    1     1 3 1 -1383.853 -1360.456
##  [7,] 0    1     1 3 3 -1383.816 -1353.735
##  [8,] 1    1     1 3 1 -1383.743 -1360.380
##  [9,] 0    1     1 1 3 -1383.558 -1360.162
## [10,] 0    1     1 2 3 -1382.031 -1355.293
```

```
print(ic_bic_arima)
```

```
##       d cons trend p q       aic       bic
##  [1,] 1    1     0 1 0 -1379.386 -1369.373
##  [2,] 0    1     1 2 0 -1384.489 -1367.778
##  [3,] 1    1     0 2 0 -1379.434 -1366.084
##  [4,] 0    1     1 3 0 -1385.796 -1365.742
##  [5,] 1    1     0 0 2 -1379.046 -1365.696
```

```
##  [6,] 1    1     0 1 1 -1378.786 -1365.436
##  [7,] 1    1     0 0 1 -1375.166 -1365.153
##  [8,] 0    1     1 2 1 -1385.080 -1365.026
##  [9,] 1    1     1 1 0 -1378.280 -1364.930
## [10,] 0    1     1 1 2 -1383.951 -1363.897
```

The AIC generally prefers models with $d = 0$, while the BIC top 10 includes a more varied mix of integrated and non-integrated ARMAs. It also seems helpful in this case to compute the intersecting set of the top 10 AIC and top 10 BIC ranked specifications.

```
ic_int_arima <- intersect(as.data.frame(ic_aic_arima),
                          as.data.frame(ic_bic_arima))

print(ic_int_arima)
```

```
##   d cons trend p q       aic       bic
## 1 0    1     1 3 0 -1385.796 -1365.742
## 2 0    1     1 2 1 -1385.080 -1365.026
## 3 0    1     1 2 0 -1384.489 -1367.778
## 4 0    1     1 1 2 -1383.951 -1363.897
```

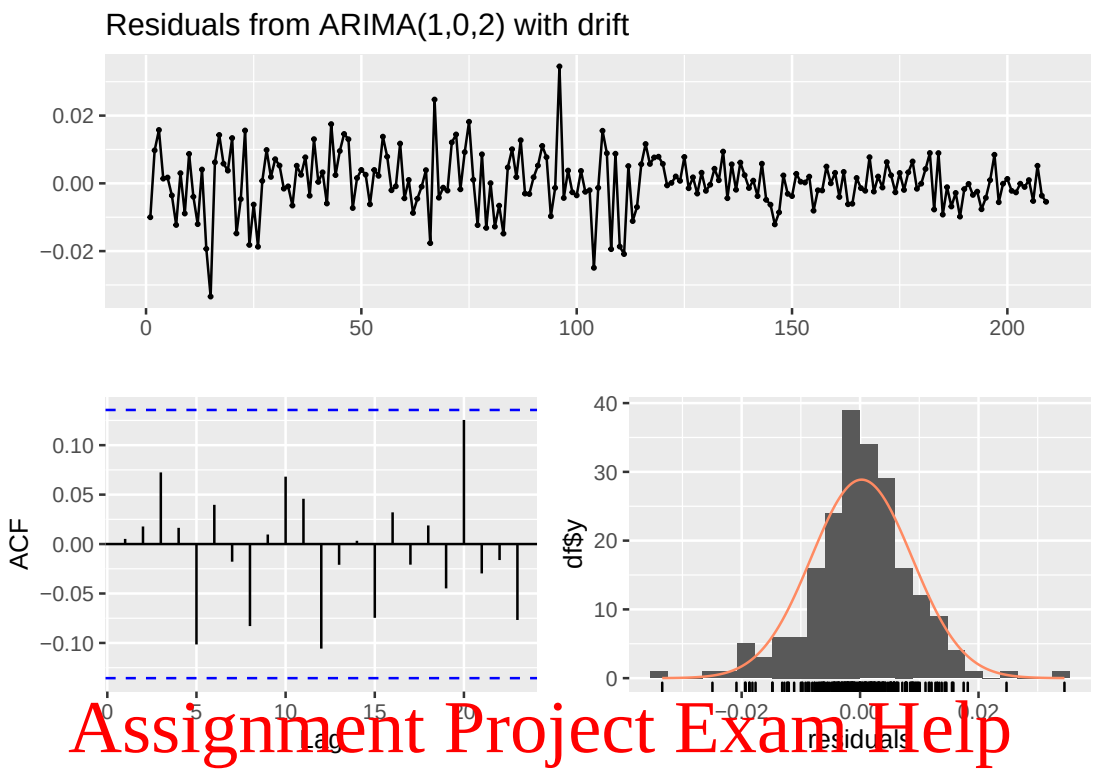

We observe that the intersection contains only specifications in levels (i.e. $d = 0$). However, given that a number of integrated specifications are in the top 10 ranked by the BIC as well as our inference that $\{y_t\}$ is not empirically distinguishable from an I(1) process, it is worth taking a closer look to see if any specifications in differences (i.e., $d = 1$) are worth considering.

We make note of the fact that ARIMA$(1, 1, 0)$ and ARIMA$(2, 1, 0)$—both with a constant only—are in the top three of the BIC ranking. In light of this and the above considerations, we will add ARIMA$(1, 1, 0)$ and ARIMA$(2, 1, 0)$ to the four models in the intersecting set.

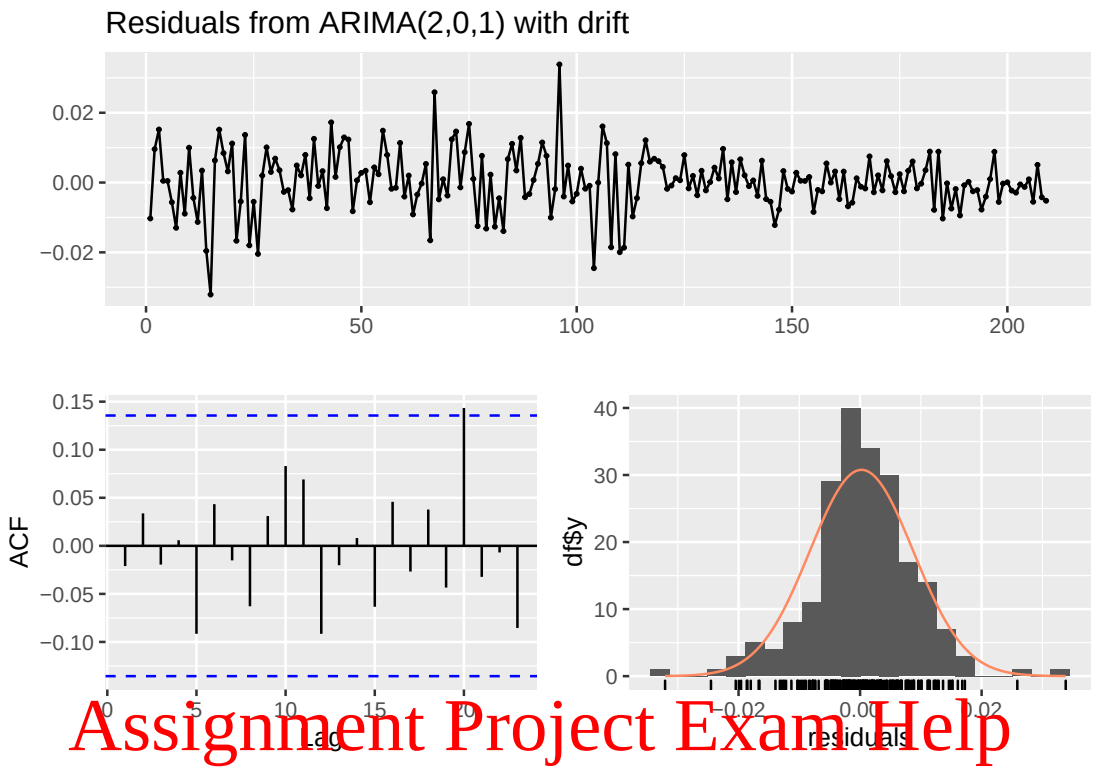
```
adq_set_arima <- as.matrix(arrange(as.data.frame(
                           rbind(ic_int_arima,
                                 ic_bic_arima[c(1, 3),])),
                                      d, const, trend, p))
adq_idx_arima <- match(data.frame(t(adq_set_arima[, 1:5])),
                       data.frame(t(ic_arima[, 1:5])))
```

Finally, we check the residuals for every model in the adequate set.

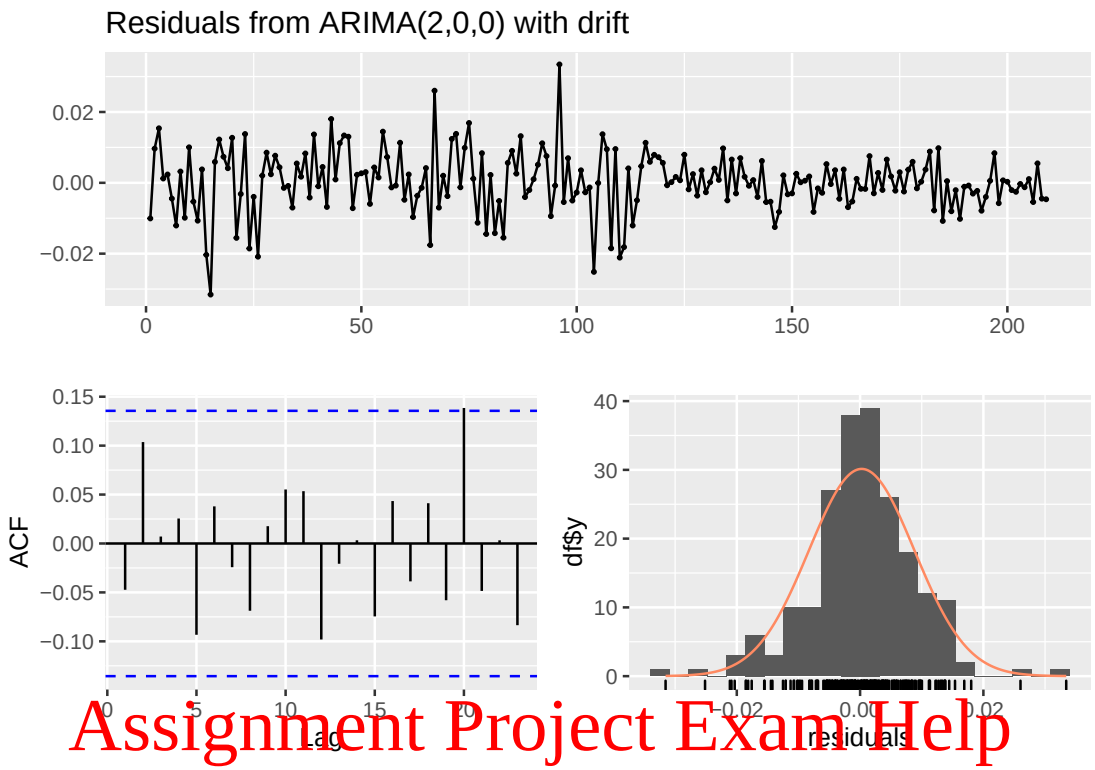
```
for (i in 1:length(adq_idx_arima))
{
  checkresiduals(ARIMA_est[[adq_idx_arima[i]]])
}
```



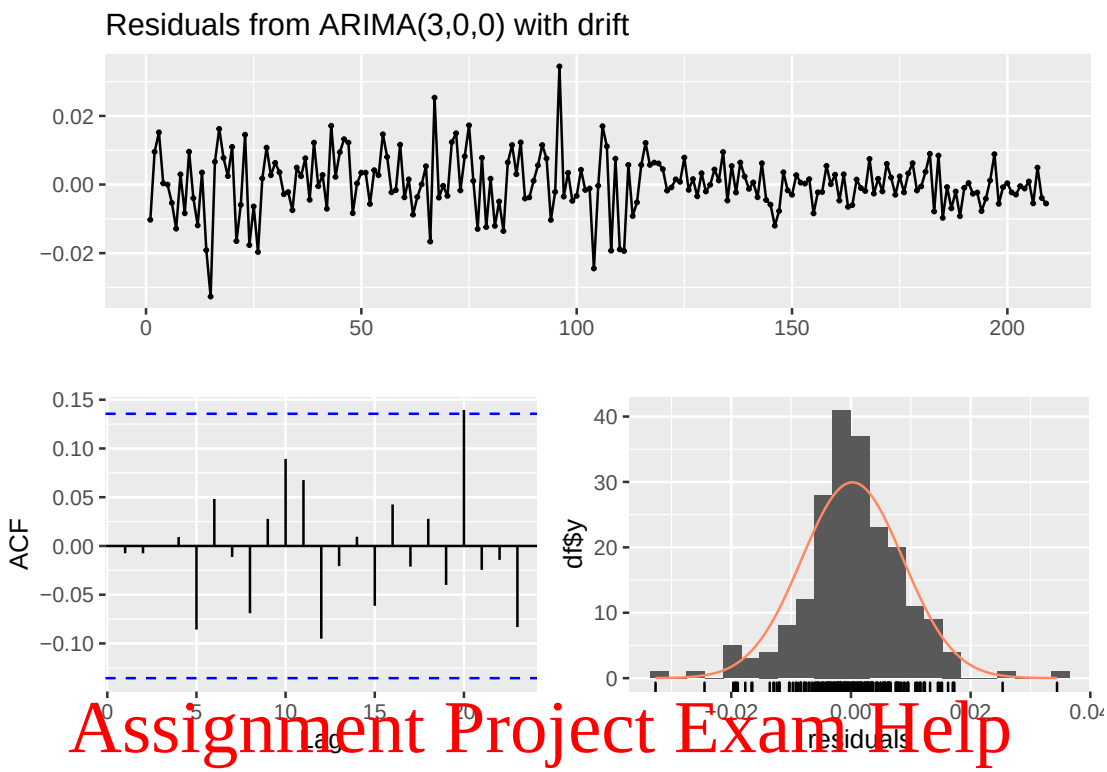

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,2) with drift
## Q* = 6.4524, df = 5, p-value = 0.2647
##
## Model df: 5.   Total lags used: 10
```



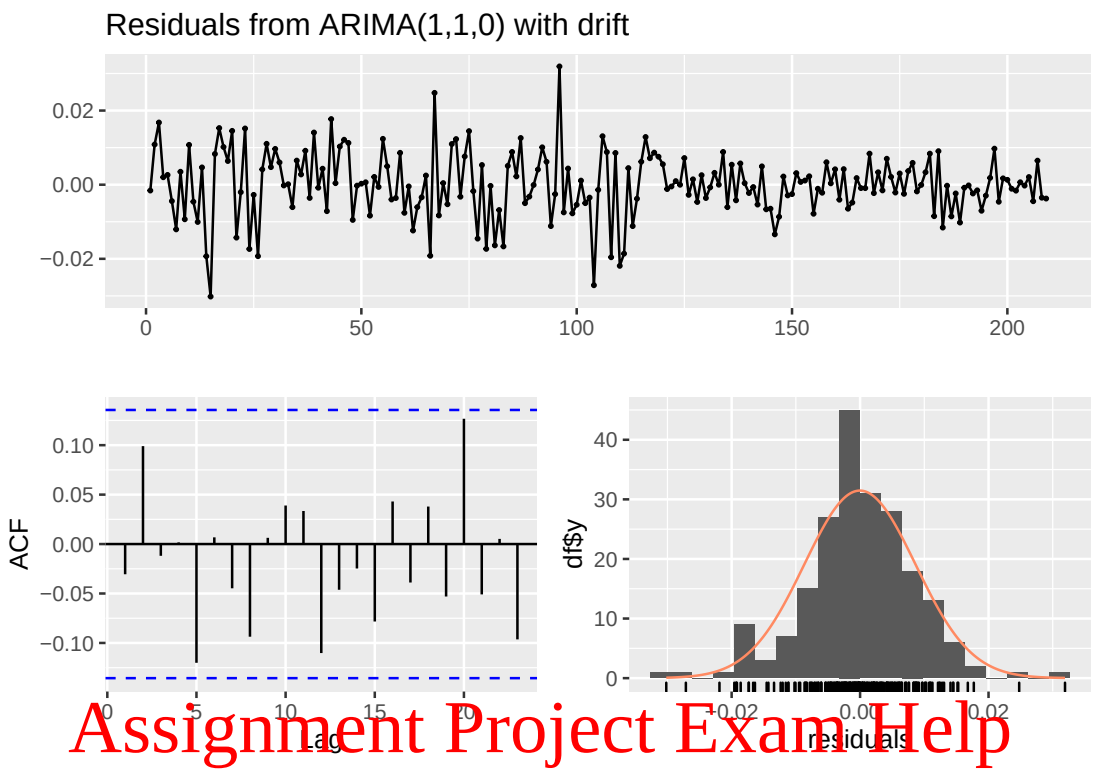

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,1) with drift
## Q* = 5.3012, df = 5, p-value = 0.3802
##
## Model df: 5.   Total lags used: 10
```



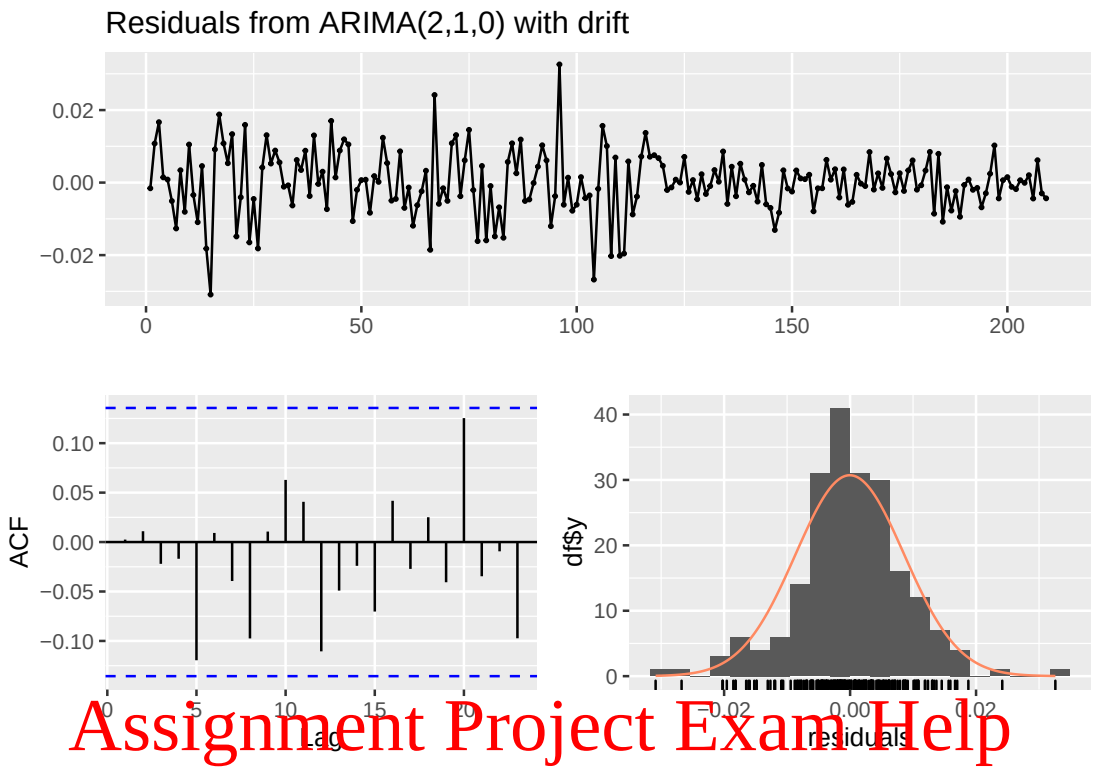

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,0) with drift
## Q* = 7.0155, df = 6, p-value = 0.3194
##
## Model df: 4.   Total lags used: 10
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,0,0) with drift
## Q* = 5.1452, df = 5, p-value = 0.3984
##
## Model df: 5.   Total lags used: 10
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,0) with drift
## Q* = 8.1496, df = 8, p-value = 0.419
##
## Model df: 2.   Total lags used: 10
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,0) with drift
## Q* = 6.6179, df = 7, p-value = 0.4697
##
## Model df: 3.   Total lags used: 10
```

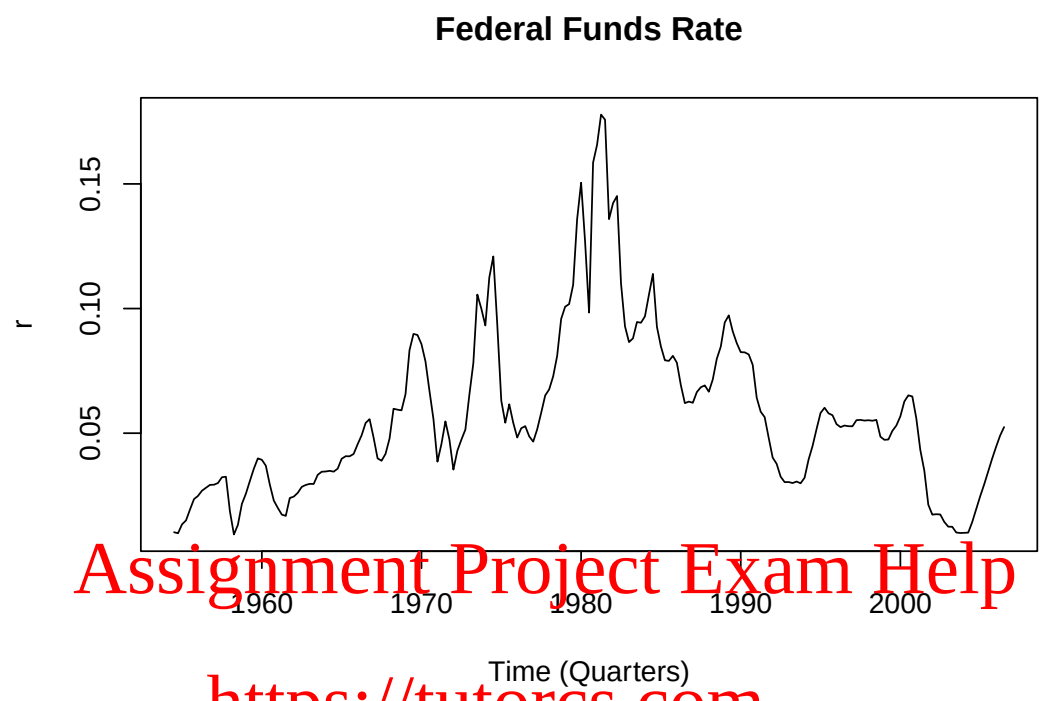

All the residuals look fine so our construction of the adequate set is complete.

---

2. Repeat parts (a)-(e) of Question 1 for r_t (you do not need to do part (f)).

Solution The approach is nearly identical to that involving y in Question 1. The parts where we need to pay particular attention is qualitatively assessing the information in AIC and BIC ranking tables as well as results of residuals analysis.

```
plot(dates, r, type = "l", xlab = "Time (Quarters)",
     main = "Federal Funds Rate")
```




```
TT <- length(r)
ADF_est <- list()
ic <- matrix(nrow = 30, ncol = 5)
colnames(ic) <- c("cons", "trend", "p", "aic", "bic")
i <- 0
for (const in 0:1)
{
  for (p in 0:9)
  {
    i <- i + 1
    ADF_est[[i]] <- Arima(diff(r), xreg = r[-TT],
                          order = c(p, 0, 0),
                          include.mean = as.logical(const),
                          include.drift = F)
    ic[i,] <- c(const, 0, p, ADF_est[[i]]$aic,
                ADF_est[[i]]$bic)
  }

  if (const)
  {
    # only add a specification with trend if there is a
    # constant (i.e., exclude no constant with trend)
```

```
    for (p in 0:9)
    {
      i <- i + 1
      ADF_est[[i]] <- Arima(diff(r), xreg = r[-TT],
                            order = c(p, 0, 0),
                            include.mean = as.logical(const),
                            include.drift = T)
      ic[i,] <- c(const, 1, p, ADF_est[[i]]$aic,
                  ADF_est[[i]]$bic)
    }
  }
}

ic_aic <- ic[order(ic[,4]),][1:10,]
ic_bic <- ic[order(ic[,5]),][1:10,]
ic_int <- intersect(as.data.frame(ic_aic),
                    as.data.frame(ic_bic))

print(ic_aic)
```



```
##       cons trend p       aic       bic
##  [1,]    1     0 7 -1379.096 -1345.720
##  [2,]    1     0 8 -1377.897 -1341.184
##  [3,]    1     1 7 -1377.218 -1340.505
##  [4,]    0     0 7 -1376.067 -1346.029
##  [5,]    1     1 8 -1375.956 -1335.906
##  [6,]    1     0 9 -1375.907 -1335.856
##  [7,]    1     0 6 -1374.959 -1344.922
##  [8,]    0     0 8 -1374.456 -1341.081
##  [9,]    1     1 9 -1373.974 -1330.586
## [10,]    1     1 6 -1372.964 -1339.588
```



```
print(ic_bic)
```

```
##       cons trend p       aic       bic
##  [1,]    1     0 3 -1369.151 -1349.126
##  [2,]    0     0 2 -1361.784 -1348.434
##  [3,]    1     0 2 -1364.814 -1348.126
##  [4,]    0     0 3 -1364.796 -1348.108
##  [5,]    1     0 1 -1360.578 -1347.228
##  [6,]    0     0 7 -1376.067 -1346.029
##  [7,]    0     0 1 -1356.003 -1345.990
##  [8,]    1     0 7 -1379.096 -1345.720
##  [9,]    1     0 5 -1371.988 -1345.288
## [10,]    1     0 6 -1374.959 -1344.922
```

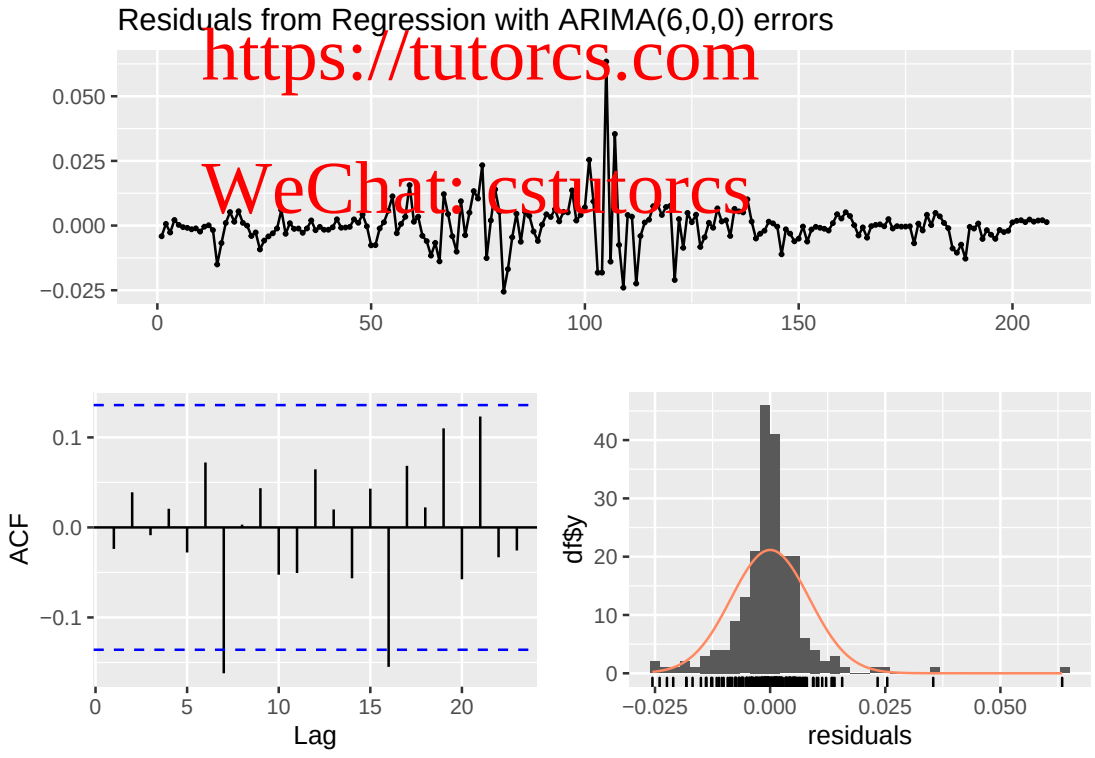
```
print(ic_int)
```

```
##   cons trend p       aic       bic
## 1    1     0 7 -1379.096 -1345.720
## 2    0     0 7 -1376.067 -1346.029
## 3    1     0 6 -1374.959 -1344.922
```

The intersecting set has three models: two with a constant and $p = 6, 7$, and one without a constant and $p = 7$; no models have a trend. We may reasonably set this to be the adequate set (although other variations are obviously justifiable as well). Importantly, we complete the procedure with a residuals analysis.

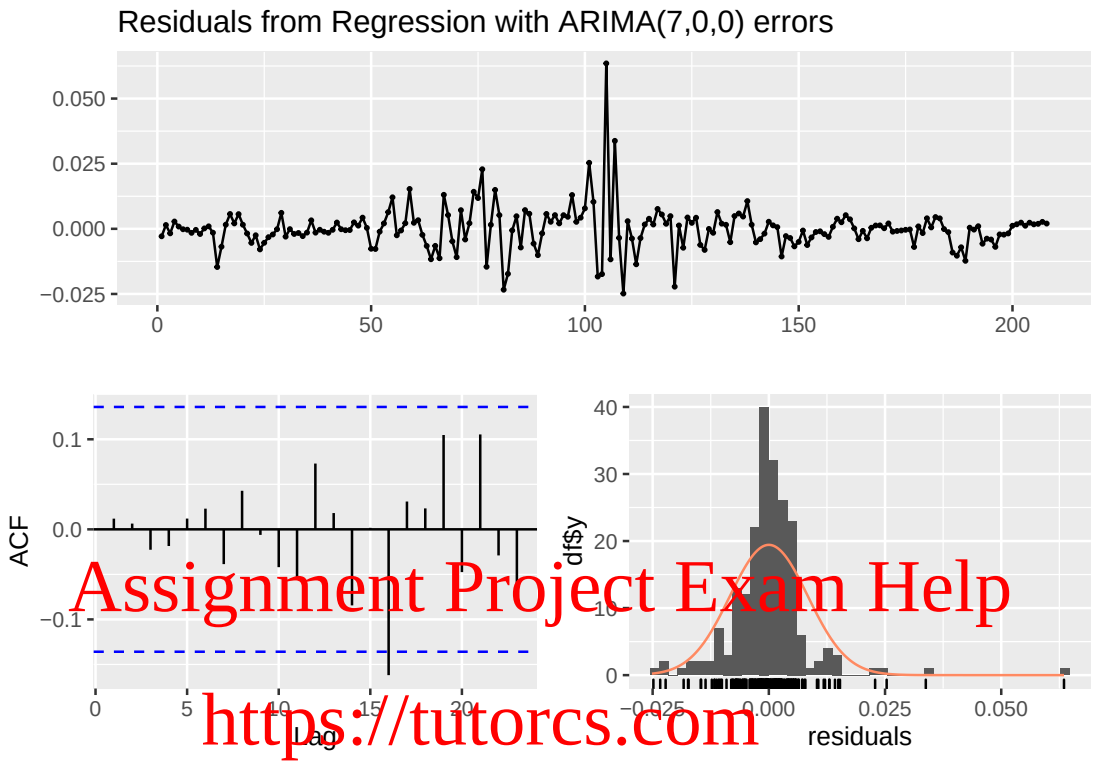
```
adq_set <- as.matrix(arrange(as.data.frame(ic_int),
                             const, trend, p))
adq_idx <- match(data.frame(t(adq_set[, 1:3])),
                 data.frame(t(ic[, 1:3])))

for (i in 1:length(adq_idx))
{
  checkresiduals(ADF_est[[adq_idx[i]]])
}
```

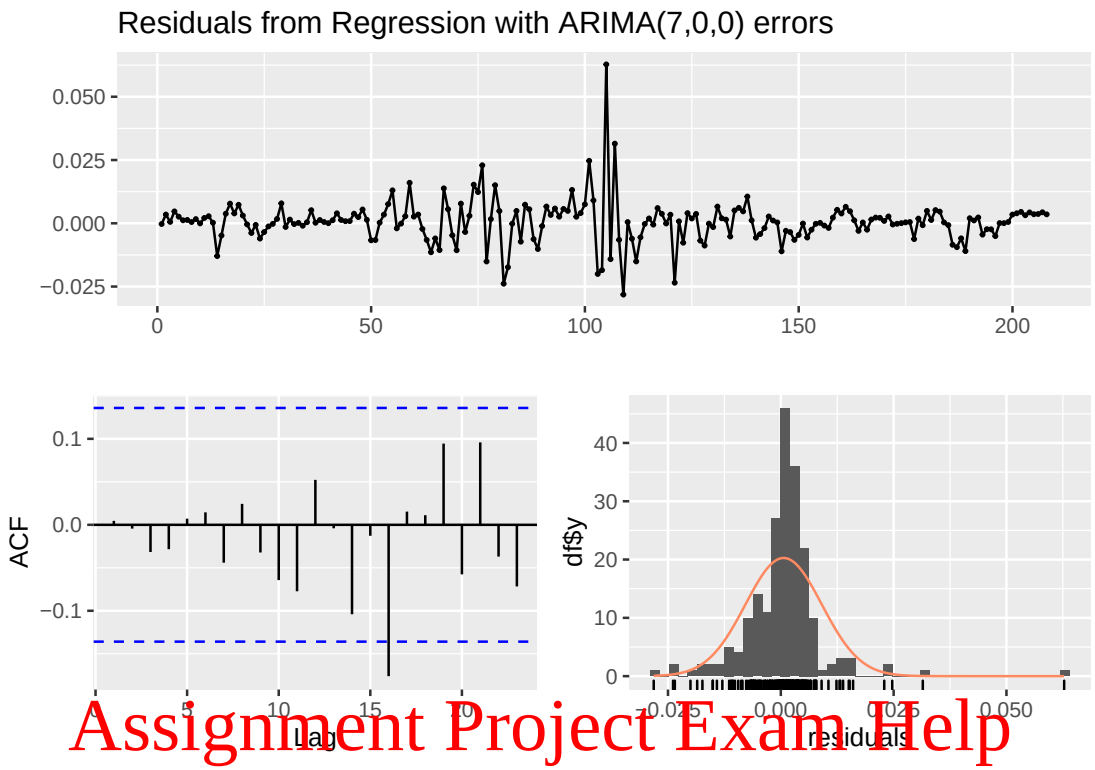




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(6,0,0) errors
```

```
## Q* = 9.1461, df = 3, p-value = 0.02741
##
## Model df: 8.   Total lags used: 11
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(7,0,0) errors
## Q* = 3.3278, df = 3, p-value = 0.3438
##
## Model df: 9.   Total lags used: 12
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(7,0,0) errors
## Q* = 3.4685, df = 3, p-value = 0.3249
##
## Model df: 8.   Total lags used: 11
```


Residuals all look reasonable, although some autocorrelations exceed the 95% confidence intervals very slightly (and at larger lags). For robustness, we might add a few models with a larger p into the adequate set, such as $p = 8, 9$.

Proceeding the the ADF test, the default option $p \leq 5$ implemented in the `adf.test` function is insuffcient, so we need to explicitly specify longer lag lengths with the `nlag` option.

```
adf.test(r, nlag = 10)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##       lag    ADF p.value
##  [1,]   0 -0.884   0.362
##  [2,]   1 -1.190   0.253
##  [3,]   2 -0.937   0.344
##  [4,]   3 -1.120   0.278
```

```
##  [5,]   4 -1.110   0.281
##  [6,]   5 -1.324   0.205
##  [7,]   6 -1.087   0.290
##  [8,]   7 -0.871   0.367
##  [9,]   8 -0.908   0.354
## [10,]   9 -0.880   0.364
## Type 2: with drift no trend
##       lag   ADF p.value
##  [1,]   0 -2.30  0.2106
##  [2,]   1 -2.86  0.0549
##  [3,]   2 -2.40  0.1734
##  [4,]   3 -2.76  0.0714
##  [5,]   4 -2.71  0.0788
##  [6,]   5 -3.16  0.0244
##  [7,]   6 -2.70  0.0804
##  [8,]   7 -2.26  0.2244
##  [9,]   8 -2.37  0.1833
## [10,]   9 -2.31  0.2055
## Type 3: with drift and trend
##       lag   ADF p.value
##  [1,]   0 -2.24  0.4757
##  [2,]   1 -2.81  0.2363
##  [3,]   2 -2.34  0.4328
##  [4,]   3 -2.71  0.2769
##  [5,]   4 -2.67  0.2935
##  [6,]   5 -3.14  0.0992
##  [7,]   6 -2.66  0.2974
##  [8,]   7 -2.20  0.4895
##  [9,]   8 -2.31  0.4439
## [10,]   9 -2.25  0.4693
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```


Only "Type 1" and "Type 2" specifications are in our adequate set, so we ignore the output related to "Type 3". For all specifications in our adequate set, the null of a unit root cannot be rejected.

Note, however, that there are specifications for which the null is indeed rejected. One such specification includes a constant, but no trend contains $p = 5$ lags. We observe that this is the 9th ranked specification by the BIC, and it is outside the top 10 in terms of AIC.

We can summarise our inference as follows. The process is not empirically distinguishable from an integrated process with a drift, but there is a small element of uncertainty in this conclusion.

Next, we repeat the construction of an adequate set of ADF regressions of Δr_t.

```
TT <- length(diff(r))
ADF_est_diff <- list()
ic_diff <- matrix( nrow = 30, ncol = 5 )
colnames(ic_diff) <- c("cons", "trend", "p", "aic", "bic")
i <- 0
for (const in 0:1)
{
  for (p in 0:9)
  {
    i <- i + 1
    ADF_est_diff[[i]] <- Arima(diff(diff(r)),
                               xreg = diff(r)[-TT],
                               order = c(p, 0, 0),
                               include.mean = as.logical(const),
                               include.drift = F)
    ic_diff[i,] <- c(const, 0, p, ADF_est_diff[[i]]$aic,
                     ADF_est_diff[[i]]$bic)
  }
  if (const)
  {
    # only add a specification with trend if there is a
    # constant (i.e., exclude no constant with trend)
    for (p in 0:9)
    {
      i <- i + 1
      ADF_est_diff[[i]] <- Arima(diff(diff(r)),
                                 xreg = diff(r)[-TT],
                                 order = c(p, 0, 0),
                                 include.mean = as.logical(const),
                                 include.drift = T)
      ic_diff[i,] <- c(const, 1, p, ADF_est_diff[[i]]$aic,
                       ADF_est_diff[[i]]$bic)
    }
  }
}

ic_aic_diff <- ic_diff[order(ic_diff[,4]),][1:10,]
ic_bic_diff <- ic_diff[order(ic_diff[,5]),][1:10,]
ic_int_diff <- intersect(as.data.frame(ic_aic_diff),
                         as.data.frame(ic_bic_diff))

print(ic_aic_diff)
```

```
##       cons trend p       aic       bic
```

```
##  [1,]    0     0 6 -1368.762 -1342.100
##  [2,]    0     0 7 -1367.974 -1337.980
##  [3,]    1     0 6 -1366.834 -1336.839
##  [4,]    1     0 7 -1366.045 -1332.718
##  [5,]    0     0 8 -1365.990 -1332.662
##  [6,]    1     1 6 -1365.403 -1332.076
##  [7,]    0     0 9 -1364.708 -1328.048
##  [8,]    1     1 7 -1364.569 -1327.909
##  [9,]    1     0 8 -1364.061 -1327.401
## [10,]    1     0 9 -1362.778 -1322.785
```

```
print(ic_bic_diff)
```

```
##       cons trend p       aic       bic
##  [1,]    0     0 2 -1357.902 -1344.572
##  [2,]    0     0 0 -1349.094 -1342.429
##  [3,]    0     0 6 -1368.762 -1342.100
##  [4,]    0     0 3 -1356.028 -1339.364
##  [5,]    1     0 2 -1355.968 -1339.305
##  [6,]    0     0 4 -1358.595 -1338.599
##  [7,]    0     0 5 -1361.729 -1338.400
##  [8,]    0     0 1 -1348.330 -1338.332
##  [9,]    0     0 7 -1367.974 -1337.980
## [10,]    1     0 0 -1347.158 -1337.160
```

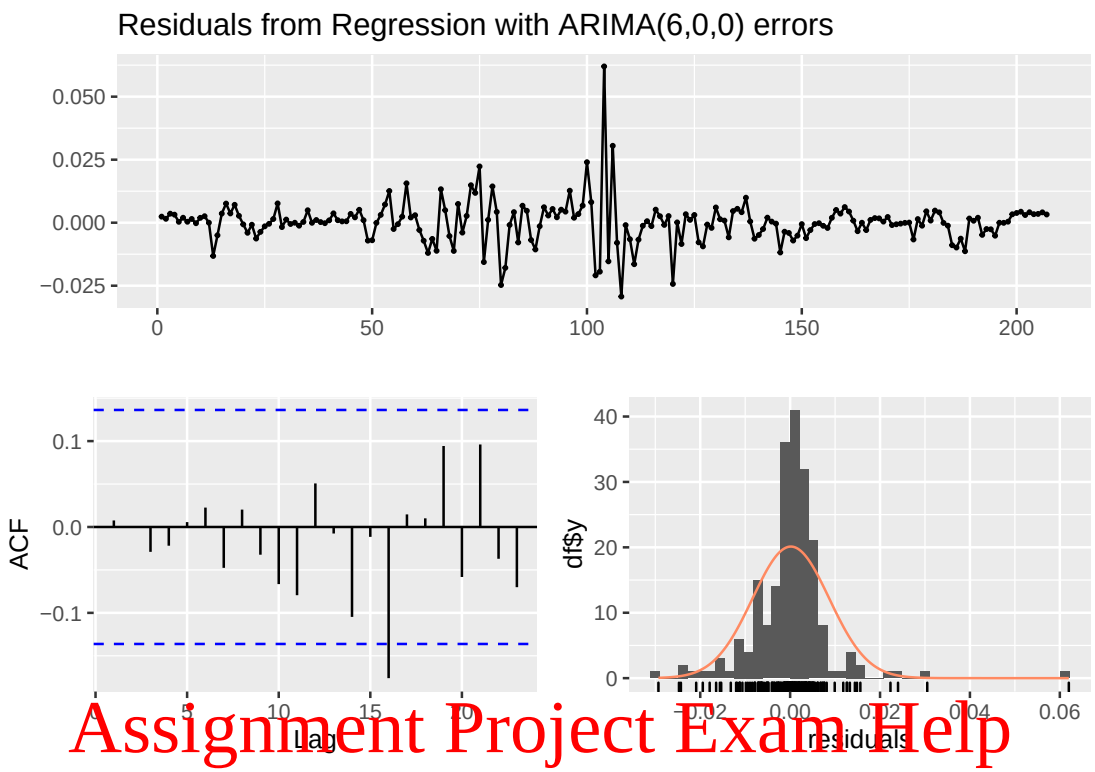
```
print(ic_int_diff)
```

```
##   cons trend p       aic      bic
## 1    0     0 6 -1368.762 -1342.10
## 2    0     0 7 -1367.974 -1337.98
```

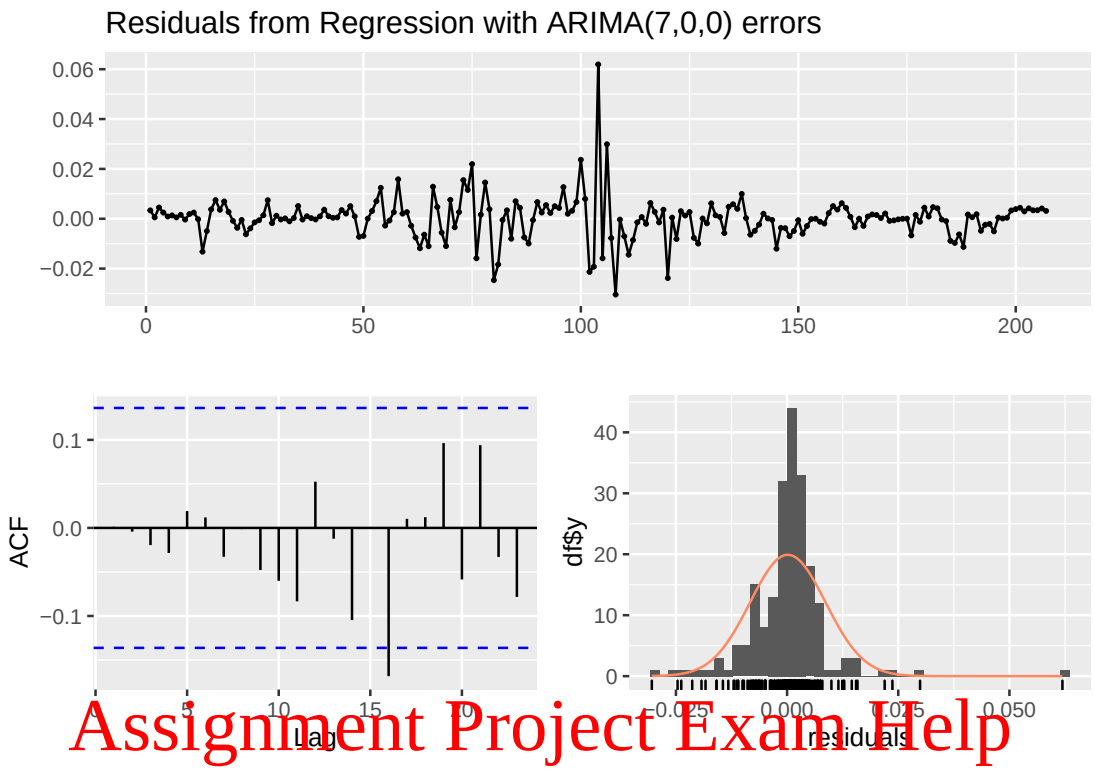

There are only two specifications in the intersecting set: both with no constant or trend and $p = 6, 7$. Overall, both AIC and BIC favour specifications without a trend, and AIC generally favours larger lag lengths, whereas the BIC yields a more dispersed ranking. Taking the intersection as the adquate set, we proceed to the residuals analysis.

```
adq_set_diff <- as.matrix(arrange(
                           as.data.frame(ic_int_diff),
                                         const, trend, p))
adq_idx_diff <- match(data.frame(t(adq_set_diff[, 1:3])),
                      data.frame(t(ic_diff[, 1:3])))

for (i in 1:length(adq_idx_diff))
{
  checkresiduals(ADF_est_diff[[adq_idx_diff[i]]])
}
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(6,0,0) errors
## Q* = 2.1855, df = 3, p-value = 0.5348
##
## Model df: 7.   Total lags used: 10
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(7,0,0) errors
## Q* = 3.424, df = 3, p-value = 0.3308
##
## Model df: 8.   Total lags used: 11
```

As with levels, residuals look ok, but with some indication that longer lag lengths should be considered. Finally, we implement the ADF test.

```
adf.test(diff(r), nlag = 10)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##       lag    ADF p.value
##  [1,]   0 -11.28    0.01
##  [2,]   1 -10.80    0.01
##  [3,]   2  -7.36    0.01
##  [4,]   3  -6.59    0.01
##  [5,]   4  -5.07    0.01
##  [6,]   5  -5.70    0.01
##  [7,]   6  -6.43    0.01
##  [8,]   7  -5.56    0.01
```

```
##  [9,]   8  -5.28    0.01
## [10,]   9  -5.40    0.01
## Type 2: with drift no trend
##       lag    ADF p.value
##  [1,]   0 -11.26    0.01
##  [2,]   1 -10.78    0.01
##  [3,]   2  -7.35    0.01
##  [4,]   3  -6.58    0.01
##  [5,]   4  -5.06    0.01
##  [6,]   5  -5.69    0.01
##  [7,]   6  -6.42    0.01
##  [8,]   7  -5.55    0.01
##  [9,]   8  -5.27    0.01
## [10,]   9  -5.39    0.01
## Type 3: with drift and trend
##       lag    ADF p.value
##  [1,]   0 -11.25    0.01
##  [2,]   1 -10.78    0.01
##  [3,]   2  -7.35    0.01
##  [4,]   3  -6.58    0.01
##  [5,]   4  -5.05    0.01
##  [6,]   5  -5.69    0.01
##  [7,]   6  -6.43    0.01
##  [8,]   7  -5.56    0.01
##  [9,]   8  -5.28    0.01
## [10,]   9  -5.42    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```


A unit root is rejected at very small significance level for all specifications. We infer that FFR is not empirically distinguishable from an I(1) process.
